PROCESS DATA ANALYSIS AND INTERPRETATION

J. F. Davis

Department of Chemical Engineering, Ohio State University, Columbus, Ohio 43210

M. J. Piovoso

DuPont, Wilmington, Delaware 19880

K. A. Hoo

Department of Chemical Engineering, University of South Carolina, Columbia, South Carolina 29208

B. R. Bakshi

Department of Chemical Engineering, Ohio State University, Columbus, Ohio 43210

ADVANCES IN CHEMICAL ENGINEERING, VOL. 25 0065-2377/00 $30.00

I. Introduction

A. Overview

The recent, unprecedented ease with which process data can be collected and recorded is the use of data to support increasingly sophisticated process management activities such as increasing process efficiency, improving product quality, identifying hazardous conditions, and managing abnormal situations. To support these activities, the raw process data must be transformed into meaningful descriptions of process conditions. As shown in Fig. 1, the data analysis and interpretation techniques that affect the required transformation can be viewed as a classical pattern recognition problem with two primary tasks: *data analysis* (or feature extraction), which consists of numerically processing the data to produce numerical features of interest, and *data interpretation* (or feature mapping), which consists of assigning symbolic interpretations (i.e., labels) to numerical features.

The general pattern recognition problem can be described as follows. The input data, or pattern X, is defined by a number of specific data measurements, x_i, defined at a particular point in time:

$$X = (x_1, x_2, \ldots, x_i, \ldots, x_d). \tag{1}$$

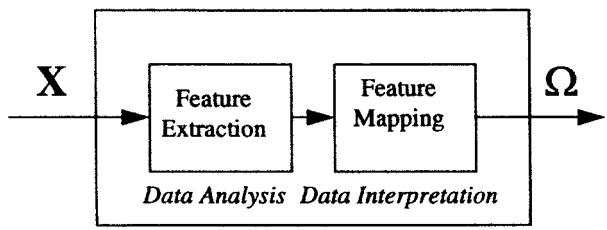


FIG. 1. Pattern recognition view of data analysis and interpretation.

The representation space, Ξ, consists of the set of all possible patterns, $X \in \Xi$. The set of all possible class labels ω_j forms the interpretation space Ω:

$$\Omega = (\omega_1, \omega_2, \ldots, \omega_j, \ldots, \omega_p). \tag{2}$$

Pattern recognition then corresponds to determining the numeric-symbolic mapping, ζ, from the representation space to the interpretation space:

$$\zeta: \Xi \rightarrow \Omega \quad \text{or} \quad \zeta: (x_1, x_2, \ldots, x_d) \rightarrow \omega_j. \tag{3}$$

Typically, determining ζ is easier after extracting relevant features, Z, from X and thereby using a mapping of the form

$$\zeta: \{\zeta': (x_1, x_2, \ldots, x_d) \rightarrow (z_1, z_2, \ldots, z_f)\} \rightarrow \omega_j. \tag{4}$$

Using this notation, X corresponds to any time series of data with x_i being a sampled value, and Z represents the processed forms of the data (i.e., a pattern). The z_i are the pattern features, ω_j is the appropriate label or interpretation, ζ' is the feature extraction or data analysis transformation, and ζ is the mapping or interpretation that must be developed.

The primary purpose of pattern recognition is to determine class membership for a set of numeric input data. The performance of any given approach is ultimately driven by how well an appropriate discriminant can be defined to resolve the numeric data into a label of interest. Because of both the importance of the problem and its many challenges, significant research has been applied to this area, resulting in a large number of techniques and approaches. With this publication, we seek to provide a common framework to discuss the application of these approaches.

B. Data Analysis

The objective of data analysis (or feature extraction) is to transform numeric inputs in such a way as to reject irrelevant information that can confuse the information of interest and to accentuate information that supports the feature mapping. This usually is accomplished by some form of *numeric–numeric* transformation in which the numeric input data are transformed into a set of numeric features. The numeric–numeric transformation makes use of a process model to map between the input and the output.

Models are generally built from either fundamental knowledge about a system or empirical data collected from a system. Models based on fundamental knowledge attempt to directly predict actual plant behavior. Therefore, they can be especially useful for those operating situations that have not been previously observed. However, accurate fundamental models are

difficult to build for many chemical processes because the underlying phenomena are often not well understood. Thus, empirical models are an extremely popular approach to map between input and output variables. In terms of feature extraction, a model captures the relationship between data features and labels of process behaviors.

As with any empirical technique, the development of a feature mapping scheme is an inherently inductive process that requires a good knowledge source to produce discriminants relevant to the numeric features to be interpreted and the labels of interest. Determination of the mapping requires a set of reference patterns with correct class labels. This *training set* consists of multiple *pattern exemplars*. The distinguishing features used in generating the empirical model can be expressed either explicitly or implicitly, depending on the underlying technique.

Data analysis techniques can be viewed as either *input mapping* or *input–output mapping,* as shown in Fig. 2. The transformations are illustrated in Fig. 3. *Input mapping* refers to manipulation of input data, X, without consideration of any output variables or features. This type of mapping is generally used to transform the input data to a more convenient representation, Z', while retaining all the essential information of the original data. *Input–output mapping* extracts important features, Z, by relating input variables, X, output variables, Y, or features of interest, Z, together. This type of mapping may include an implicit input transformation step for reducing input dimensionality.

Input mapping methods can be divided into *univariate, multivariate,* and *probabalistic* methods. *Univariate* methods analyze the inputs by extracting the relationship between the *measurements*. These methods include various types of single-scale and multiscale filtering such as exponential smoothing, wavelet thresholding, and median filtering. *Multivariate* methods analyze

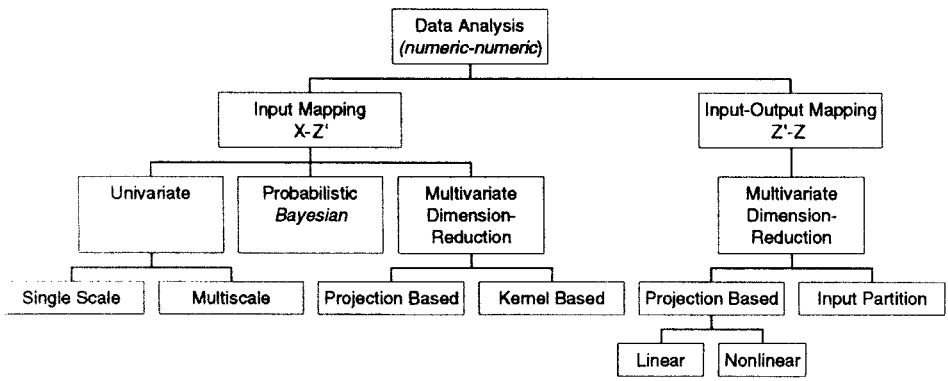


FIG. 2. A classification of data analysis methods.

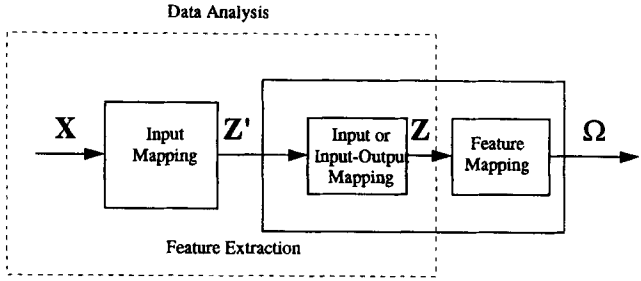


FIG. 3. Data analysis applied to data interpretation.

the inputs by extracting the relationship between the *variables*. These methods reduce dimensionality and include both projection and kernel based approaches, such as principal component analysis and clustering. All share the characteristic of transforming a given matrix of data without information on any output variables. As will be discussed, these methods either produce or support the extraction of features that can be used to assign labels. As shown in Fig. 3, it is often useful to apply an input mapping approach to preprocess data to produce a more useful representation of the input data. Input–output methods map input numeric data to a transformed set of numeric features *with* knowledge of the output variables. These methods can be classified based on the method used to transform the input space (linear projection, nonlinear projection, or partition). Methods based on *linear projection* combine the inputs as a linear weighted sum. These methods include popular empirical modeling methods such as partial least squares (PLS) regression (both linear and nonlinear) and back propagation networks (BPNs) with a single hidden layer. Methods based on *nonlinear projection* can be divided further into *local* and *nonlocal* methods. Nonlocal methods have at least one infinite dimension in a projection surface. Local methods include such approaches as radial basis function networks (RBFNs) or Adaptive Resonance Theory (ART), whereas nonlocal methods include approaches like BPNs with multiple hidden layers. *Subset selection* or *partition-based* methods include classification and regression trees (CARTs) and multivariate adaptive regression splines (MARSs).

C. DATA INTERPRETATION

In this publication, the purpose of data analysis is to drive toward data interpretation that consists of assigning various types of labels. These label

types include *state descriptions* (e.g., normal or high), *trends* (e.g., increasing or pulsing), *landmarks* (e.g., process change), *shape descriptions* (e.g., skewed, tail), and *fault descriptions* (e.g., flooding or contamination). Assignment of labels requires that feature extraction be extended by establishing decision discriminants based on labeled (known and observed) situations of sufficiently similar characteristics (i.e., *pattern exemplars*). The combination of feature extraction and feature mapping specifically for label assignment is called *numeric–symbolic mapping,* as shown in Fig. 4. A number of approaches integrate both the feature extraction and feature mapping aspects of numeric–symbolic mapping. Although both aspects are occurring in these methods, the individual steps cannot be distinguished. The resultant symbolic output of a numeric–symbolic interpreter may itself need further interpretation. This requires a *symbolic–symbolic* mapper such as a knowledge-based system (KBS) that can further refine symbolic interpretations. Figure 4 indicates this as a mapping from Ω' to Ω.

Feature mapping (i.e., numeric–symbolic mapping) requires decision mechanisms that can distinguish between possible label classes. As shown in Fig. 5, widely used decision mechanisms include linear discriminant surfaces, local data cluster criteria, and simple decision limits. Depending on the nature of the features and the feature extraction approaches, one or more of these decision mechanisms can be selected to assign labels.

Labels are distinguished based on whether they are context dependent or context-free. *Context-dependent* labels require simultaneous consideration of time records from more than one process variable; *context-free* labels do not. Thus, generating context-free trend, landmark, and fault descriptions is considerably more simple than generating context-dependent descriptions. Context-free situations can take advantage of numerous methods for common, yet useful, interpretations. Context-dependent situations, however, require the application of considerable process knowledge to get a useful interpretation. In these situations, performance is dependent on the availability, coverage, and distribution of labeled process data from

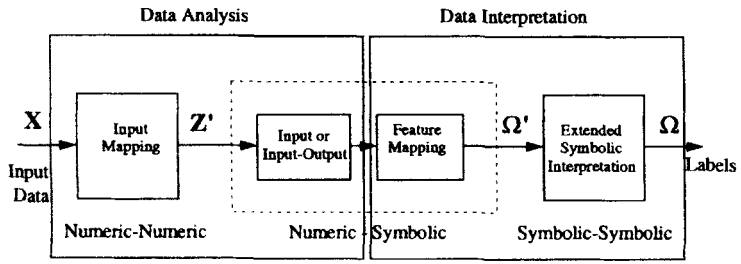


FIG. 4. The numeric–symbolic and symbolic–symbolic views of data interpretation.

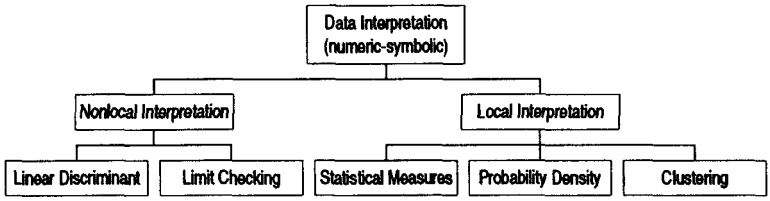


FIG. 5. Label class decision methods for data interpretation.

known process situations. Because of the complexity of chemical processes, most labels in this domain are context dependent.

Context-dependent situations often lead to a large-scale input dimension. Because the required number of training examples increases with the number of measured variables or features, reducing the input dimensionality may improve system performance. In addition, decision discriminants will be less complex (because of fewer dimensions in the data) and more easily determined. The reduction in dimensionality can be most readily achieved by eliminating redundancy in the data so that only the most relevant features are used for mapping to a given set of labels.

In addition to a large input dimension, complex processes also typically have a large output dimension (i.e., many possible plant behaviors). Although only a few events occur at any given time, identification of those few events requires consideration of a large subset of all possible events. Without careful management of both input and output dimensionality, interpretation performance deteriorates rapidly with scale and complexity.

D. CHALLENGES

Several significant challenges exist in applying data analysis and interpretation techniques to industrial situations. These challenges include (1) the scale (amount of input data) and scope (number of interpretations) of the problem, (2) the scarcity of abnormal situation exemplars, (3) uncertainty in process measurements, (4) uncertainty in process discriminants, and (5) the dynamic nature of process conditions.

1. Process Scale and Scope. Within the chemical process industry, a typical operator console will collect 2000 to 5000 measurements every time increment, typically 1 minute or less. Despite this large amount of process data being generated, relatively few events occur at any single time. The data must be used to determine which, out of a typical range of 300 to 600 possible descriptions (process labels), reflect current conditions. In addition to these "final" process labels, many useful, intermediate labels may also

be developed. Thus, any approach must be able to manage the large dimensionality of the problem.

2. Lack of Abnormal Situation Exemplars. Because chemical processes are highly controlled, labeled sensor patterns that are available for inductive learning techniques are limited to primarily normal operating conditions. Although the abundance of process data provides for many process exemplars, very few are available for general abnormal situations—even fewer are available for a specific fault. Because most approaches to data analysis and interpretation depend heavily on using known patterns to define decision boundaries, the limited availability of these data severely affect performance and resolution. Ideally, the data would be well distributed (particularly in the vicinity of pattern class boundaries) so as to minimize uncertainty in the boundary regions.

3. Uncertainty in Process Measurements. Sensor measurements are always subject to noise, calibration error, and temporary signal loss, as well as various faults that may not be immediately detected. Therefore, data preprocessing will often be required to overcome the inherent limitations of the process measurements.

4. Uncertainty in Process Discriminants. Because processes operate over a continuum, data analysis generally produces distinguishing features that exist over a continuum. This is further compounded by noise and errors in the sensor measurements. Therefore, the discriminants developed to distinguish various process labels may overlap, resulting in uncertainty between data classes. As a result, it is impossible to define completely distinguishing criteria for the patterns. Thus, uncertainty must be addressed inherently.

5. Dynamic Changes to Process Conditions. Because of the dynamic nature of chemical processes, conditions change over time as a result of changes in production rates, quality targets, feed compositions, and equipment conditions. This in turn changes how certain process information may be interpreted. Thus, adaptive interpretation ability is often required.

E. Overview of Discussion

Data analysis and interpretation appears overwhelmingly complex when viewed as a single, unilateral approach (i.e., a single feature extraction step and a single interpretation step). This is because practical considerations demand (1) modularization to manage the input and output scale associated with a complex plant, (2) integration of a variety of data interpretation methods to address differing data analysis requirements throughout a plant, and (3) the use of multiple interpreters to map to specific labels.

Because the techniques for data analysis and interpretation are targeted to address different process characteristics, care must be taken in choosing the most appropriate set of techniques. For example, some techniques work best with abundant process data; others, with limited process data. Some can handle highly correlated data, while others cannot. In selecting appropriate methods, two practical considerations stand out:

1. Availability of good quality and adequate quantity of training data is essential for all empirical modeling methods.
2. Selecting the best method for a given task is not straightforward. It requires significant insight into the methods and the task.

In this chapter, we focus on recent and emerging technologies that either are or soon will be applied commercially. Older technologies are discussed to provide historic perspective. Brief discussions of potential future technologies are provided to indicate current development directions. The chapter substantially extends an earlier publication (Davis *et al.*, 1996a) and is divided into seven main sections beyond the introduction: Data Analysis, Input Analysis, Input–Output Analysis, Data Interpretation, Symbolic–Symbolic Interpretation, Managing Scale and Scope of Large-Scale Process Operations, and Comprehensive Examples.

Data Analysis presents a unifying framework for discussing and comparing input and input–output methods.

Input Analysis addresses input mapping approaches that transform input data without knowledge of or interest in output variables.

Input–Output Analysis considers feature extraction in the context of empirical modeling approaches.

Data Interpretation extends data analysis techniques to label assignment and considers both integrated approaches to feature extraction and feature mapping and approaches with explicit and separable extraction and mapping steps. The approaches in this section focus on those that form numeric–symbolic interpreters to map from numeric data to specific labels of interest.

Symbolic–Symbolic Intepretation addresses knowledge-based system (KBS) extensions to numeric–symbolic interpreters.

Managing Scale and Scope of Large-Scale Process Operations discusses the performance issues associated with data analysis and interpretation that occur as the scale of the problem increases (such as in complex process operations).

Finally, the chapter illustrates the integrated application of multiple analysis and interpretation technologies with several illustrative process examples.

This chapter provides a complementary perspective to that provided by Kramer and Mah (1994). Whereas they emphasize the statistical aspects of the three primary process monitoring tasks, data rectification, fault detection, and fault diagnosis, we focus on the theory, development, and performance of approaches that combine data analysis and data interpretation into an automated mechanism via feature extraction and label assignment.

II. Data Analysis: A Unifying Framework

Data analysis methods transform inputs into more meaningful and concentrated sources of information. As discussed in the introduction, measured variables can be analyzed by input or input–output transformation methods. These methods include a wide variety of linear and nonlinear modeling methods developed across a wide range of technical areas, including statistics, process simulation, control, and intelligent systems. Although the two categories of methods are quite different with respect to consideration of output variables, greater insight into similarities and differences can be achieved by comparing them in the context of a unified framework.

More specifically, input data analysis methods are similar to input–output methods, but rely on different strategies for extracting the relevant information. With reference to the general expression in Eq. (4), the resulting analyzed or latent variable for all input methods can be represented as

$$z_m = \phi_m(\alpha; x_1, \ldots, x_d), \tag{5}$$

where $x_1, \ldots, x_d$ are the input variables, α is the matrix of input transformation parameters, ϕ_m is the mth input transformation function, and z_m is a latent or transformed variable. The objective of input analysis methods is to determine α and ϕ to satisfy a selected optimization criterion. Univariate input analysis methods transform a single input variable at a time, whereas multivariate methods transform all inputs simultaneously. Equation (5) indicates that input analysis methods require decisions about the nature of the input transformation, ϕ, and the optimization criteria used for determining both α and ϕ. By definition, input analysis methods do not consider the outputs for determining the latent variables. Because the ultimate objective of data analysis and interpretation is to relate inputs to output labels, input analysis methods require an additional interpretation step to map the latent variables to the labels.

Sometimes more meaningful features may be obtained by considering the behavior of both input and output variables together in the analysis.

Typically, input–output analysis methods extract the most relevant signal features by relating the analyzed variables to process output variables, y_i,

$$y_i = \sum_{m=1}^{M} \beta_{mk}\theta_m(\phi_m(\alpha; x_1, x_2, \ldots, x_J)), \tag{6}$$

where y_i is the ith predicted output or response variable, θ_m is the mth basis or activation function, and β_{mk} is the output weight or regression coefficient relating the mth basis function to the kth output. Equation (6) indicates that in addition to decisions about the nature of the input transformation and optimization criteria for the input transformation, input–output analysis methods require decisions about the type of basis functions, θ_m, and additional optimization criteria for the parameters, β_{mk}, and basis functions. With regard to interpretation, the y_i, β_{mk}, or the implicit latent variables, z_m, can be used as features.

Specific data analysis methods can be derived from Eqs. (5) and (6) depending on decisions about the input transformation, type of activation or basis functions, and optimization criteria. These decisions form the basis of a common framework for comparing all empirical modeling methods (Bakshi and Utojo, 1999).

Based on the nature of input transformation, both input and input–output analysis methods can be classified into the following three categories:

- *Methods based on linear projection* exploit the linear relationship among inputs by projecting them on a linear hyperplane before applying the basis function (see Fig. 6a). Thus, the inputs are transformed in combination as a linear weighted sum to form the latent variables. Univariate input analysis is a special case of this category where the single variable is projected on itself.
- *Methods based on nonlinear projection* exploit the nonlinear relationship between the inputs by projecting them on a nonlinear hypersurface resulting in latent variables that are nonlinear functions of the inputs, as shown in Figs. 6b and 6c. If the inputs are projected on a localized hypersurface such as a hypersphere or hyperellipse, then the basis functions are local, depicted in Fig. 6c. Otherwise, the basis functions are nonlocal, as shown in Fig. 6b.
- *Partition-based methods* address dimensionality by selecting input variables that are most relevant to efficient empirical modeling. The input space is partitioned by hyperplanes that are perpendicular to at least one of the input axes, as depicted in Fig. 6d.

With reference to Eq. (6), an input transformation is determined by the function, ϕ_m, and parameters, α, whereas the model relating the trans-

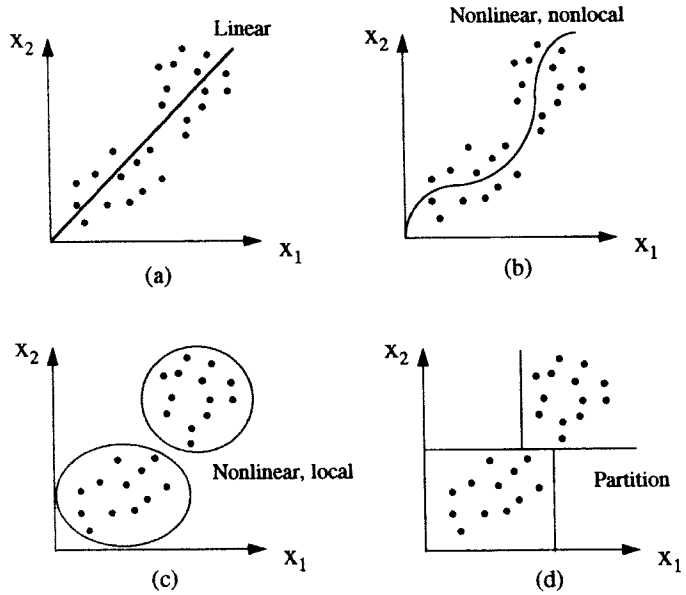


FIG. 6. Input transformation in (a) methods based on linear projection, (b) methods based on nonlinear projection, nonlocal transformation, (c) methods based on nonlinear projection, local transformation, and (d) partition-based methods. (From Bakshi and Utojo, 1998.)

formed inputs to the output is determined by the parameters, β_m, and basis functions, θ_m. Input–output methods often use different objective functions for determining the input transformation, and the transformed input–output relationship. This separation of optimization criteria provides explicit control over the dimensionality reduction by input transformation and often results in more accurate empirical models. Input–output methods therefore can be categorized depending on whether the optimization criterion for the input transformation contains information from

- *Only the inputs,* as in all input analysis methods
- *Both inputs and outputs,* as in input–output analysis and interpretation methods

An optimization criterion for determining the output parameters and basis functions is to minimize the output prediction error and is common to all input–output modeling methods. The activation or basis functions used in data analysis methods may be broadly divided into the following two categories:

- *Fixed-shape basis functions.* The basis functions are of a fixed shape, such as linear, sigmoid, Gaussian, wavelet, or sinusoid. Adjusting the

basis function parameters changes their location, size, and orientation, but their shape is decided a priori and remains fixed.

- *Adaptive-shape basis functions.* Some input–output modeling methods relax the fixed-shape requirement and allow the basis functions to adapt their shape, in addition to their location, size, and orientation, to the training and testing data. This additional degree of freedom provides greater flexibility in determining the unknown input–output surface and often results in a more compact model. Adaptive-shape basis functions are obtained through the application of smoothing techniques such as splines, variable span smoothers, and polynomials to approximate the transformed input–output space.

The nature of the input transformation, type of basis functions, and optimization criteria discussed in this section provide a common framework for comparing the wide variety of techniques for input transformation and input-output modeling. This comparison framework is useful for understanding the similarities and differences between various methods; it may be used to select the best method for a given task and to identify the challenges for combining the properties of various techniques (Bakshi and Utojo, 1999).

III. Input Analysis Methods

As stated, *input* analysis methods operate directly on the measurement input data set and map the numeric measurement input information into a modified numeric form. The objective is to emphasize the relevant process information that addresses or describes a particular set of behaviors while reducing or eliminating unwanted information. Input analysis methods belong to the categories of univariate and multivariate methods, depending on whether a single variable or multiple variables are analyzed. Univariate methods, also widely known as *filtering* methods, remove noise and errors based on their expected behavior in the time, frequency, or time–frequency domain. Multivariate methods reduce errors by exploiting redundancy in the variables. In either case, the principle of modeling the input data is the same: that is, to obtain a set of values that more clearly contain the relevant information present in the original set of measurements. In this section, we describe primary families of methods and illustrate their features with several specific examples. Consistent with the overall document, the objective of this section is to present key features of methods so it is clear how families of methods for input analysis relate to each other. We focus on two primary families: linear and nonlinear projection methods.

A. Methods Based on Linear Projection

Univariate input analysis or filtering is implemented as a mathematical algorithm where a data sequence corresponding to a single process variable, $x_i(n)$, is transformed by some mathematical computation into a different sequence, $z_m(n) = \phi_m x_i(n)$, where ϕ_m represents the mapping or transformation from the x_i into the new sequence z_m. A filtering operation usually generates a sequence z_m that is like x_i and maximizes the retention of relevant process information while minimizing the amount of interference that remains in the data. All filtering methods can be interpreted as analyzing data using a set of basis functions that are localized in the time and/or frequency space as

$$z_m = \sum h_i \Psi_i(s, u, \omega), \tag{7}$$

where $\psi_i(s, u, \omega)$ is the ith basis function at temporal location u, frequency location ω, and scale s. The scale parameter determines the extent of localization of the basis function in time and frequency. The input transformation function, Ψ_i, involves a linear or nonlinear transformation of the basis function coefficients, h_i. Key process considerations in determining filter characteristics are as follows:

- *Superposition* implies that whenever the filter input can be decomposed into a sum of two or more signals, the output can be formed as the sum of the responses of the filter to each of the components of the input acting one at a time.
- *Homogeneity* implies that a constant gain can be applied at either the input or the output of a system without changing the response.
- *Causality* requires that the filter response at time n be computed on the basis of present and past information and not require knowledge of either the future input or output. Thus, the computation of $z_m(n)$ involves only present and past (values of the) inputs and only past outputs.
- *Shift invariance* implies that the computational algorithm to determine the filter output does not change with time. Thus, a given sequence $x_i(n)$ produces the same output $z_m(n)$ regardless of the time at which it is applied.

Depending on how the previous measurements are combined in Eq. (7), univariate filtering methods can be categorized as linear or nonlinear. In terms of Eq. (7), linear filtering methods use a fixed scale parameter or are single-scale, whereas nonlinear filtering methods are multiscale. Figure 7 summarizes decompositions in terms of time and frequency.

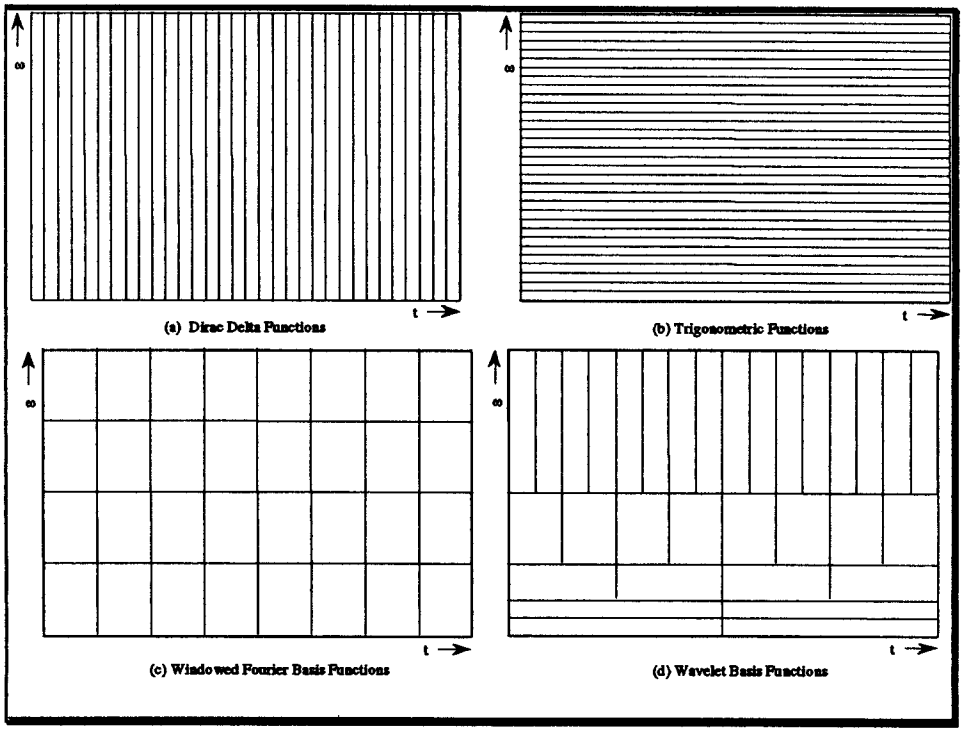


FIG. 7. Decomposition of the time (t)–frequenty (ω) by different basis functions.

1. *Linear or Single-Scale Filtering*

The general view just given offers a perspective on scale filtering methods that are in wide use. Newer methods expand on these.

In single-scale filtering, basis functions are of a fixed resolution and all basis functions have the same localization in the time–frequency domain. For example, frequency domain filtering relies on basis functions localized in frequency but global in time, as shown in Fig. 7b. Other popular filters, such as those based on a windowed Fourier transform, mean filtering, and exponential smoothing, are localized in both time and frequency, but their resolution is fixed, as shown in Fig. 7c. Single-scale filters are linear because the measured data or basis function coefficients are transformed as their linear sum over a time horizon. A finite time horizon results in *finite impulse response* (FIR) and an infinite time horizon creates infinite impulse response (IIR) filters. A linear filter can be represented as

$$z_m = \sum_{i=1}^{T} w_i x(n-i), \tag{8}$$

where z_m is the transformed variable containing the information of interest for data interpretation. The w_i are the weights, T is the length of the time horizon. This equation is called the *convolution sum*. In practice, most FIR and IIR filters are linear, causal, and shift-invariant and are characterized by an *impulse response*, w_i, that is of finite duration and has zero value for $i < 0$ and $i > T$, the filter impulse response duration. An impulse response is defined as the output of a filter when the input is a sequence that is zero everywhere except for $x_i(0) = 1$ (the sequence element at $n = 0$). Knowledge of the impulse response is sufficient to specify the behavior of the filter. The impulse response can be interpreted as a weighting function that determines how past inputs affect the present filter output.

For a mean filter with a window width of T, the impulse response is given by

$$w_i = \begin{cases} \dfrac{1}{T}, & i = 1, \ldots T; \quad 0 \text{ otherwise.} \end{cases} \tag{9}$$

This filter produces a response at time n that is the average of the present plus $(n - 1)$ previous values of the input x_i.

For an IIR filter, the parameter T in Eq. (9) tends to infinity. IIR filters can be represented as a function of previous filter outputs and often can be computed with fewer multiplications and reduced data storage requirements compared to a FIR filter. A popular example of an IIR filter is the exponentially weighted moving average (EWMA) or exponential smoothing, which is represented as

$$z_m(n) = \alpha z_m(n - 1) + (1 - \alpha)x(n). \tag{10}$$

In general, the specifications that define filter behavior usually are based on their frequency domain behavior (Mitra and Kaiser, 1993). The frequency response of a system is therefore a useful tool for defining the desired behavior of a linear filter and for decomposing signals into features that might be beneficial for numeric–symbolic mapping. Frequency domain defines the effect of the filter on a sinusoid of a given frequency f_0. For any linear system, when a sinusoid is applied, the output is a sinusoid of the same frequency; only the amplitude and the relative location of the zero crossings change. The factor by which the amplitude changes is called the *filter gain* at f_0, and the shift in the zero locations as a fraction of a full period is the *phase shift* at f_0. Knowledge of the gain and phase at all possible frequencies completely defines the behavior of the filter. A linear system is said to pass a certain frequency if the amplitude of the sinusoid is substantially maintained in passing the signal through the filter. Con-

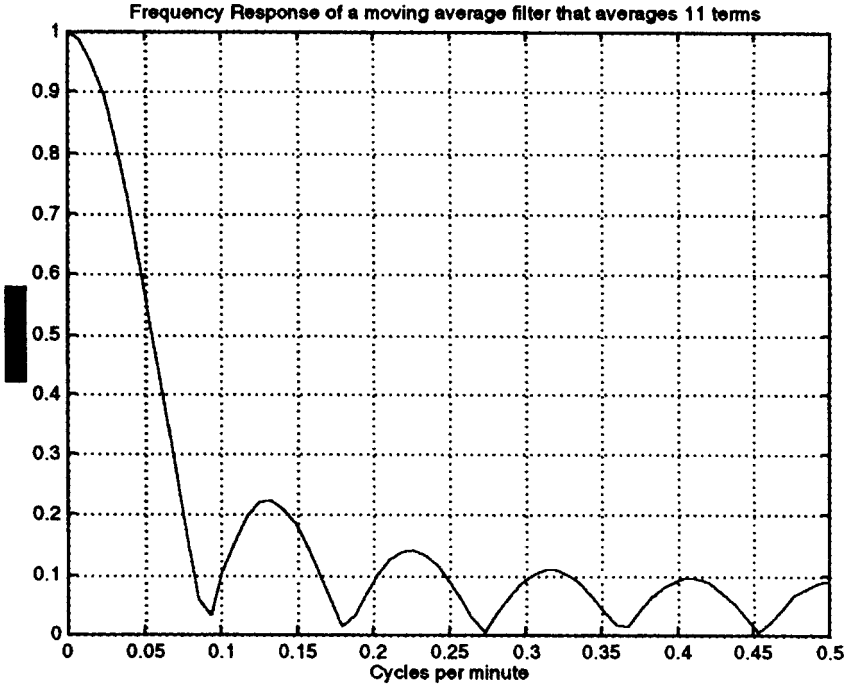


FIG. 8. Frequency response of a moving average filter.

versely, a frequency is rejected if the output amplitude drops below a specified threshold. The desired filter behavior is defined in terms of those frequencies passed and those rejected.

Filter specifications are matched to the response times of the process. For example, a given process has a time constant of 5 minutes. That means that it can respond over the frequency range of 0 to 1/20 of a cycle per minute. Higher frequencies are attenuated naturally by the process. Thus, if the data contain components beyond 0.05 cycles per minute, then those components are likely to be unwanted interferences. The linear filter would pass the frequencies between 0 and 1/20 and reject frequencies outside this range. The filter should attenuate frequencies higher than one decade above the break-point frequency. Process measurements processed by this filter are transformed to a new sequence with less interference than the original data. In this way, an input mapping has been defined.

To illustrate behaviors of different filters, consider a moving average filter that averages over 11 terms. Such a filter has the frequency response shown in Fig. 8. Note that this filter has a relatively low gain of 0.55 at the break-point frequency of 0.05 cycles per minute. So in the range of

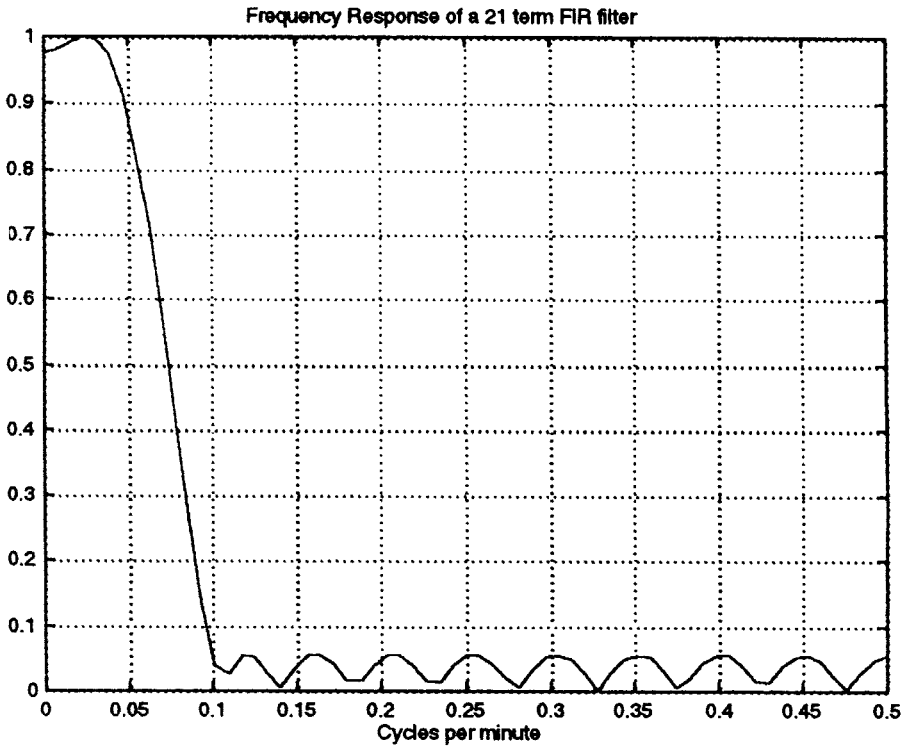


FIG. 9. Example of an FIR filter.

frequencies that it should pass, there is significant attenuation. In the frequency range beyond 0.05 where there is noise, the gain still can be quite large, thereby passing undesirable frequencies. Increasing the number of terms over which averaging occurs reduces the gain in the 0- to 0.05-cycle range and only makes the problem worse.

Compare this result with the FIR filter in Fig. 9, which is specifically designed to pass frequencies between 0 and 0.05 cycles per minute and to reject frequencies beginning at 0.1 cycles. Notice that the gain is much higher over the 0 to 0.05 range, allowing considerably more of the desired signal through the filter. Over the range from 0.1 to 0.5, the gain is much smaller than the moving average filter, removing the unwanted signal. This filter can be improved by increasing the number of impulse response coefficients. In terms of Eq. (8), FIR and IIR filtering is equivalent to expanding the signal on a set of time–frequency localized basis functions whose scale is determined by the window size T.

FIR and IIR filters both are able to meet a given filter specification and, in that sense, they are identical. However, FIR filters can be designed for

linear phase where all input sinusoidals are delayed a fixed number of samples. Only the amplitude changes are determined by the filter gain. Thus, linear phase means that the filter output is the filter version of the input delayed by $(n - 1)/2$ samples, where $n - 1$ is the number of past values of x_i used in the computation. This is important if time correlations of the input measurements are critical. For example, assume there are two noisy signals corresponding to related process measurements. If an FIR filter is used, then the time correlation between the two is maintained because they are both delayed the same number of samples even though they are composed of different frequencies. IIR filters involve fewer multiples and achieve the same reductions of unwanted frequencies. However, they delay different frequency components by differing amounts. If a predictive model is needed, IIR filtering can change the underlying correlation structure of the input data and affect the final model form.

2. *Nonlinear or Multiscale Filtering*

The fixed resolution of single-scale filtering methods is inappropriate for signals containing contributions from events at multiple scales. Multiscale signals are very common in the chemical process industry because the underlying physical and chemical phenomena occur over different temporal and spatial horizons, making them inherently multiscale by nature. For slowly varying or steady portions of a signal, the temporal localization of the filter must be coarse. If this signal also contains any contribution from faster events, the coarse temporal localization will oversmooth the fine-scale events. In contrast, a fine temporal resolution will capture fast events accurately, but will retain too much noise when the changes are slow.

For example, consider a continuous process designed to operate at a specified steady state. The steady-state values may experience variation due to grade changes, throughput, or unmeasured disturbances. In this case, the data can be modeled effectively as a noisy measurement of a piecewise constant signal. The signal is constant until a new set point is defined, after which it takes on a new constant value. If single-scale filters are used, they will blur the points of transition because the high-frequency components of the transition are suppressed. In contrast, multiscale filters can be designed to remove this blurring at transition by adapting the filter resolution to the nature of the measured signal. Multiscale filters represent the next leap in input mapping technology by virtue of their ability to address the inherent multiscale nature of processes. Multiscale filters transform the measured data in either a nonlinear or a time-varying manner.

To illustrate, the FIR Median Hybrid (FMH) as described by Heinonen

and Neuvo (1987) is particularly effective at removing noise from a piecewise constant signal corrupted by noise. Just as with FIR filters, its output depends on a finite number of present and past inputs. However, it produces an output that is the median if the data are rank ordered. The filter operates on $2n + 1$ data points to compute forward and backward predictions of the center or $(n + 1)$st value (as described next). n is the number of points in a selected window of observations. The filter output is then the median of these two predictions and the actual nth value. The main advantage of FMH filters over ordinary median filters is the reduced computational complexity. It involves the determination of the median of three elements rather than a sort of $2n + 1$ elements.

The FMH filter is configured by selecting a window half-width of size $n > 0$ but smaller than half the length of the shortest expected region of constant value in the data. Values of n that are too large result in a window width that is too large, leading to distortion of a signal whose constant region is shorter than n. However, there is a trade-off in that a large n contributes to improving the noise suppression capabilities because more data are used to compute the forward and backward predictions. Ideally, n should be chosen as large as possible without producing a window so large that the time length of constant values is smaller.

As stated, the computation involves a forward prediction and a backward prediction of the $(m + 1)$st data point. At a time instance m under consideration, the value of $x_i(m + 1)$ is the median of $x_i(m+1)$, $\hat{x}_i(m + 1)$, the average of n previous values, and $\tilde{x}_i(m + 1)$, the average of n future values. The backward prediction operates on a sequence, $\{x_i(m - n + 1), \ldots, x_i(n)\}$ to compute an average and uses that as an estimate of the next value. That is,

$$\hat{x}_i(m + 1) = \frac{1}{n} \sum_{k=m-n+1}^{n} x_i(k). \tag{11}$$

The forward prediction $\tilde{x}_i(m + 1)$ is computed by averaging the n sample values in the future of the $(m + 1)$st one. Since future data are required, this computation is not possible in real time and hence is noncausal.

$$\tilde{x}_i(m + 1) = \frac{1}{n} \sum_{k=1}^{n} x_i(m + 1 + k) \tag{12}$$

Mathematically, $(m + 1)$st FMH filter output is given as

$$x_i(m + 1) = \text{med}\{\hat{x}_i(m + 1), \tilde{x}_i(m + 1), x_i(m + 1)\}. \tag{13}$$

The filter can be made iterative by reapplying the same algorithm to the output of the filter. The advantage of doing this is to obtain greater reduction

in process interference (Heinonen and Neuvo, 1987; Gallagher and Wise, 1981).

More recently, the development of wavelets has allowed the development of fast nonlinear or multiscale filtering methods that can adapt to the nature of the measured data. Multiscale methods are an active area of research and have provided a formal mathematical framework for interpreting existing methods. Additional details about wavelet methods can be found in Strang (1989) and Daubechies (1988).

In brief, wavelets are a family of functions of constant shape that are localized in both time and frequency. A family of discrete dyadic wavelets is represented as

$$\Psi_{mk}(t) = \frac{1}{\sqrt{d}}\Psi\left(\frac{t-u}{d}\right), \tag{14}$$

where d is the parameter that deletes or compresses the mother wavelet, ψ, and k is the translation parameter that translates to produce the multiscale character to the filter. Equation (14) can be discretized dyadically to

$$\psi_{dk}(t) = 2^{-d/2}\Psi(2^{-d}t - k), \qquad 0 \le d \le L, 0 < k \le N, \tag{15}$$

where L and N are the maximum number of scales and signal length, respectively. Wavelets decompose the time–frequency domain as shown in Fig. 7d, with the temporal localization being finer at higher frequencies. Figure 10 shows that any square integrable signal can be represented at multiple scales by decomposition on a family of wavelets and scaling functions. The signals in Figs. 10b through 10f represent information from the original signal in Fig. 10a, at increasingly coarse scales or lower frequency bands. Each point in these figures is the projection of the original signal on a wavelet. The last signal in Fig. 10f represents the coarsest scale or lowest frequencies.

Each wavelet is localized in time and frequency and covers a region of the time–frequency space depending on the value of its translation and dilation parameters. For orthonormal wavelets, fast filtering algorithms of wavelets proportional to the number of measurements have been developed for multiscale decomposition of a discrete signal of dyadic length (Mallat, 1989). Given these general properties, techniques for multiscale filtering rely on the observation that contributions from an underlying deterministic signal are concentrated in relatively few of the time–frequency localized basis functions, while the random errors are distributed over a large number of basis functions (Donoho *et al.*, 1995). Thus, the problem of filtering a noisy signal is that of recovering coefficients that correspond to the underlying signal. The magnitudes of the wavelet coefficients corresponding to the

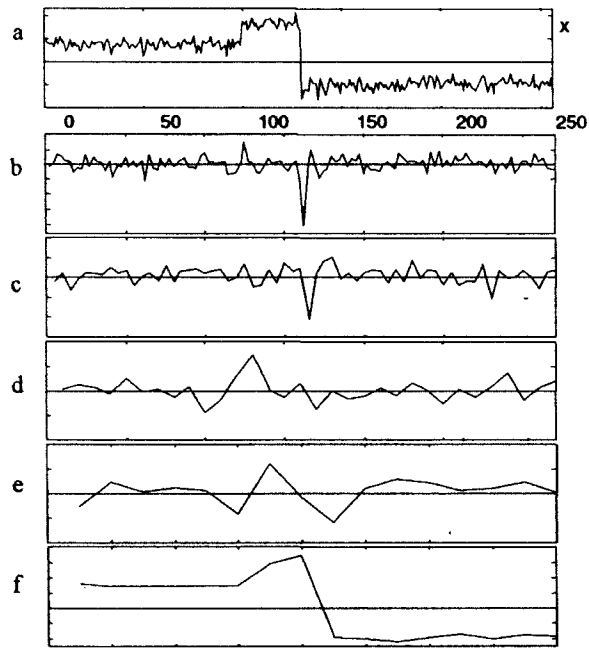


FIG. 10. Wavelet decomposition.

underlying noise-free signal are larger than the magnitude of the coefficients corresponding to the noise. The underlying signal can be recovered by eliminating wavelet coefficients smaller than a threshold. These properties are the basis of the wavelet thresholding approach developed by Donoho *et al.* (1995).

The methodology for multiscale filtering consists of the following three steps:

1. *Decompose* measured data on a selected family of basis functions
2. *Eliminate* coefficients smaller than a selected threshold to reduce contribution of the errors
3. *Reconstruct* the rectified or filtered signal

The multiscale basis functions capture the fast changes in coefficients corresponding to the fine-scale basis functions, while the slower changes are captured by the coarse-scale basis functions. Thus, the wavelet thresholding method adapts its resolution to the nature of the signal features and reduces the contribution of errors with minimum distortion of the features retained in the rectified signal.

The quality of the filtered signal depends on proper selection of the

threshold, which can be determined based on knowledge about the variance of the errors, the stochastic nature of the errors, and the known properties of stochastic processes in the wavelet domain. The variance of the noise at each scale represents the energy of the stochastic process in the corresponding frequency band, which is equivalent to the power spectrum of the noise in the corresponding range of frequencies. If the noise is known to be uncorrelated, then the constant power spectrum at all frequencies indicates that the threshold should be identical at each scale and equal to the standard deviation of the noise in the original signal. If the noise is autocorrelated, then the threshold at each scale should change according to the variation of the power spectrum of the noise. Several methods have been suggested by statisticians for estimating the threshold for filtering from the measured data (Donoho *et al.*, 1995; Nason, 1996). Several of these methods are practical for process data rectification. The *VisuShrink* method (Donoho *et al.*, 1995) determines the threshold at each scale as

$$t_m = \sigma_m \sqrt{2 \log N}, \tag{16}$$

where σ_m is the standard deviation of the errors at scale m and N is the length of the signal. The error between the actual error-free signal and the rectified signal is guaranteed to be within $O(\log N)$ of the error between the error-free signal and the signal rectified with a priori knowledge about the smoothness of the underlying signal. Thus, the rectified signal by wavelet thresholding is nearly optimal for a wide variety of theoretical objectives such as various error norms and smoothness. These and other theoretical properties of wavelet thresholding are discussed in detail by Donoho *et al.* (1995).

The value of the standard deviation of the errors at each scale, σ_m, can be determined from knowledge of the power spectrum of the errors, if available. In most practical situations, information about the stochastic character and variance of the errors is not available, so σ_m must be estimated from the measured data. Because most of the small coefficients in the multiscale decomposition of a measured signal correspond to the errors, and those corresponding to the underlying signal are larger in magnitude, the variance of the errors at each scale can be estimated as a function of the wavelet coefficients at the selected scale. The accuracy of this estimation decreases for a noisy signal at coarser scales because the number of available data points decreases, and the contribution from the underlying signal may increase.

Several extensions of wavelets have been developed to improve their ability to solve practical problems. Wavelet packets (Coifman and Wickerhauser, 1992) are a library of basis functions that cover a wide variety of shapes. The library can be searched efficiently to select the best set of

orthonormal basis functions for representing or filtering a signal. Wavelet packets overcome the fixed shape of wavelets and allow greater flexibility in decomposing the time–frequency space.

The multiscale filtering methods described in the preceding paragraphs can only be applied off-line to data sets of dyadic length because of the noncausal nature of the wavelet filters and the dyadic discretization of the translation and dilation parameters. Work by Nounou and Bakshi (1999) has focused on addressing these disadvantages with multiscale methods for online filtering of data of arbitrary length. This approach filters the data in a moving window of dyadic length.

3. *Multivariate Input Analysis*

Filters are designed to remove unwanted information, but do not address the fact that processes involve few events monitored by many measurements. Many chemical processes are well instrumented and are capable of producing many process measurements. However, there are far fewer independent physical phenomena occurring than there are measured variables. This means that many of the process variables must be highly correlated because they are reflections of a limited number of physical events. Eliminating this redundancy in the measured variables decreases the contribution of noise and reduces the dimensionality of the data. Model robustness and predictive performance also require that the dimensionality of the data be reduced.

Techniques for multivariate input analysis reduce the data dimensionality by projecting the variables on a linear or nonlinear hypersurface and then describe the input data with a smaller number of attributes of the hypersurface. Among the most popular methods based on linear projection is principal component analysis (PCA). Those based on nonlinear projection are nonlinear PCA (NLPCA) and clustering methods.

Principal component analysis encompasses the most important family of linear projection methods. In general, PCA models the correlation structure in a set of data assembled in a matrix X, where the rows are the process measurements at a fixed sampling time, and a column is a uniformly sampled variable. PCA produces a mapping of the data set X onto a reduced subspace defined by the span of a chosen subset of eigenvectors (*loadings*) of the variance–covariance matrix of the X data. The net effect is that a given vector can be represented approximately by another vector with fewer components. The components not included in the approximation capture the errors or noise in the data matrix, thereby providing PCA with the ability to perform multivariate filtering. In terms of the unifying framework described in Section II, PCA transforms the data by taking a

linear combination of the measured variables that captures the variance in the data while being orthogonal to the previously determined transformed or latent variables. One advantage of this technique is that it permits the development of a linear model that produces an orthogonal set of pseudomeasurements (*scores*) that contain the significant variations of the X data. A thorough review of PCA can be found in the article by Wold *et al.* (1987a) and in books by Jolliffe (1986) and Jackson (1992).

PCA involves several steps. First, the data are mean-centered and often normalized by the standard deviation. Mean centering involves subtracting the average value for each variable from the corresponding measurement. Scaling is necessary to avoid problems associated with having some measurements with a larger range of values than others. This larger range may appear to be more meaningful than it really is. Scaling puts all the measurements on the same magnitude, which means multiplying the mean-centered data by an appropriate constant, usually the inverse of the standard deviation. If the variables are noisy, they must be normalized by the covariance matrix of the noise.

The variance–covariance matrix is generated from this normalized data by the relationship X^TX, where X is the normalized data matrix. The X^TX matrix is positive semidefinite and its eigenvectors and their associated eigenvalues determine the variability in the X data. The eigenvectors define the directions in the X space where most of the variability occurs. The first eigenvector defines the direction along which the largest variations are located, and so forth. Overall, the eigenvectors define a subspace within the X space on which each data point is projected. These eigenvectors or directions form a matrix of the *principal component loadings* and are the input transformation parameters, or projection directions. The eigenvalue related to each eigenvector indicates the variance explained by the corresponding eigenvector.

The projection of the X data vectors onto the first eigenvector produces the first latent variable or pseudomeasurement set, Z_1. Of all possible directions, this eigenvector explains the greatest amount of variation in X. The second eigenvector explains the largest amount of variability after removal of the first effect, and so forth. The pseudomeasurements are called the *scores, Z,* and are computed as the inner products of the true measurements with the matrix of loadings, α:

$$Z = X\alpha. \tag{17}$$

Because the eigenvectors are ordered according to the maximum variability, a subset of the first $k < N$ scores define the most variability possible in a k-dimensional subspace. This subspace is referred to as the *score space.*

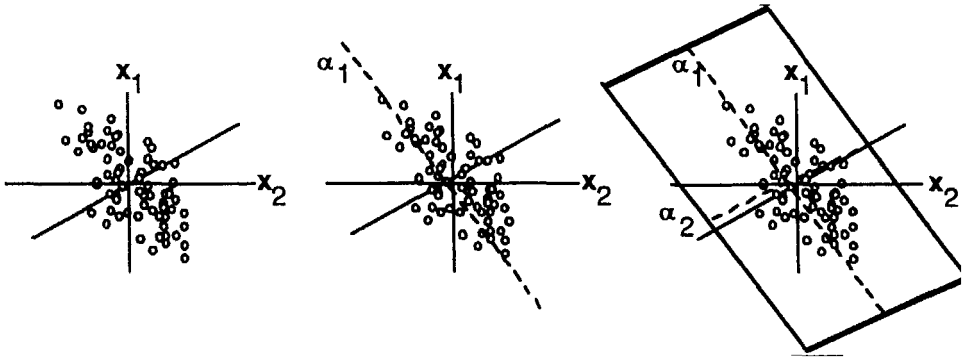


FIG. 11. Simple illustration of PCA. α_1 is the first principal component and α_2 is the second principal component.

The remaining $N - k$ variables capture the errors in the variables. The data matrix may then be represented as

$$X = Z_k \alpha_k^T + E_k, \tag{18}$$

where Z_k, α_k, and E_k are the matrices of the first k scores, loadings, and reconstruction errors, respectively.

PCA is useful not only from a dimension reduction standpoint, but also in that the scores themselves provide additional information important in data analysis (Kresta and MacGregor, 1991; Piovoso and Kosanovich, 1994). A simple example of PCA is illustrated in Fig. 11.

In this example, there are two measurements shown by the left panel. In general, the measurements define an N-dimensional hyperspace, where N is the rank of X^TX. Observe that the data are not scattered randomly in the variable space. Rather, they lie primarily along the dotted line drawn through the data shown in the middle panel. This line, α_1, is defined by the first eigenvector or principal component and represents those linear combinations of the data that capture the maximum variability of X. The projections of the data onto α_1, defined by their distances along it from the origin, constitute the scores. If this approximation is not sufficiently accurate as determined by the size of the residuals, a second eigenvector, α_2, can be found as a function of the residual data as shown in the right panel. The second eigenvector will be orthogonal to the first, in the direction of greatest variability of the residual $(X - z_1\alpha_1^T)$.

The two eigenvectors define a plane in the original variable space. This process can be repeated systematically until the eigenvalue associated with each new eigenvector is of such a small magnitude that it represents the noise associated with the observations more than it does information. In the limit where the number of significant eigenvectors equals the number

of variables, there is no reduction in the dimension of the variable space. If the eigenvalues reached exactly zero, it would suggest that the variables are linearly independent. This does not happen in practice because of noise in the data.

PCA is a data preprocessor in that it provides a mechanism to reduce data dimensionality and filter noise. Because only a few of the possible loadings are kept, data are projected into a lower dimensionality where most of the process information is located. Eliminating less significant loadings also eliminates the portions of the data space where the noise level is likely to be high. In this sense, PCA is a form of input mapping like filtering, but with extended capability. We note that PCA can be extended to handle data in three-dimensional arrays, making it also convenient for batch operations (Kosanovich *et al.,* 1995; Nomikos and MacGregor, 1994; Wold *et al.,* 1987b). The three dimensions arise from batch trajectories that consist of batch runs, variables, and sample times. These data can be organized into an array X of dimension $(I \times J \times N)$, where I is the number of batches, J is the number of variables, and N is the number of time samples over the duration of the batch. This extended application of PCA, called *multidimensional PCA (MPCA),* is equivalent to performing ordinary PCA on a two-dimensional matrix formed by unfolding X so that each of its vertical slices contains the observed variables for all batches at a given time. In this approach, MPCA explains the variation of variables about their mean trajectories.

This decomposition summarizes and compresses the data, with respect to both variables and time, into low-dimensional score spaces representing the major variability over the batches at all points in time. Each loading matrix summarizes the major time variations of the variables about their average trajectories over all the batches. In this way, MPCA uses not just the magnitude of the deviation of each variable from its mean trajectory, but also the correlations among them. To analyze the performance of a set of batch runs, an MPCA analysis can be performed on all the batches and the scores for each batch can be plotted in the space of the principal components. All batches exhibiting similar time histories will have scores that cluster in the same region of the principal component space. Batches that exhibit deviations from normal behavior will have scores falling outside the main cluster.

B. Methods Based on Nonlinear Projection

Methods based on nonlinear projection are distinguished from the linear projection methods that they transform input data by projection on a nonlin-

ear hypersurface. Although this general distinction is straightforward, the infinite number of possible nonlinear surfaces leads to a wide range of transformation behaviors. Broadly speaking, these methods can be further subdivided into nonlocal and local methods depending on the nature of the hypersurface. The hypersurface for nonlocal methods is unbounded along at least one direction, whereas that for local methods defines a closed region such as a hyperellipsoid.

While the basic objective of linear projection methods is to summarize the data by linear approximations, thereby minimizing the orthogonal deviations in the least-squares sense, the aim of nonlinear, nonlocal methods such as nonlinear principal component analysis is to fit a smooth curve through the data such that the data themselves are allowed to dictate the form of the nonlinear dependency. These methods should not be confused with techniques such as spline smoothers and kernel smoothers that produce a curve that minimizes the vertical deviations subject to some assumed smoothness constraint. Nonlinear PCA determines the principal curve that minimizes the shortest distance to the points with the assumption that the curves are self-consistent for a distribution or data set (Hastie and Tibshirani, 1990). A nonlinear principal component may not exist for every distribution and there is no guarantee that a unique set of nonlinear components exists for a given distribution. For example, the principal axes of an ellipsoidal distribution are the principal curves and, for spherically symmetric distributions, any line through the mean vector is a principal curve. The principal curve algorithm is not guaranteed to converge because of its nonlinear nature. Several algorithms have been developed for nonlinear PCA (Kramer, 1991; Hastie and Tibshirani, 1990; Dong and McAvoy, 1996; Tan and Mavrovouniotis, 1995).

Local methods, on the other hand, are characterized by input transformations that are approached using partition methods for cluster seeking. The overall thrust is to analyze input data and identify clusters of the data that have characteristics that are similar based on some criterion. The *objective* is to develop a description of these clusters so that plant behaviors can be compared and/or data can be interpreted.

Because of these characteristics, local methods are naturally used for interpretation. In this section, we present some of the fundamental elements of these methods as input analysis techniques. However, this discussion closely ties with the interpretation discussion in Section V.

The most commonly used family of methods for cluster seeking uses optimization of a squared-error performance criterion in the form

$$j = \sum_{k=1}^{N_c} \sum_{x \in S_k} \|x - t_k\|^2, \tag{19}$$

where N_c is the possible number of data classes or clusters, S_k is the set of data corresponding to the kth class, and t_k is a prototype defined as the sample mean of the set S_k, which provides the cluster description. The performance index J corresponds to the overall sum of the squared error between the input data in all classes and the prototypes represented by the sample means for each class.

This index is employed by both the k-means (MacQueen, 1967) and the isodata algorithms (Ball and Hall, 1965), which partition a set of data into k clusters. With the k-means algorithm, the number of clusters are prespecified, while the isodata algorithm uses various heuristics to identify an unconstrained number of clusters.

In applying Equation (19), the objective is to find a set of cluster centers that minimize the Euclidean distance between each data point and the cluster center,

$$\min \sum_{k=1}^{N_c} \sum_{i=1}^{n} B_{ki} \|x_i - t_k\|^2, \tag{20}$$

where B_{ki} is an $N_c \times n$ matrix of the cluster partition or membership function.

Of the several approaches that draw upon this general description, radial basis function networks (RBFNs) (Leonard and Kramer, 1991) are probably the best-known. RBFNs are similar in architecture to back propagation networks (BPNs) in that they consist of an input layer, a single hidden layer, and an output layer. The hidden layer makes use of Gaussian basis functions that result in inputs projected on a hypersphere instead of a hyperplane. RBFNs therefore generate spherical clusters in the input data space, as illustrated in Fig. 12. These clusters are generally referred to as receptive fields.

The algorithm originally proposed by Moody and Darken (1989) uses k-means clustering to determine the centers of the clusters. The hypersphere around each cluster center is then determined to ensure sufficient overlap between the clusters for a smooth fit by criteria such as the P-nearest neighbor heuristic,

$$\sigma_k = \left[\frac{1}{P} \sum_{i=1}^{P} \|x_i - t_k\|^2 \right]^{1/2}, \tag{21}$$

where x_i are the P-nearest neighbors of the centers, t_k, resulting in hyperspherical receptive fields. Figure 12 illustrates this graphically. Results generated typically include numerous statistical measures such as distance and variance tables that enable the user to interpret the results and modify adjustable parameters.

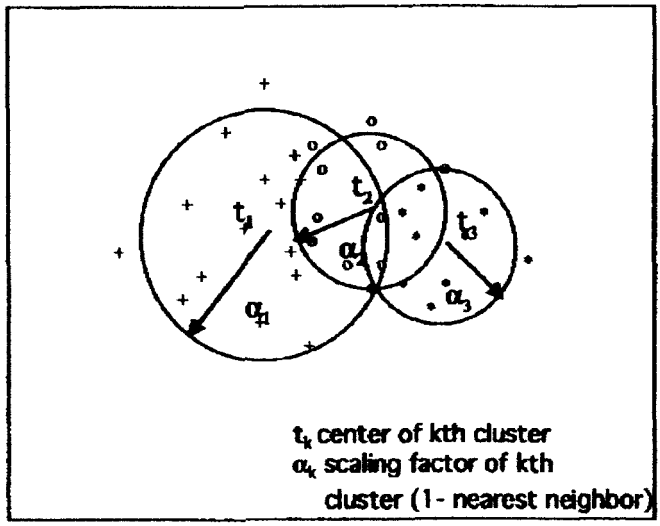


FIG. 12. Receptive fields in RBFNs.

With respect to cluster seeking performance, RBFNs are influenced heavily by the occurrences of input data patterns because clustering effectively is averaged over individual pattern occurrences. It has been shown that this decreases the utility for interpretation (Kavuri and Venkatasubramanian, 1993). Current approaches also require a priori specification of the number of clusters, a drawback from an adaptation standpoint.

Ellipsoidal basis function networks (EBFNs) (Kavuri and Venkatasubramanian, 1993) can be regarded as modified RBFNs with an ellipsoidal basis function and a different learning algorithm. EBFNs also differ from RBFNs by constraining each input data set to one ellipsoid. Figure 13 illustrates EBFNs and compares them with RBFNs in representing the same region of data. The biggest advantage is there are more degrees of freedom to "shape" the clusters so they can better describe the input data. Both algorithms are best employed in an interactive manner because the algorithms are free to identify clusters without consideration of the analysis of interest.

The objective functions for both k-means clustering and the P-nearest neighbor heuristic given by Eqs. (20) and (21) use information only from the inputs. Because of this capacity to cluster data, local methods are particularly useful for data interpretation when the clusters can be assigned labels.

The Adaptive Resonance Theory (ART) family of approaches (described for process applications by Whiteley and Davis, 1994), although

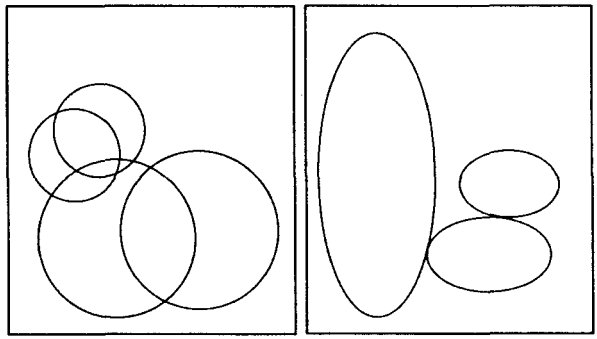


FIG. 13. A comparison of RBFNs and EBFNs.

functionally the same as RBFNs and EBFNs in a number of ways, represents an entirely different philosophy of cluster seeking and therefore embodies a much different algorithm. As shown in Fig. 14, ART forms clusters and regions of clusters in a manner different from EBFNs and RBFNs.

Input data mapping still corresponds to projection on a hypersphere; however, ART uses vector direction to assess similarity rather than using a distance measure as shown in Fig. 15. This translates into the use of hypercone clusters in a unit hypercube.

Unlike RBFNs and EBFNs, the ART approach is aimed at self-organizing properties that address the knowledge management issues associated with incorporating new input data without corrupting the existing information. This avoids the problem described in preceding paragraphs for RBFNs, in which the cluster performance is overly influenced by the number of occurrences of input data patterns. ART is the only clustering approach designed for evolutionary adaptation. This aspect of ART is discussed more fully in Section V, Data Interpretation.

Procedurally, an input data vector is presented to the ART network.

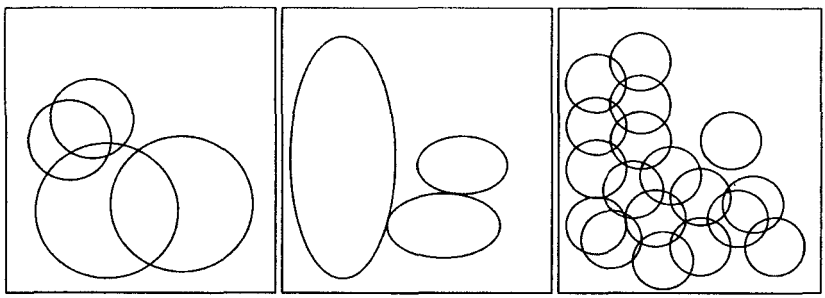


FIG. 14. RBFN, EBFN, and ART approaches.

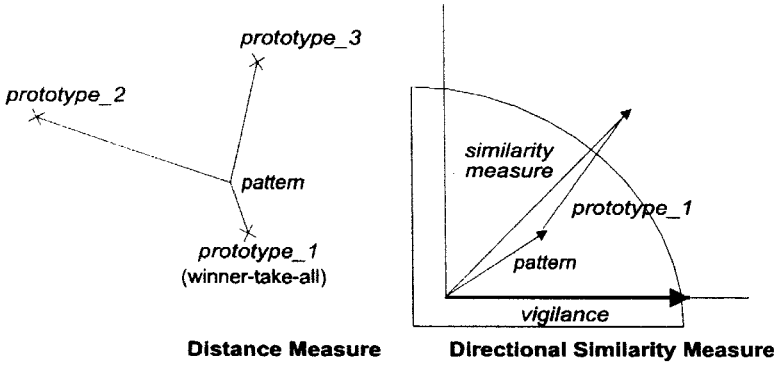


FIG. 15. Distance vs directional similarity measures.

After some specialized processing by the network, the input pattern is compared to each of the existing cluster prototypes in the top layer. The "winner" in the top layer is the prototype most similar to the input. If the similarity between the winner and the input exceeds a predetermined value (the vigilance parameter), learning is enabled and the winner is modified slightly to more closely reflect the input pattern. If the similarity between the winner and input is less than that required by the vigilance parameter, a new unit is initialized in the top layer and learning is enabled to create a prototype similar to the input. This ability to modify existing cluster definitions based on similarity criteria or to generate new cluster definitions provides ART with its adaptive capacity. This subject will be discussed in detail in Section V.

IV. Input–Output Analysis

In sharp contrast to input methods discussed in the last section, input–output analysis is aimed at the construction of models generally for the purpose of predicting output process behaviors given the input data. Although emphasis is on input–output modeling, several analysis methods account for the behavior of the outputs when extracting relevant input features. The analysis may therefore be used in two different ways: (1) the numeric output of the input–output model can be used as a predictive model, or (2) the transformed inputs or latent variables, z_m, which have been influenced by the output, can be used as data interpretation input. Although our focus is on interpretation, the close relationship between empirical modeling and data interpretation is recognized. In this section, we summarize those modeling methods that offer latent variables or trans-

formations that could be used in interpretation. In Section V we focus specifically on the use of both input and input–output analysis for data interpretation.

Following the same generalized model given by Eq. (6), input–output methods may be broadly classified as either projection-based or partition-based methods, as listed in Fig. 2.

A. Methods Based on Linear Projection

Methods based on linear projection transform input data by projection on a linear hyperplane. Even though the projection is linear, these methods may result in either a linear or a nonlinear model depending on the nature of the basis functions. With reference to Eq. (6), the input–output model for this class of methods is represented as

$$y_j = \sum_{m=1}^{M} \beta_{mi}\theta_m \left\{ \sum_{i=1}^{d} \alpha_{mi} x_i \right\}. \tag{22}$$

The reference to *linear* stems from the linearity of the term in brackets even though the overall function can be nonlinear. Among the most popular linear methods for input–output analysis are ordinary least squares regression (OLS), principal component regression (PCR), and partial least squares regression (PLS). Popular nonlinear methods in this class are back propagation neural networks with one hidden layer (BPN), projection pursuit regression (PPR), nonlinear PCR (NLPCR), and nonlinear PLS (NLPLS).

The comparison of these methods based on the type of input transformation, the nature of the basis functions, and the optimization criteria is presented in Table I. These methods are all very similar, to such a degree that it is often difficult to select which would be most appropriate for a given application. In this section, we highlight the key differences that form the basis for choosing relative to the data in a particular application.

1. Ordinary Least Squares

OLS is also called *multiple linear regression* (MLR) and is a commonly used method to obtain a linear input–output model for a given data set. The model obtained by OLS for a single output is given by

$$y = \beta\alpha_1 x_1 + \beta\alpha_2 x_2 + \ldots + \beta\alpha_d x_d. \tag{23}$$

TABLE I
COMPARISON MATRIX FOR INPUT–OUTPUT METHODS

Method	Input transformation	Basis function	Optimization criteria
Ordinary least squares	Linear projection	Fixed shape, linear	α, maximum squared correlation between projected inputs and output β, minimum output prediction error
Principal component regression	Linear projection	Fixed shape, linear	α, maximum variance of projected inputs β, minimum output prediction error
Partial least squares	Linear projection	Fixed shape, linear	α, maximum covariance between projected inputs and output β, minimum output prediction error
Back propagation network (single)	Linear projection	Fixed shape, sigmoid	$[\alpha, \beta]$, minimum output prediction error
Projection pursuit regression	Linear projection	Adaptive shape, supersmoother	$[\alpha, \beta, \theta]$, minimum output prediction error
Back propagation network (multiple)	Nonlinear projection, nonlocal	Fixed shape, sigmoid	$[\alpha, \beta]$, minimum output prediction error
Nonlinear principal component analysis	Nonlinear projection, nonlocal	Adaptive shape	$[\alpha, \phi]$, minimum input prediction error
Radial basis function network	Nonlinear projection, local	Fixed shape, radial	$[\sigma, t]$, minimum distance between inputs and cluster center β, minimum output prediction error
Classification and regression trees	Input partition	Adaptive shape, piecewise constant	$[\beta, t]$, minimum output prediction error
Multivariate adaptive regression splines	Input partition	Adaptive shape, spline	$[\beta, t]$, minimum output prediction error

It can be considered a special case of Eq. (22) with only one linear basis function:

$$y_j = \beta_1 \sum_{i=1}^{d} \alpha_{i1} x_i. \tag{24}$$

For multiple observations, Eq. (24) constitutes a system of linear equations,

$Y = X\beta\alpha$. If the number of input variables is greater than the number of observations, there is an infinite number of exact solutions for the least squares or linear regression coefficients, $\beta\alpha$. If the variables and observations are equal, there is a unique solution for $\beta\alpha$, provided that X has full rank. If the number of variables is less than the number of measurements, which is usually the case with process data, there is no exact solution for $\beta\alpha$ (Geladi and Kowalski, 1986), but α can be estimated by minimizing the least-squares error between the actual and predicted outputs. The solution to the least-squares approximation problem is given by the pseudoinverse as

$$\beta\alpha = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y. \tag{25}$$

If the inputs are correlated, then the inverse of the covariance matrix does not exist and the OLS coefficients cannot be computed. Even with weakly correlated inputs and a low observations-to-inputs ratio, the covariance matrix can be nearly singular, making the OLS solution extremely sensitive to small changes in the measured data. In such cases, OLS is not appropriate for empirical modeling.

2. *Principal Component Regression*

Principal component regression (PCR) is an extension of PCA with the purpose of creating a predictive model of the Y-data using the X or measurement data. For example, if X is composed of temperatures and pressures, Y may be the set of compositions that results from thermodynamic considerations. Piovoso and Kosanovich (1994) used PCR and a priori process knowledge to correlate routine pressure and temperature measurements with laboratory composition measurements to develop a predictive model of the volatile bottoms composition on a vacuum tower.

PCR is based on a PCA input data transformation that by definition is independent of the Y-data set. The approach to defining the X–Y relationship is therefore accomplished in two steps. The first is to perform PCA on the X-data, yielding a set of scores for each measurement vector. That is, if x_k is the kth vector of d measurements at a time k, then z_k is the corresponding kth vector of scores. The score matrix Z is then regressed onto the Y data, generating the predictive model

$$Y = Z\beta + E_y, \tag{26}$$

where β is the matrix of regression coefficients and E_y are the reconstruction errors. Using the orthogonality of the matrix of eigenvectors, α, and the relationship between X and α

$$Z = X\alpha, \tag{27}$$

gives the relationship between X and Y as

$$Y = \alpha\beta X + E_y, \tag{28}$$

where $\alpha\beta$ are the principal component regression coefficients of X onto Y.

3. Partial Least Squares

PLS was originally proposed by Herman Wold (Wold, 1982; Wold *et al.*, 1984) to address situations involving a modest number of observations, highly collinear variables, and data with noise in both the X- and Y-data sets. It is therefore designed to analyze the variations between two data sets, $\{X, Y\}$. Although PLS is similar to PCA in that they both model the X-data variance, the resulting X space model in PLS is a rotated version of the PCA model. The rotation is defined so that the scores of X data maximize the covariance of X to predict the Y-data.

By relation, PLS is similar to PCR. Both decompose the X-data into a smaller set of variables, i.e., the scores. However, they differ in how they relate the scores to the Y-data. In PCR, the scores from the PCA decomposition of the X-data are regressed onto the Y-data. In contrast, PLS decomposes both the Y- and the X-data into individual score and loading matrices. The orthogonal sets of scores for the X- and Y-data, $\{T, U\}$, respectively, are generated in a way that maximizes their covariance. This is an attractive feature, particularly in situations where not all the major sources of variability in X are correlated to the variability in Y. PLS attempts to find a different set of orthogonal scores for the X-data to give better predictions of the Y-data. Thus, orthogonal vectors may yield a *poorer* representation of the X-data, while the scores may yield a *better* prediction of Y than would be possible with PCR.

Mathematically, X is decomposed into a model for the first principal component:

$$X = z_1\alpha_1 + E_{x,1}. \tag{29}$$

The Y-data are similarly decomposed,

$$Y = u_1q_1 + E_{y,1}, \tag{30}$$

where Q is the matrix of eigenvectors such that

$$u = YQ. \tag{31}$$

The score vector, z_1, is found in a two-step operation to guarantee that the covariance of the scores is maximized. Once z_1, α_1, u_1, and q_1 have been found, the procedure is repeated for the residual matrices $E_{x,1}$ and $E_{y,1}$ to find z_2, α_2, u_2, and q_2. This continues until the residuals contain no

useful information. The z's and u's are the scores, and the α's and q's are the loadings of the X- and Y-data, respectively. An excellent tutorial can be found in Geladi and Kowalski (1986).

Trying to preserve information in both the X- and Y-data puts constraints on the corresponding scores and loadings. To have orthogonal score matrices U and Z, two sets of loadings or basis matrices (generally termed Q and α) are needed to represent the X data. The Q matrix is orthogonal, but α is not. If α is forced to be orthogonal, then Z cannot be. Having Z orthogonal implies that the estimate of the inner relationship between the score matrix U for the Y-data and Z for the X-data is more reliable. That is, the variance of the estimates of the regression coefficients α are smaller than would be obtained otherwise.

In developing the relationship between the X- and Y-data, maintaining orthogonality of the scores is important and unlike PCR, is no longer guaranteed. Without orthogonality, simultaneous regression of all the X- and Y-data onto all the scores would have to be done. Not only is that computationally more expensive, but it can lead to a less robust model in that variances in the elements of the regression vector that relates the scores of the X-data to the scores of the Y-data are smaller. Again, having orthogonal Z's minimizes this variance. To generate orthogonality, an algorithm that is more complex than PCR is used. It guarantees that the scores of the X-data are orthogonal; however, to achieve this, orthogonality of the loadings is conceded. Examples of PLS are given by Piovoso *et al.* (1992a, b), where an estimate of the infrequently measured overhead composition in a distillation tower is obtained using many tray temperature measurements.

4. Nonlinear PCR and PLS

Linear PCR can be modified for nonlinear modeling by using nonlinear basis functions θ_m that can be polynomials or the supersmoother (Frank, 1990). The projection directions for both linear and nonlinear PCR are identical, since the choice of basis functions does not affect the projection directions indicated by the bracketed term in Eq. (22). Consequently, the nonlinear PCR algorithm is identical to that for the linear PCR algorithm, except for an additional step used to compute the nonlinear basis functions. Using adaptive-shape basis functions provides the flexibility to find the smoothed function that best captures the structure of the unknown function being approximated.

Similarly, extending linear PLS to nonlinear PLS involves using nonlinear basis functions. A variety of nonlinear basis functions have been used to model the inner relationship indicated in Eq. (22), including quadratic

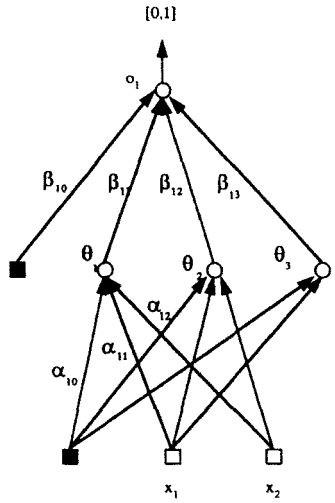


FIG. 16. Representation of BPN as a network of nodes.

functions (Wold *et al.,* 1989), supersmoother (Frank, 1990), BPNs (Qin and McAvoy, 1992), and splines (Wold, 1992). Some nonlinear PLS algorithms prefer to determine the projection directions by linear PLS (Qin and McAvoy, 1992; Frank, 1990), which does not take the nonlinearity of the basis functions into account. This approach is computationally less expensive, but the model is usually less accurate than that obtained by the optimization criterion with the nonlinear basis function (see Table I).

5. *Back Propagation Networks*

Extensive study of BPNs shows they are very similar to nonparametric nonlinear statistical methods such as projection pursuit regression and nonlinear PLS (Ripley, 1994; Hwang *et al.,* 1994; Bakshi and Utojo, 1998). From a statistical point of view, a BPN is a special case of Eq. (22) where the basis functions are of a fixed shape, usually sigmoidal. BPNs are used widely in the chemical industry, and they have already had a significant impact on solving problems in image and speech processing, inexact knowledge processing, natural language processing, sensor data processing, control, forecasting, and optimization (Maren *et al.,* 1990; Miller *et al.,* 1990).

The BPN usually is represented as a network of nodes as shown in Fig. 16. Although the figure illustrates a single output, the framework readily extends to multiple outputs. The input transformation parameters, β, are the weights of the edges connecting the input nodes to the first hidden layer, while the output parameters, α, are the weights on the edges connecting the

hidden layer to the output nodes. The basis functions, θ, usually are sigmoidal, but can be of any other fixed shape. The training methodology for BPNs determines the model parameters for all the nodes simultaneously to minimize the mean squares error as shown in Table I. Different techniques exist for minimizing the error. A technique called *least mean squared error* (LMS), developed by Widrow and Hoff (1960) for the perception, was generalized for a multilayer BPN in Rumelhart *et al.* (1986) and is the most widely used neural network training method. It is based on a gradient descent approach to determine the changes in the weights needed to minimize the squared error. The LMS approach to training has the disadvantage that it can fail to converge or, conversely, converge too slowly if parameters are not set properly. There have been numerous empirical and heuristic approaches to overcome these difficulties, including the addition of a momentum term (Rumelhart *et al.*, 1986). Conjugate gradient techniques as an alternative have also been used (Leonard and Kramer, 1990).

Independent studies (Cybenko, 1988; Hornik *et al.*, 1989) have proven that a three-layered back propagation network will exist that can implement any arbitrarily complex real-valued mapping. The issue is determining the number of nodes in the three-layer network to produce a mapping with a specified accuracy. In practice, the number of nodes in the hidden layer are determined empirically by cross-validation with testing data.

6. *Projection Pursuit Regression*

PPR is a linear projection-based method with nonlinear basis functions and can be described with the same three-layer network representation as a BPN (see Fig. 16). Originally proposed by Friedman and Stuetzle (1981), it is a nonlinear multivariate statistical technique suitable for analyzing high-dimensional data, Again, the general input–output relationship is again given by Eq. (22). In PPR, the basis functions θ_m can adapt their shape to provide the best fit to the available data.

Optimization of the PPR model is based on minimizing the mean-squares error approximation, as in back propagation networks and as shown in Table I. The projection directions α, basis functions θ, and regression coefficients β are optimized, one at a time for each node, while keeping all other parameters constant. New nodes are added to approximate the residual output error. The parameters of previously added nodes are optimized further by backfitting, and the previously fitted parameters are adjusted by cyclically minimizing the overall mean-squares error of the residuals, so that the overall error is further minimized.

The PPR model usually has fewer hidden nodes than a BPN model and is easier and faster to train (Hwang *et al.*, 1994). The PPR basis functions

are computed by smoothing the projected data vs the output by using a variable-span smoother such as the supersmoother (Friedman and Stuetzle, 1981). Determining the basis functions is an important, and often delicate, task, and researchers have suggested several alternatives to the supersmoother, including Hermite function regression (Hwang *et al.,* 1994) and splines (Roosen and Hastie, 1994). The smoothness of the basis functions is usually determined via cross-validation.

B. Methods Based on Nonlinear Projection

This class of methods transforms the inputs in a nonlinear manner. The distinction is readily seen by referring once again to Eq. (22). This family of methods makes use of nonlinear functions both in the bracketed term and in the inner relation. Most of the popular methods project the inputs on a localized hypersurface such as a hypersphere or hyperrectangle.

Input–output analysis methods that project the inputs on a nonlocal hypersurface have also been developed, such as BPNs with multiple hidden layers and regression based on nonlinear principal components.

Among these, the most widely used is the radial basis function network (RBFN). The key distinctions among these methods are summarized in Table I and discussed in detail in Bakshi and Utojo (1998). An RBFN is an example of a method that can be used for both input analysis and input–output analysis. As discussed earlier, the basis functions in RBFNs are of the form $\theta(\|x_i - t_m\|^2)$, where t_m denotes the center of the basis function. One of the most popular basis functions is the Gaussian,

$$\theta_m = \exp\left(-\frac{(x_i - t_m)^2}{\sigma_m^2}\right), \tag{32}$$

where the parameter σ_m determines the extent of localization of the basis function. It should be noted that multidimensional Gaussians can be obtained by multiplying univariate Gaussians.

The most popular modeling method using RBFNs involves separate steps for determining the basis function parameters, σ_m and t_m, and the regression coefficients, β_{mi}. Without considering the behavior of the outputs, the basis function parameters are determined by k-means clustering and the p-nearest neighbor heuristic, as described earlier in the input analysis methods section. The regression parameters are then determined to minimize the output mean-squares error. The optimization criterion for determining the input transformation is given in Table I and considers the input space only. As Table I shows, training RBFNs by this method is analogous to PCR and PLS. Various approaches have been suggested for

incorporating information about the output error in determining the basis function parameters (Chen *et al.*, 1991). Hierarchical methods also have been developed for modeling in a stepwise or node-by-node manner using Gaussian basis functions (Moody, 1989) and wavelets (Bakshi and Stephanopoulos, 1993).

Variations on RBFNs have been developed that project the inputs on hyperellipses instead of hyperspheres. These ellipsoidal basis function networks allow nonunity and unequal input weights, except zero and negative values, causing elongation and contraction of the spherical receptive fields into ellipsoidal receptive fields (Kavuri and Venkatasubramanian, 1993).

C. Partition-Based Methods

Partition-based modeling methods are also called *subset selection methods* because they select a smaller subset of the most relevant inputs. The resulting model is often physically interpretable because the model is developed by explicitly selecting the input variable that is most relevant to approximating the output. This approach works best when the variables are independent (De Veaux *et al.*, 1993). The variables selected by these methods can be used as the analyzed inputs for the interpretation step.

The general empirical model given by Eq. (22) can be specialized to that determined by partition-based methods as

$$Y = \sum_{m=1}^{M} \beta_m \theta_m(XER_m), \tag{33}$$

where R_m is the set of the selected inputs. The basis functions θ_m can be a fixed or adaptive shape. As in other input–output analysis methods, the regressions are usually determined to minimize the output prediction error.

1. Classification and Regression Trees

The basis functions in a CART or *inductive decision tree* model are given by

$$\theta_m(X) = \prod_{p=1}^{P_m} H[s_{pm}(x_{v(p,m)} - t_{pm})], \tag{34}$$

where H is the Heaviside or step function,

$$H[\eta] \begin{cases} 1 \text{ if } \eta \geq 0 \\ 0 \text{ otherwise} \end{cases}. \tag{35}$$

P_m is the number of partitions or splits; $s_{pm} = \pm 1$ and indicates the right or left of the associated step function; $v(p, m)$ indicates the selected input variables in each partition; and t_{pm} represents the location of the split in the corresponding input space. The indices p and m are used for the split and node or basis function, respectively. The basis functions given by Eq. (34) are of a fixed, piecewise constant shape.

The CART method selects the variable for partitioning the input space as the one that minimizes the output mean-square error of the approximation. Recursive partitioning of the input space is continued until a large number of subregions or basis functions are generated. After a region splits, it is removed from the model. Overfitting is avoided by penalizing the output prediction error for the addition of basis functions and by eliminating unnecessary splits by a backward elimination procedure (Breiman *et al.*, 1984). The objective function used for determining all model parameters focuses entirely on minimizing the output error of approximation. Neither distribution of data in the input space nor relationships among the input variables are exploited. The CART model can be represented as a binary tree and is physically interpretable. However, the discontinuous approximation at the partitions prohibits its application to continuous input–output models. CART is often unable to identify interactive effects of multiple inputs.

2. *Multivariate Adaptive Regression Splines*

MARS overcomes the disadvantages of CART for multivariate regression (Friedman, 1991) by dividing the input space into overlapping partitions and keeping both the parent and daughter nodes after splitting a region. This prevents discontinuities in the approximation at the partition boundaries and produces continuous approximations with continuous derivatives. As in CART, the MARS model is represented by Eq. (34), but instead of using the fixed-shape and piecewise constant Heaviside or step functions as the basis functions, MARS uses multivariate spline basis functions obtained by multiplying univariate splines represented by a two-sided truncated power basis,

$$\theta_m(x) = \prod_{p=1}^{P_m} H[s_{pm}(x_{v(p,m)} - t_{pm})] + q, \tag{36}$$

where q is the order of the splines. Comparison of Eqs. (34) and (36) shows that a value of $q = 1$ results in the CART basis functions. For $q > 0$, the approximation is continuous and has $q - 1$ derivatives.

Empirical comparison of MARS with BPN has shown that the perfor-

mance of MARS on problems with correlated inputs is often inferior to that of BPN (DeVeaux *et al.,* 1993). This can be explained by the fact that the ability to approximate correlated inputs efficiently depends on the ability of the projection directions to assume any value. MARS restricts projection directions to assume 0 or 1 values to improve the physical interpretability of the model.

V. Data Interpretation

As discussed and illustrated in the introduction, data analysis can be conveniently viewed in terms of two categories of numeric–numeric manipulation, *input* and *input–output,* both of which transform numeric data into more valuable forms of numeric data. Input manipulations map from input data without knowledge of the output variables, generally to transform the input data to a more convenient representation that has unnecessary information removed while retaining the essential information. As presented in Section IV, input–output manipulations relate input variables to numeric output variables for the purpose of predictive modeling and may include an implicit or explicit input transformation step for reducing input dimensionality. When applied to data interpretation, the primary emphasis of input and input–output manipulation is on feature extraction, driving extracted features from the process data toward useful numeric information on plant behaviors.

Figure 4 in the introduction is a defining view showing *data interpretation* as an extension of data analysis where labels are assigned to extracted features by establishing decision criteria or discriminants based on labeled or annotated data (known and observed) that exhibit sufficiently similar characteristics. Input and input–output manipulations therefore provide the critical feature extraction capability. This publication refers to input and input–output mapping extended to label assignment as *numeric–symbolic mapping; feature mapping* is used to describe the extension of feature extraction to label assignment.

Numeric–symbolic approaches are particularly important in process applications because the time series of data is by far the dominant form of input data, and they are the methods of choice if annotated data exist to develop the interpretation system. With complete dependence on the annotated data to develop the feature mapping step, numeric–symbolic mappers can be used to assign labels directly. However, as the amount and coverage of available annotated data diminishes for the given label of interest, there is a need to integrate numeric–symbolic approaches with

knowledge-based system (KBS) approaches where the numeric–symbolic approaches are used to produce intermediate labels as input to the KBS approaches.

Figure 3 illustrates a comprehensive data interpretation system that maps directly from an input time series of data, X, to the desired vector of labels, Ω. After data are preprocessed to produce a more valuable form of the input data, the numeric–symbolic interpreter is shown to consist only of a feature extractor and feature mapper. This implies the availability of a sufficient amount of annotated data, i.e., labeled training exemplars, $\Omega \rightarrow X$, from existing operating experiences, such that the labels can be assigned with certainty to arbitrary data from the current operation of the plant. This implies that if there is sufficient operating history to provide the labeled data, then it is desirable to rely on this information for data interpretation because it reflects the actual operation. With sufficient operating history, the burden of label assignment is on feature extraction.

In practice, there may not be sufficient operating experience and resultant data to develop a numeric–symbolic interpreter that can map with certainty to the labels of interest, Ω. Under these circumstances, if sufficient knowledge of process behaviors exists, it is possible to construct a KBS in place of available operating data. But the KBS maps symbolic forms of input data into the symbolic labels of interest and is therefore not sufficient in itself. A KBS depends on intermediate interpretations, Ω', that can be generated with certainty from a numeric–symbolic mapper. This is shown in Fig. 4. In these cases, the burden of interpretation becomes distributed between the numeric–symbolic and symbolic–symbolic interpreters. Figure 4 retains the value of input mapping to preprocess data for the numeric–symbolic interpreter.

General considerations of data availability lead immediately to the recognition that detection systems are more likely to be designed as comprehensive numeric–symbolic interpreters as illustrated in Fig. 3. State description systems may be configured as shown in either Fig. 3 or Fig. 4. Fault classification systems are most likely to require the symbolic–symbolic mapping to compensate for limited data as shown in Fig. 4. Many practical data interpretation problems involve all three kinds of interpreters. In all situations, there is a clear need for interpretation systems to adapt to and evolve with changing process conditions and ever-increasing experience.

The front-end and back-end boundaries on each analysis and interpretation component can be defined so that a particular approach or methodology can be determined based on the specific mapping requirements. Practical data interpretation applications often involve the integration of multiple technologies as required by these three distinct forms of data mapping. This

section focuses on several families of numeric–symbolic pattern-recognition methods. The input and input–output mapping approaches discussed previously apply to data interpretation and data analysis in the same way.

Numeric–symbolic pattern recognition employs deterministic and statistical methods when patterns can be represented as vectors or probability density functions (PDFs). These methods are also called *geometric* approaches because they identify regions in a mathematical or statistical representation space that are used to distinguish classes of data (Duda and Hart, 1973; Tou and Gonzales, 1974). Numeric–symbolic approaches often can be characterized by closely linked feature extraction and feature mapping steps. Unlike KBS approaches, geometric approaches are characterized by a heavy emphasis on feature extraction followed by a critical definition for a discriminant or test to assign the labels. Often the feature extraction and mapping steps are embedded and not easily separated.

The number of numeric–symbolic approaches is too vast to address individually in this section. Discussion requires categorization and analysis by key characteristics. As a result, a decomposition by categories brings out the practical advantages and limitations and provides a useful map for selecting approaches for particular problems. Note that, in general, no one family of methods stands out uniformly as the best. Rather, different approaches offer different capabilities that can be used to advantage when addressing particular problem characteristics. Problem characteristics change dramatically from process to process and from label to label. It is therefore critical to approach a complex problem expecting to configure, integrate, and distribute multiple methods and technologies to construct a particular interpretation system.

Data interpretation methods can be categorized in terms of whether the input space is separated into different classes by local or nonlocal boundaries. Nonlocal methods include those based on linear and nonlinear projection, such as PLS and BPN. The class boundary determined by these methods is unbounded in at least one direction. Local methods include probabilistic methods based on the probability distribution of the data and various clustering methods when the distribution is not known a priori.

As introduced earlier, inputs can be transformed to reduce their dimensionality and extract more meaningful features by a variety of methods. These methods perform a numeric–numeric transformation of the measured input variables. Interpretation of the transformed inputs requires determination of their mapping to the symbolic outputs. The inputs can be transformed with or without taking the behavior of the outputs into account by univariate and multivariate methods. The transformed features or latent variables extracted by input or input–output analysis methods are given by Eq. (5) and can be used as input to the interpretation step.

Alternatively, some input–output methods may be used to analyze and interpret the data by directly determining a model between the numeric inputs and symbolic class labels.

The features extracted by univariate and multivariate methods based on linear projection are global, while those extracted by methods based on localized nonlinear projection are restricted to a closed region in the input space, as shown in Fig. 6. With a focus on data interpretation or label assignment, the extracted features are classified based on various distance measures such as simple limit checking and Mahalanobis or Euclidean distances (discussed in detail later). Local methods, including pattern clustering approaches, are distinguished by their direct emphasis on feature extraction for nearest neighbor similarity or pattern feature similarity, a distinction that brings out spatial similarities rather than, direct feature similarity. When extended for interpretation, all data analysis methods require explicit or implicit methods for establishing decision surfaces for assigning labels to data patterns with similar characteristics in the context of the extracted features: that is, interpreting the θ_m and the corresponding weights β_{mi} and parameters α_i. Methods based on nonlocal projection can exhibit desirable properties for useful extrapolation, but also can lead to dangerous misclassifications without any warning about the unreliable nature of the predicted outputs. It is possible to establish decision surfaces that exist as hyperplanes and that are infinite in one or more dimensions. Problems with extrapolation are present when these methods are used to directly map the inputs to the output classes or labels. The potential for extrapolation error necessarily must be managed by estimating the reliability of the predicted values. This is particularly important for process applications where data associated with specific operational classes can be limited or process units can transition into novel operating states.

Alternatively, methods based on nonlocal projection may be used for extracting meaningful latent variables and applying various statistical tests to identify *kernels* in the latent variable space. Figure 17 shows how projections of data on two hyperplanes can be used as features for interpretations based on kernel-based or local methods. Local methods do not permit arbitrary extrapolation owing to the localized nature of their activation functions.

In general, a given numeric–symbolic interpreter will be used in the context of a locally constrained set of labels defining and limiting both the input variables and possible output interpretations. In this context, feature extraction is intended to produce the features that are resolved into the labels. The numeric–symbolic problem boundary is defined backward from the labels of interest so that feature extraction is associated only with the input requirements for a given approach to produce the relevant features needed to generate the label. The distinctions of *relevant features* and

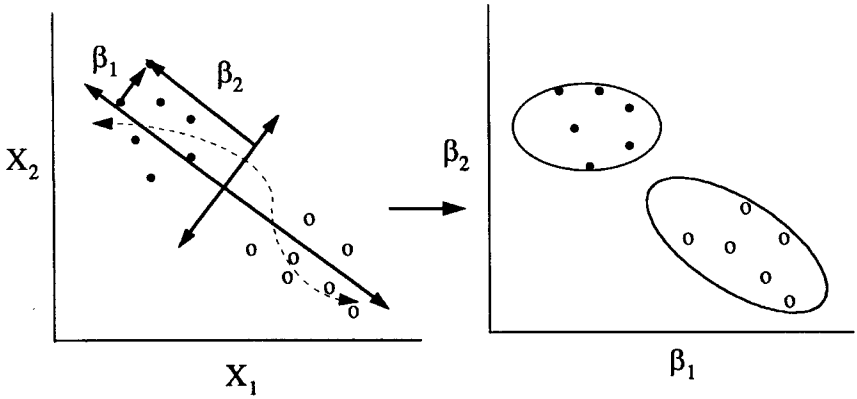


FIG. 17. Using features from projection-based methods for local interpretations.

required input data are made because a separate data preprocessing component may be necessary to condition the input so that interpretation performance is more effective, i.e., filtering. This preprocessing is a more generalized application of feature extraction.

Work on dimension reduction methods for both input and input–output modeling and for interpretation has produced considerable practical interest, development, and application, so that this family of nonlocal methods is becoming a mainstream set of technologies. This section focuses on dimension reduction as a family of interpretation methods by relating to the descriptions in the input and input–output sections and then showing how these methods are extended to interpretation.

A. NONLOCAL INTERPRETATION METHODS

Among nonlocal methods, those based on linear projection are the most widely used for data interpretation. Owing to their limited modeling ability, linear univariate and multivariate methods are used mainly to extract the most relevant features and reduce data dimensionality. Nonlinear methods often are used to directly map the numerical inputs to the symbolic outputs, but require careful attention to avoid arbitrary extrapolation because of their global nature.

1. Univariate Methods

Univariate methods are among the simplest and most commonly used methods that compose a broad family of statistical approaches. Based on

anecdotal observation of their widespread use, these methods represent the state of the art in practical data interpretation.

The most commonly implemented method of limit checking is the *absolute value check* as referred to by Iserman (1984),

$$\phi(x_i)_{\min} < \phi(x_i(t)) < \phi(x_i)_{\max}, \tag{37}$$

where $\phi(x_i(t))$ is the value of the extracted feature at any time t, and $\phi(x_i)_{\min}$ and $\phi(x_i)_{\max}$ are the lower and upper mapping limits, respectively. As shown in Fig. 18, limit checking can be obtained from Eq. (5) by specializing it to the form $z = \phi(x_i)$. Interpretation is achieved when ω_1 is the classification if $\phi(x_i)_{\min} < z_m < \phi(x_i)_{\max}$ and ω_2 otherwise. Note that each input x_i is considered to be a feature. No matter what the value of x_i, the class z is determined by relation to the specified range on x_i. This form of limit checking is easy to implement and places the interpretation emphasis on the specification of the limits.

In the extended family of limit-checking approaches, there is a variety of definitions for $\phi(x_i)$. All definitions are designed to render a time series of data points into a feature that can be appropriately labeled with static limits. These include an individual point, statistically averaged data points, integral of the absolute value of the error (IAE), moving average filtering, cumulative sum (CUSUM), Shewart Charts, and exponentially weighted moving average (EWMA) models. A survey of the use of these methods for statistical process control (SPC) is given by MacGregor and Kourti (1995) and for fault detection by Basseville and Benveniste (1986). The interpretation capability of these approaches is concentrated in establishing appropriate values for $\phi(x_i)_{\min}$ and $\phi(x_i)_{\max}$ that account for the fluctuations in the operation relative to the class labels of interest.

Although usable in many situations, limit-checking methods have limited applicability. First, these methods are univariate and depend on reducing the time series of data points into a single feature. In cases where the

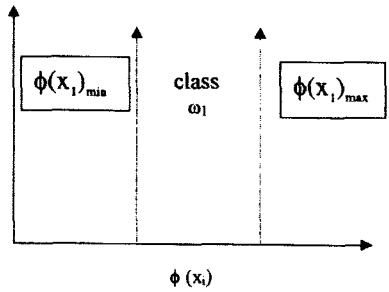


FIG. 18. Univariate limit checking.

process operation is not in a pseudo-steady-state, this has the effect of eliminating much of the useful information contained in the data. These approaches work very well during periods of pseudo-steady-state behavior where various averaging methods represent the true behavior exhibited by the data. However, without the ability to make use of information during transient periods, it is difficult to interpret information during critical periods such as startup, shutdown, and process transitions. Secondly, these methods work best for data interpretations that require static limits. However, many interpretations require changing limits. For example, "normality" interpretation requires definition of a fixed range of normality. When process conditions change and therefore the normality range changes, either the limits must be set at a maximum width to accommodate an entire range of conditions, but with a corresponding degradation in interpretation capability, or the limits must change dynamically. The latter case has been implemented with reasonable success in the form of context-dependent interpretations. This method requires considerable definition of the process operation.

2. *Multivariate Linear Discriminant Methods*

Multivariate linear discriminant methods use the structure of patterns in the representation space to perform pattern recognition. Patterns in a common class are assumed to exhibit similarities with the analytic result that different pattern classes occupy different regions in the representation space. Such a pattern structure enables the representation space to be partitioned into different pattern classes using linear decision surfaces (hyperplanes). There is therefore an important up-front assumption that the pattern classes must be linearly separable. For an arbitrary pattern, a decision on class membership is made by comparing the projection of the pattern on the hyperplane with some threshold value. The global nature of this family of approaches is associated with the infinite length of the decision surfaces, regardless of the data patterns that were used to establish their positions. Figure 19 illustrates a hyperplane (line) in a two-dimensional space.

Because a hyperplane corresponds to a boundary between pattern classes, such a discriminant function naturally forms a decision rule. The global nature of this approach is apparent in Fig. 19. An infinitely long decision line is drawn based on the given data. Regardless of how closely or distantly related an arbitrary pattern is to the data used to generate the discriminant, the pattern will be classified as either ω_1 or ω_2. When the arbitrary pattern is far removed from the data used to generate the discriminant, the approach is extremely prone to extrapolation errors.

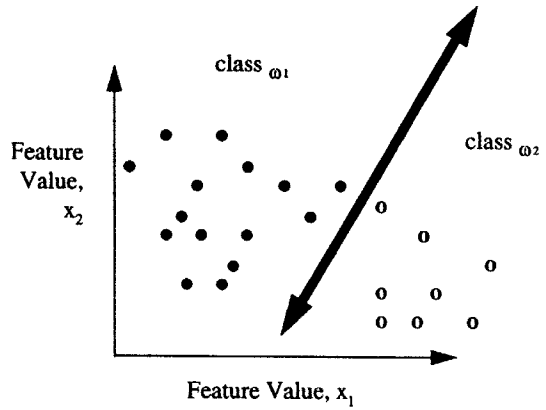


FIG. 19. Partitioning of representation space into regions of discrete pattern class using linear discriminants.

Figure 20 shows more definitively how the location and orientation of a hyperplane is determined by the projection directions, α_i, and the bias, α_0. Given a pattern vector $\mathbf{x}$, its projection on the linear discriminant is in the α direction and the distance is calculated as $|d(\mathbf{x})|/\|\alpha\|$. The problem is the determination of the weight parameters for the hyperplane(s) that separate different pattern classes. These parameters are typically learned using labeled exemplar patterns for each of the pattern classes.

Once determined, these parameters can be used to create linear discriminant functions of the form

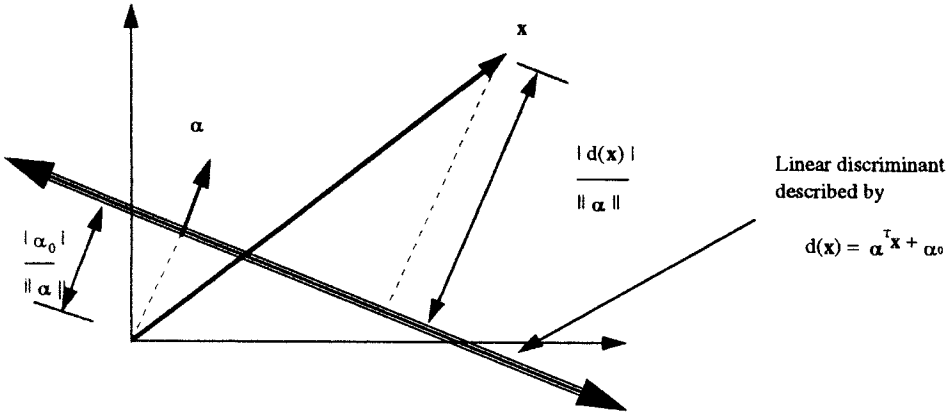


FIG. 20. Orientation and location of linear decision surface.

$$z_m = \sum_{i=1}^{n} \alpha_j x_j + \alpha_0. \tag{38}$$

This function provides a convenient means of determining the location of an arbitrary pattern $\mathbf{x}$ in the representation space. As shown, patterns above the hyperplane result in $z(\mathbf{x}) > 0$, while patterns below generate values of $z(\mathbf{x}) < 0$. The simplest form of the decision rule then is

$$\begin{aligned} &\omega_1 \text{ if } z_m(\mathbf{x}) > 0 \\ &\text{else} \\ &\omega_2 \text{ if } z_m(\mathbf{x}) < 0 \end{aligned} \tag{39}$$

With this approach, all that matters is on which "side" of the hyperplane the pattern lies. Because the relative position between the pattern and a discriminant is arbitrary in that the discriminant is determined by available data and is scaled arbitrarily, it is easy to misclassify any pattern outside the range of training data. Successful application of this method is dependent on (1) linear separability of the pattern classes and (2) determination of the proper weight parameters to describe each hyperplane. Assuming the pattern classes are linearly separable, several iterative procedures have been developed to adaptively learn the hyperplane parameters from representative pattern class exemplars, including the perceptron algorithm (Rosenblatt, 1962), the Widrow–Hoff algorithm (1960), and the Ho–Kashyap algorithm (Simon, 1986).

By definition, the exemplar patterns used by these algorithms must be representative of the various pattern classes. Performance is tied directly to the choice and distribution of these exemplar patterns. In light of the high dimensionality of the process data interpretation problem, these approaches leave in question how reasonable it is to accurately partition a space such as R^{6+} (six-dimensional representation space) using a finite set of pattern exemplars. This degradation of interpretation performance as the number of possible labels (classes) increases is an issue of output dimensionality.

Furthermore, the pattern structures in a representation space formed from raw input data are not necessarily linearly separable. A central issue, then, is feature extraction to transform the representation of observable features into some new representation in which the pattern classes are linearly separable. Since many practical problems are not linearly separable (Minsky and Papert, 1969), use of linear discriminant methods is especially dependent on feature extraction.

With regard to linear projection based methods, the latent variables or scores determined by linear multivariate statistical methods such as

PCA and PLS can be used to interpret data by defining distance metrics for classification. The scores are the extracted features, and significant changes in them with respect to a set of reference scores enables assignment of labels. Reference scores are developed from a careful selection of process data, i.e., normal operating conditions. From a training perspective, these methods require that the relationship between variables be sufficiently linear and/or the calibration sample span a narrow enough portion of space for a linear approximation. The methods are therefore sensitive to variations in the data and outliers and work best for steady-state behaviors. It is particularly important to remove variables that have global effects because they appear in the first principal component can mask other information about the state of the process (Kosanovich *et al.*, 1995).

The most serious problem with input analysis methods such as PCA that are designed for dimension reduction is the fact that they focus only on pattern representation rather than on discrimination. Good generalization from a pattern recognition standpoint requires the ability to identify characteristics that both define and discriminate between pattern classes. Methods that do one or the other are insufficient. Consequently, methods such as PLS that simultaneously attempt to reduce the input and output dimensionality while finding the best input–output model may perform better than methods such as PCA that ignore the input–output relationship, or OLS that does not emphasize input dimensionality reduction.

The global nature of these nonlocal modeling methods manifests itself in the determination of the principal components, which are vectors of infinite length. A new data vector that is far removed from the reference data still will be scored against the principal component eigenvectors. As a result, it is important to manage extrapolation that could lead to incorrect classifications. For example, it is possible that, for a reasonable value of the proximity measure, the sampled variance could be very large, implying significant unexplained variability. By calculating the sampled residual variance, it is possible to estimate the probability that the sample residuals are similar to the data from which the reference score set was developed. This gives a direct indication that the data vector is an outlier or is a member of the reference, or that the process has been modified such that the scores are outside the range defined by the calibration set.

Nonlinear methods based on linear projection also can be used for data interpretation. Since these methods require numeric inputs and outputs, the symbolic class label can be converted into a numeric value for their training. Proposed applications involving numeric to symbolic transformations have a reasonably long history (e.g., Hoskins and Himmel-

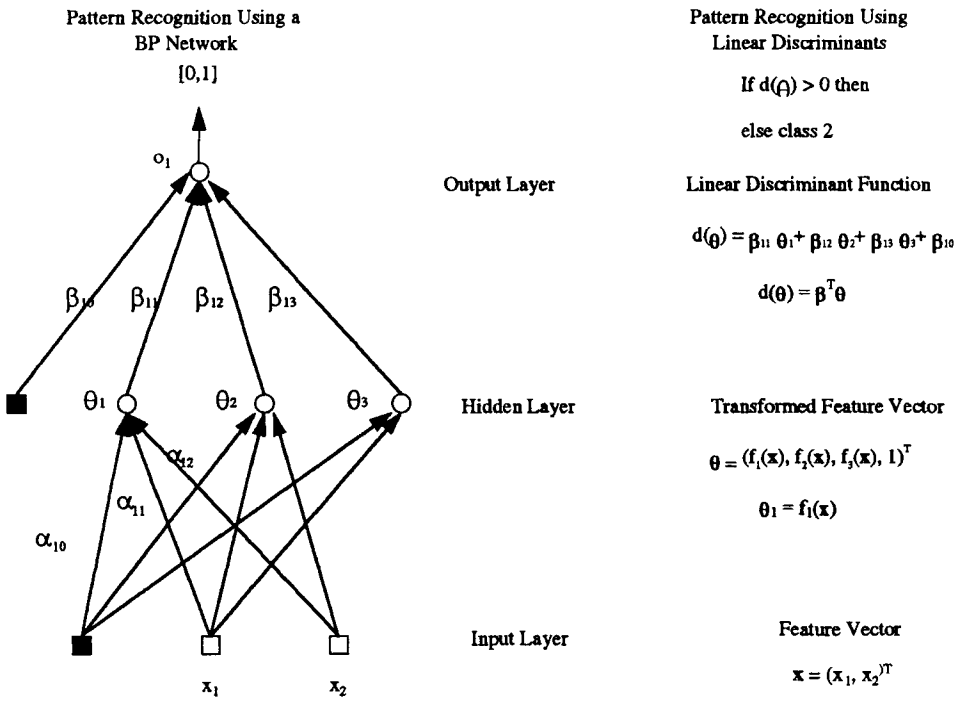


FIG. 21. Correspondence between BP networks and traditional pattern recognition using linear discriminants.

blau, 1988; Hoskins *et al.*, 1991; Naidu *et al.*, 1989; Ungar *et al.*, 1990; Whiteley and Davis, 1992a and b). Figure 21 illustrates the correspondence between the neural network representation of methods based on linear projection and pattern recognition using linear discriminants. Each step of the linear discriminant approach to pattern recognition has a mathematical equivalent in its neural network representation. Both methods utilize a numeric feature vector **x** as input. Patterns are represented by two observable features so that the network contains two input units plus a bias unit for the hidden layer. The presence of a hidden layer indicates that the problem is linearly nonseparable based solely on the observable features. Consequently, it is necessary to perform feature extraction with the objective of identifying higher order features that yield a distribution of patterns in the transformed feature space that are linearly separable. With respect to Fig. 21, three higher order features are required. These scalar-valued features are generated by transformation functions $\theta = f(\mathbf{x})$ in the traditional pattern recognition approach. In the corresponding network, each higher order feature is represented by a single hidden

unit. Consequently, the hidden layer contains three processing units and one bias unit.

The feature transformations $\theta_m(\mathbf{x})$ can take any form depending on the selected method. In Fig. 21, the higher order features are generated by applying the sigmoid function to a linear combination of the observable pattern features. The weights, α_{ij}, associated with the input layer of a BP network define the nonlinear feature extraction performed by the network. The next step is identification of a set of parameters that defines a hyperplane or decision surface that separates the pattern classes in the transformed feature space R^3. These parameters are represented by the vector β in traditional pattern recognition on the right side of Fig. 21. In the BP network, these parameters correspond to the connection weights, β_{mi}, between the units in the hidden and output layers. Pattern classification using the traditional approach involves application of a discriminant function that is similar to a hard limit. The BP network approximates this decision function using a sigmoid. Consequently, the sigmoid function plays two different roles in BP networks with respect to pattern recognition: one associated with transforming the feature space into linearly separable features, and another associated with formation of the linear discriminant as a decision surface.

The BP network is ideally suited to implementation of the linear discriminant approach to pattern recognition. However, the great potential for using BP networks does not reside in the architecture, but rather in the generalized delta rule. The most difficult problem associated with the linear discriminant method is feature extraction, and this is where BP networks make their most significant contribution. By using the generalized delta rule to learn the weights between the input and hidden layers, BP networks effectively automate the feature extraction process. The generalized delta rule also is used to learn the weights associated with the linear discriminant. However, this represents less of a contribution, because there are numerous procedures available when pattern classes are linearly separable.

Given the power of BP networks as methods for automating feature extraction and for combining feature extraction with the linear discriminant, we note that BP networks are ultimately in the family of linear discriminant methods. Therefore, they suffer from the same problems with extrapolation errors associated with infinitely long decision surfaces, as described previously. In general, feedforward networks offer effective feature extraction and interpretation capabilities. Their primary drawback is that a linear discriminant decision surface grossly overstates the size of the true class regions as they are described by the existing data. Data patterns outside the data distributions used to define the decision surfaces are subject to

misclassification. As with projection methods, it is critical to employ safeguards to manage the potential for extrapolation errors. Great care must be taken to use BP networks to interpret only those data that are within the coverage of the training exemplars.

B. Local Interpretation Methods

Local interpretation methods encompass a wide variety of approaches that resolve decisions about input data relative to annotated data or known features that cluster. By characterizing the cluster or grouping, it is possible to use various measures to determine whether an arbitary pattern of data can be assigned the same label as the annotated grouping. All approaches are statistical, but they vary in terms of measures, which include statistical distance, variance, probability of occurrence, and pattern similarity.

1. Statistical Measures for Interpretation

If the probability distribution of the data is or assumed Gaussian, several statistical measures are available for interpreting the data. These measures can be used to interpret the latent variables determined by a selected data analysis method. Those described here are a combination of statistical measures and graphical analysis. Taken together they provide an assessment of the statistical significance of the analysis.

The Q-statistic or square of predicted errors (SPE) is the sum of squares of the errors between the data and the estimates, a direct calculation of variability:

$$Q_i = \sum (x_{ik} - \hat{x}_{ik})^2. \tag{40}$$

Here $\hat{x}_{ik}$ is an estimated value of a variable at a given point in time. Given that the estimate is calculated based on a model of variability, i.e., PCA, then Q_i can reflect error relative to principal components for known data. A given pattern of data, x, can be classified based on a threshold value of Q_i determined from analyzing the variability of the known data patterns. In this way, the Q-statistic will detect changes that violate the model used to estimate $\hat{x}$. The Q-statistic threshold for methods based on linear projection such as PCA and PLS for Gaussian distributed data can be determined from the eigenvalues of the components not included in the model (Jackson, 1992).

The Mahalanobis distance measures the degree to which data fit the calibration model. It is defined as

$$h_i = \{(x_i - \bar{x}_i)^T S^{-1}(x_i - \bar{x}_i)\}^{1/2}, \tag{41}$$

where S is the estimate of the covariance matrix and $\bar{x}_i$ is the estimate of the mean. If the calibration model data represent process operation at one set of operating conditions, and the process has shifted to a different set, then the statistic will show that data at this new operating condition cannot be classified with the calibration data. The use of this statistic in monitoring and analysis is illustrated in Examples 1 and 2 in Section VIII.

2. *Probability Density Function Methods*

PDF approaches represent a statistically formal way of accomplishing local kernel definition. Although intent and overall results are analogous to defining kernels of PCA features, considerable work currently is required for PDF approaches to be viable in practice. It is presently unrealistic to expect them to adequately recreate the underlying densities. Nevertheless, there are advantages to performing data interpretation based on direct PDF estimation and, as a result, work continues.

From both a theoretical and practical view, it is ideal to use Bayesian Decision Theory because it represents an optimal classifier. From a theoretical perspective, Bayesian Decision Theory offers a general definition of the pattern recognition problem and, with appropriate assumptions, it can be shown to be the basis of many of the so-called non-PDF approaches. In practice, however, it is typically treated as a separate method because it places strong data availability requirements for direct use compared to other approaches.

In brief, the Bayesian approach uses PDFs of pattern classes to establish class membership. As shown in Fig. 22, feature extraction corresponds to calculation of the a posteriori conditional probability or joint probability using the Bayes formula that expresses the probability that a particular pattern label can be associated with a particular pattern.

Class membership is assigned using some decision rule that is typically some inequality test performed on $P(\omega_i|x_k)$.

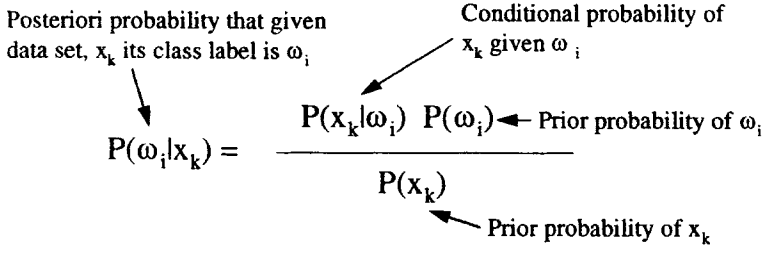


FIG. 22. Bayesian decision theory for pattern recognition.

The knowledge required to implement Bayes' formula is daunting in that a priori as well as class conditional probabilities must be known. Some reduction in requirements can be accomplished by using joint probability distributions in place of the a priori and class conditional probabilities. Even with this simplification, few interpretation problems are so well posed that the information needed is available. It is possible to employ the Bayesian approach by estimating the unknown probabilities and probability density functions from exemplar patterns that are believed to be representative of the problem under investigation. This approach, however, implies supervised learning where the correct class label for each exemplar is known. The ability to perform data interpretation is determined by the quality of the estimates of the underlying probability distributions.

Estimation of class-conditional densities is difficult. If the form of the probability function is known, then the problem becomes one of parameter estimation. Unfortunately, for process data interpretation, there is no basis for assuming the form of the function and one is forced to use nonparametric methods such as Parzen windows or the k-nearest neighbor method. More recently, radial basis function neural networks have been used to estimate probability density functions (Leonard and Kramer, 1991). These methods construct an approximation of the class-conditional densities by sampling the representation space containing the available training exemplars. The quality of these estimations is a strong function of the number of available exemplar patterns. For problems characterized by a large number of available exemplars and relatively small dimensionality, use of nonparametric techniques may be feasible. However, data interpretation is generally characterized by a large dimensionality and limited numbers of exemplars.

We include Bayesian Belief Networks (Kramer and Mah, 1994) in this category because they combine the Bayesian classification approach with a knowledge-based approach in the form of a semantic network. Ultimately, the approach attempts to calculate probabilities of events and subsequently assign labels to those events with the highest probabilities. Each node in the network represents the probability of some event. Events are linked based on some defining relationship, i.e., causality. Data interpretation is done by calculating and propagating probabilities exhaustively through the network. Although an exact computation may be intractable for realistic problems, several approximate inferencing methods have been proposed (Henrion *et al.*, 1991). Although a number of desirable properties of the belief network have been demonstrated and discussed, in practice the major consideration is availability of probability information. It requires considerable probability information that simply does not exist. Prototype demonstration results have indicated that rough and subjective estimates of proba-

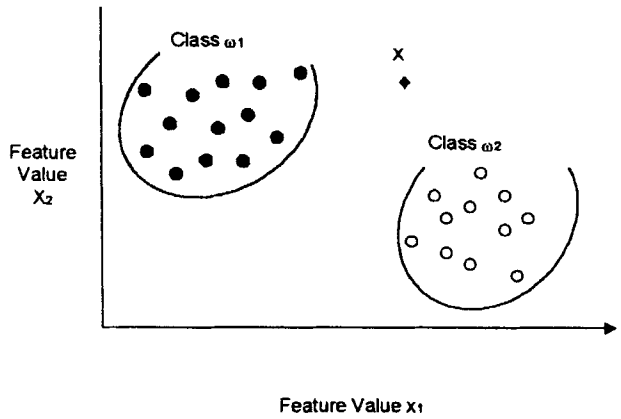


FIG. 23. Clustering characteristics of a two-class problem involving two observable features.

bilities may suffice, and therefore data requirements may be somewhat reduced.

3. *Clustering Methods*

As described by Oja (1995), kernel-based or clustering approaches are a general category distinguished by an ability to map highly nonlinear input data into prototype vector descriptions that form a multidimensional lattice. Unlike the nonlocal methods that draw upon latent variables generated from the application of dimension reduction and modeling techniques, clustering approaches are designed directly for pattern recognition. The underlying assumption is that patterns in a common pattern class exhibit similar features and that this similarity can be measured using an appropriate proximity index. Using this concept, patterns are assigned to the class of the pattern(s) to which they are most similar. This is illustrated in Fig. 23, where the concept is illustrated for two pattern classes in a two-dimensional representation space, R^2.

The training objective is to define an appropriate prototype vector description for a given set of sufficiently similar input data patterns and to establish a topological mapping of the prototypes such that each has a set of neighbors. Feature extraction is associated with forming the prototype descriptions and then placing them in an abstract feature space. Mapping is based on the similarity or nearness of a given pattern with labeled prototype descriptions, as shown in Fig. 23.

Clustering lacks the strict Bayesian requirement that input patterns be identifiable with a known prototype for which underlying joint distributions or other statistical information is known. Rather, the clustering approach

leverages similarities between pattern features so that patterns that have never been seen before and for which no statistical data are available still can be processed. This is illustrated in Fig. 23, by the diamond representing an arbitrary pattern, x.

The technology of proximity indices has been available and in use for some time. There are two general types of proximity indices (Jain and Dubes, 1988) that can be distinguished based on how changes in similarity are reflected. The more closely two patterns resemble each other, the larger their *similarity* index (e.g., correlation coefficient) and the smaller their *dissimilarity* index (e.g., Euclidean distance). A proximity index between the ith and jth patterns is denoted by $D(i, j)$ and obeys the following three relations:

1. $D(i, i) = 0$ for all i, using dissimilarity index
   $D(i, i) \geq \max_k D(i, j)$ for all i, using a similarity index
2. $D(i, j) = D(j, i)$ for all i, j
3. $D(i, j) \geq 0$ for all i, j

The Euclidean distance is the dissimilarity index most frequently used. It is characterized by invariance to translation and rotation,

$$D(z, x) = \|z - x\| = [(z - x)^T(z - x)]^{0.5}, \tag{42}$$

where z and x are two pattern vectors. Other proximity indices can be used, including the Mahalanobis distance (see section on statistical measures), the Tanimoto measure, and the cosine of the angle between x and z (Tou and Gonzalez, 1974).

The most straightforward decision rule that can be employed is what is referred to as the *1-nearest neighbor* (1-NN) rule. This rule assigns a pattern x of unknown classification to the class of its nearest neighbor z_i from the set of known patterns and pattern classes;

$$x \in \omega_i \text{ iff } D(z_{i,}\, x) = \min\{D(z_j, x)\} \quad \text{for } j = 1, N, \tag{43}$$

where N is the number of possible pattern classes. This was illustrated previously in Fig. 15.

This rule can be easily extended so that the classification of the majority of k nearest neighbors is used to assign the pattern class. This extension is called the *k-nearest neighbor* rule and represents the generalization of the 1-NN rule. Implementation of the k-NN rule is unwieldy at best because it requires calculation of the proximity indices between the input pattern x and all patterns z_i of known classification.

For process systems, one key practical advantage of clustering is the ability to work with limited and poorly distributed pattern data, typical of many labels of interest. Rather than attempting to partition unknown re-

gions of the representation space with projection-based methods, clustering approaches simply identify the structure in the existing representation space based on available data. These kernal-based methods are more robust to unreliable extrapolation, but they give up the potential for extrapolation. The focus on local data structures produces bounded decision regions and identifies regions of indecision (Kavuri and Venkatasubramanian, 1993; Davis *et al.*, 1996a). The notion of proximity or distance can be used to make additional decisions about the data patterns, such as novel pattern identification, or to establish gradations of uncertainty (Davis and Wang, 1995).

Algorithms fundamental to kernel-based approaches involve cluster seeking, the task of identifying class prototypes. In cluster seeking, as described in the Data Analysis section, pattern classes are defined based on discovery of similarities between patterns. For interpretation, the situation is reversed; that is, the pattern classes are defined a priori and the task is to identify the attributes of the exemplar patterns that cause an expert to assign them to specified classes. Supervised training is necessary to build the interpretation system based on an expert's collected knowledge in the form of annotated training sets. In practice, generating the annotated training set is probably the largest barrier to developing the system.

The applicability of a clustering algorithm to pattern recognition is entirely dependent upon the clustering characteristics of the patterns in the representation space. This structural dependence emphasizes the importance of representation. An optimal representation uses pattern features that result in easily identified clustering of the different pattern classes in the representation space. At the other extreme, a poor choice of representation can result in patterns from all classes being uniformly distributed with no discernible class structure.

Even when the patterns are known to cluster, there remain difficult issues that must be addressed before a kernel-based approach can be used effectively. Two of the more fundamental conceptual issues are the number and size of clusters that should be used to characterize the pattern classes. These are issues for which there are no hard and fast answers. Despite the application of well-developed statistical methods, including squared-error indices and variance analysis, determining the number and size of clusters remains extremely formidable.

Performance requires a careful balance between *generalization* and *memorization* (Davis and Wang, 1995). On one hand, a cluster can be defined too broadly, leading to misclassification of data. On the other hand, clusters can be so tightly defined that similar patterns in the same pattern classes cannot be classified with confidence. The appropriate balance is dependent on the desired functionality.

Differences in approaches rest with how the prototype vectors are gener-

ated, how the topological map is formed, and how the proximity indices are calculated. Most network performance comparisons have been made against back propagation neural networks (Kavuri and Venkatasubramanian, 1993; Leonard and Kramer, 1990; Whiteley and Davis, 1992a, b). Clustering approaches clearly outperform BPN primarily because of extrapolation errors. There have been some comparisons between clustering networks (e.g., Kavuri and Venkatasubramanian, 1993). These comparisons do not lead to any general recommendations based on classification performance. All are highly dependent on available data, and it appears that relatively close classification performances can be obtained when the various approaches are optimized for a particular static set of data patterns. From a practical point of view, differences manifest themselves more dramatically in the ability to initially establish and train the network and to accommodate new information as adaptive units.

It is possible to group approaches along several possible dimensions that include distance vs directional similarity measures, variations of winner-take-all strategies as illustrated earlier in Fig. 15, and fixed vs variable clustering capability. Fixed vs variable clusters appears to have the greatest practical significance because these categories reflect perspectives on initial set up of a cluster approach and adaptation, so they are considered here as the two primary categories.

(i) *Fixed Cluster Approaches.* This category represents those approaches that require a fixed number of clusters to be specified a priori. Probably the most commonly discussed approach is the radial basis function network (Moody and Darken, 1989; Leonard and Kramer, 1991). Referring to the earlier discussion of cluster seeking approaches in Section III, RBFNs using the exact same techniques can be readily extended for feature mapping. As discussed, an RBFN represents a neural network mapping procedure performing a nonlinear transformation at the hidden layer followed by a linear combination at the output layer. In terms of general structure, the RBFN is the same as the BPN. The extension to interpretation is accomplished in the same way as for BPN, namely, by declaring the numeric output to be 1 or 0, corresponding to the presence or absence of a particular interpretation. Distinctions are associated only with the basis functions identified with the hidden layer of nodes. The use of a Gaussian basis function rather than a sigmoid gives RBFNs local interpretation characteristics.

As described in detail in Section III, Input Analysis, training an RBFN for interpretation generally can be regarded as a two-step process consisting of clustering and supervised learning. A topological structure is formed by specifying the number of clusters that will be used and then using training data to converge on an optimal set of cluster centers via k-means clustering.

The selection of cluster number, which is generally not known beforehand, represents the primary performance criterion. Optimization of performance therefore requires trial-and-error adjustment of the number of clusters. Once the cluster number is established, the neural network structure is used as a way to determine the linear discriminant for interpretation. In effect, the RBFN makes use of known transformed features space defined in terms of prototypes of similar patterns as a result of applying k-means clustering.

As new data patterns become available, RBFNs can be updated incrementally, but the number of clusters cannot be changed without retraining the new network with all the data. This is a serious limitation from an adaptation standpoint. The second performance consideration stems from the formation of the cluster structure itself. Once the cluster centers have been established using k-means clustering, the cluster structure is formed using some nearest neighbor heuristic, i.e., 2-NN (Duda and Hart, 1973). Clustering based on k-NN approaches is effectively forming clusters by proximity and not class. As a result it is possible and likely to have mixed clusters that diminish the utility of the radial unit.

Ellipsoidal basis functions networks (EBFNs) represent a rather complex approach to addressing some of the shortcomings of RBFNs for interpretation (Kavuri and Venkatasubramanian, 1993). An EBFN is structured with a version of k-means clustering. Rather than establishing a cluster center based only on the winning cluster, the EBFN approach uses a fuzzy membership cluster neighborhood concept that is an extension of the Kohonen algorithm (Kohonen, 1989). The result is that there is no single winning cluster; rather, there is a degree of member for each cluster. The claim is that cluster centers are moved into different positions that enhance the classification performance. The problem of mixed clusters that result from the application of the k-NN algorithm is avoided by explicitly clustering by class. Increased performance using ellipsoidal rather than radial basis functions is not established. RBFN and EBFN approaches are compared

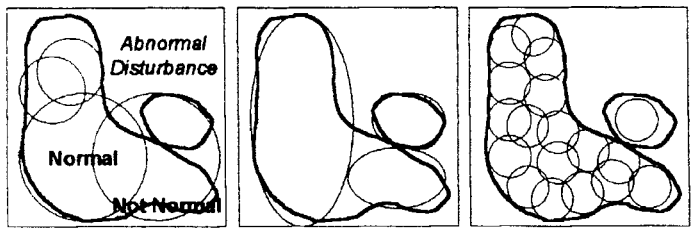


FIG. 24. RBFN, EBFN, and ART approaches.

graphically in Fig. 24 in terms of how they represent regions of normal and not normal plant operation.

We mention qualitative trend analysis (QTA) (Vedam *et al.*, 1998) in this section on clustering. QTA is a syntactic approach to numeric–symbolic mapping in that input data is mapped into a grammar of letters and words that capture a sequence of component shapes to form a pattern. Interpretation is accomplished by comparing the sequence of letters describing the input pattern signatures for known situations. A separate set of decision rules is invoked to monitor changes in the magnitudes of patterns.

(ii) *Variable Cluster Approaches.* The distribution of patterns available in chemical processes predestines the incompleteness of available data. Evolutionary adaptation is therefore necessary for online use because a practical data interpretation system must adapt smoothly to an ever-increasing amount of sensor data. The most effective measure of the maximum coverage of the interpretation capacity of a system is its ability to be updated on an ongoing basis.

To be effective, the kernel-based approach must be adaptive to the situations that have not occurred previously while maintaining information already residing in the system. This requires that the system be plastic so that it can incorporate new knowledge while maintaining a level of stability relative to the existing knowledge. An appropriate stability–plasticity balance is critical to adaptation. This implies that the number of clusters should be variable and that exiting clusters should be adjustable. Variable cluster approaches address this with the capacity for adding new clusters or removing unused or unimportant clusters.

Adaptive Resonance Theory (ART) (and, in particular, ART2), developed for both binary and analog input patterns, is the only kernel-based architecture that addresses variable clustering capacity and supports the information management components required for determining whether a new cluster is needed or whether an existing cluster should be modified. ART represents a philosophy as much as a model and continues to be refined with time (Grossberg, 1982, 1987a, 1987b, 1988). The theory has been substantiated in a series of neural network models called ART1, ART2, and ART3 (Carpenter and Grossberg, 1987a, 1987b, 1988, 1990).

ART2 forms clusters from training patterns by first computing a measure of similarity (directional rather than distance) of each pattern vector to a cluster prototype vector, and then comparing this measure to an arbitrarily specified proximity criterion called the *vigilance.* If the pattern's similarity measure exceeds the vigilance, the cluster prototype or "center" is updated to incorporate the effect of the pattern, as shown in Fig. 25 for pattern 3. If the pattern fails the similarity test, competition resumes without the node

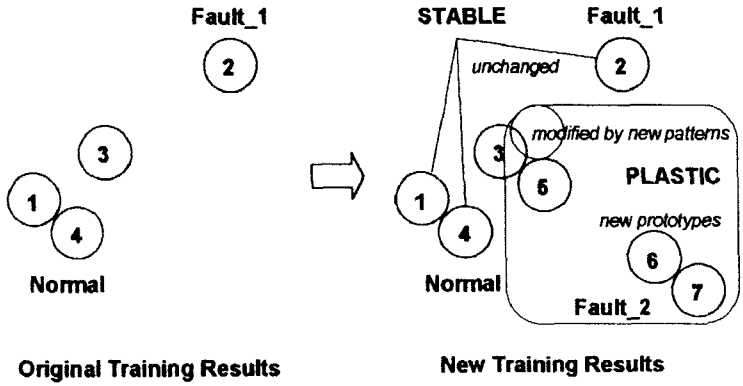


FIG. 25. Clustering in ART.

corresponding to the cluster. This match/reset operation produces a crisp decision boundary for each cluster. When no match is found between an input pattern and an existing cluster, a new, uncommitted node is allocated for the input, which then becomes the prototype for the new cluster. This is shown in Fig. 25 in terms of the formation of new clusters 5, 6, and 7. No existing node is altered by the pattern. The new cluster then participates in future competitions with previous nodes. The size of the hyperspherical cluster is determined by the vigilance value and characteristics of the training data. Stability–plasticity issues are addressed because the algorithm comprises the information management mechanisms to decide when to form a new cluster and how to adjust an existing cluster. Unlike RBFNs and EBFNs, the algorithm asks which cluster is most similar and then asks and answers the question about the pattern being close enough.

No single ART2 equation or set of equations provides the desired adaptive properties for the system. Rather, it is the synergistic effect of this architecture as a whole that gives rise to these properties. A description of the ART network applied to process situations can be found in Whiteley and Davis (1996); a complete description of an ART network can be found in Carpenter and Grossberg (1987a).

VI. Symbolic–Symbolic Interpretation: Knowledge-Based System Interpreters

If there is sufficient operating history to provide the labeled data and sufficient resources to label it, then it is desirable to rely on this information for data interpretation because it reflects the actual operation. With respect

to Fig. 3, this is the case of using a numeric–symbolic interpreter for assigning the labels of interest. However, when operating histories (or trustworthy simulations) do not provide sufficient exemplars to develop an adequate mapping using training based methods alone, knowledge-based systems (KBSs) can be used for interpretations, and they are especially useful for extending intermediate interpretations to the labels of interest (McVey *et al.,* 1997).

There are degrees of capacity, however, that need to be taken into account. If operating history is less than fully sufficient, but is nevertheless plentiful, training based methods may be able to carry *most* of the burden of interpretation by providing effective intermediate labeling. Then the requirements for the KBS become very modest. At run time, the KBS will perform the final mapping to appropriate labels, typically through some form of *table lookup*: e.g., simple rule sets or a decision tree. On the other hand, if the intermediate labeling requires considerably more analysis to reach the desired classifications, then the KBS portion might be very extensive. This sequential integration of a numeric–symbolic interpreter and the KBS interpreter is illustrated in Fig. 4, where the measurement data is mapped into intermediate interpretations that are then input to a KBS interpreter. The KBS interpreter ultimately produces the final interpretations of interest.

In lieu of data exemplars, KBS approaches use expert descriptions to accomplish the mapping. Explicit items of knowledge provide pathways between combinations of symbolic input values and possible output labels. KBS approaches provide both means for capturing this knowledge, and control constructs for exploiting efficient strategies used by experts. There have been many reports on knowledge-based system constructs (e.g., Quantrille and Liu, 1991; Rich and Knight, 1991; Shapiro and Eckroth, 1987; Stephanopoulos and Davis, 1990, 1993; McVey et al., 1997; Stephanopoulos and Han, 1994). A review of intelligent systems in general is given in Davis *et al.* (1996b).

Consider the example in Fig. 26 of determining if there is a problem with the feed injection system of a fluidized catalytic cracking unit (Ramesh *et al.,* 1992). In this example, there is a set of rules that relate combinations of process observations to establish or reject this possibility.

The label of interest is *Feed Injection System Problem*. The *if, and,* and *or* statements relate specific process observations that can establish that there is a likelihood of an injection system problem. *Injector header pressure* is a process measurement and *abnormal* is an intermediate label of interest. The label *abnormal* can be determined by developing a numeric–symbolic interpreter that maps injector header pressure data as either normal or abnormal.

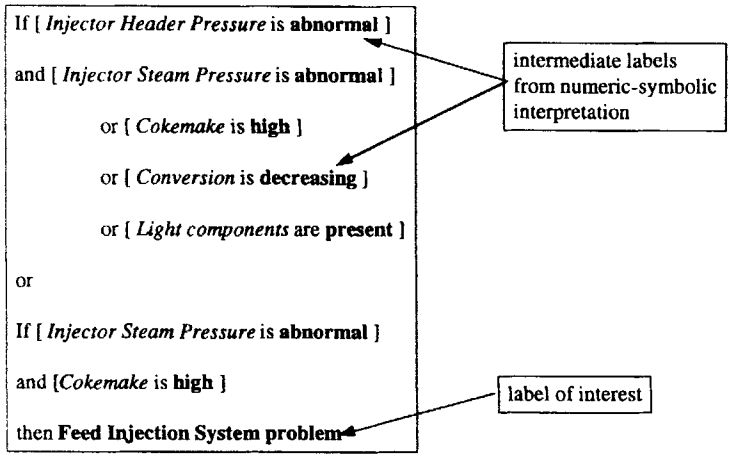


FIG. 26. A fluidized catalytic cracking unit example of combining numeric–symbolic and symbolic–symbolic interpretation (Ramesh *et al.*, 1992).

It is common to view KBSs from a mechanistic perspective only and, as a result, to oversimplify the construction of a system and miss out on potential capabilities. At a mechanistic level, the constructs for representing knowledge are fundamentally lists, rules, objects, and so on, and constructs for manipulating knowledge are matching, chaining, attribute inheritance, and so forth. This view is in danger of missing the importance of higher-level analysis that considers types of knowledge, types of problem-solving activity, and ways of organizing knowledge. Diagnostic reasoning, for example, relies on forms of knowledge and has processing strategies distinct from the knowledge forms and processing strategies most useful for, for example, response planning. Constructing a KBS by choosing a mechanism, prior to higher-level analysis of the task to be performed or of the reasoning strategies and types of knowledge required, is likely to lead to disappointment, especially when constructing large systems (McVey *et al.*, 1997).

A useful characteristic of KBS interpreters is that of generality of coverage: That is, they use representations and manipulations to reach correct conclusions without having to explicitly enumerate all possible relationships between input combinations and conclusions. Situations can be covered that are theoretically or realistically possible, but that have not been observed or recorded. Moreover, a well-designed KBS degrades gracefully when presented with novel or unanticipated situations. The mapping performed by a KBS might be direct, essentially table lookup, or it might employ problem-solving using semantic representations and thereby be capable of providing broad generality of coverage.

Many applications of knowledge based systems have been described

(e.g., Antsaklis and Passino, 1993; Badiru, 1992; Mavrovouniotis, 1990; Myers *et al.,* 1990; Reklaitis and Spriggs, 1987; Rippin *et al.,* 1994). The absence of extensive training data, together with the value of the anticipated conclusions, explains the wide interest in the use of KBS systems for fault classification and corrective action planning.

A. Families of KBS Approaches

Representing knowledge requires selecting a set of conventions for describing devices and relations (i.e., causal relations, processes, and so on). This selection strongly affects the ability of the KBS to make appropriate connections between relations. The advantages and limitations of various knowledge representation schemes vary considerably with the characteristics of the problem, the types of knowledge, and the requirements for manipulation. In general, no one representation emerges as uniformly best.

Two broad categories of approaches have emerged and are in wide use as symbolic–symbolic interpreters: *semantic networks* and *tables.* Semantic networks refer to a very large family of representations comprising nodes to represent objects or concepts and nodal links representing the relations between them. Semantic networks are particularly useful for representing taxonomies, spatial relations, causal relations, and temporal relations. In general, semantic networks take on many forms, including hierarchies, decision trees, relationship structures, directed graphs, and others (Stephanopoulos and Davis, 1990). The other very popular representation is the table. In tables, input and output labels (variables) are structured as one-to-one relations. In the simplest case, a given input label maps directly into an output. The strategy involves simply searching the input table for the relevant conditions and then noting the output. Most applications are more complicated in that input–output conditions exhibit more complex relations. For example, an input can be associated with more than one output, an input can only partially explain an output, or multiple inputs can explain an output. With these kinds of complex relations, the procedure for mapping is more complicated and involves constructing a best explanation for a given set of inputs by accounting for implications and incompatibilities (Josephson and Josephson, 1994).

With the view that a KBS interpreter is a method for mapping from input data in the form of intermediate symbolic state descriptions to labels of interest, four families of approaches are described here, each offering inference mechanisms and related knowledge representations that can be used to solve interpretation problems: namely, model-based approaches, digraphs, fault trees, and tables. These methods have been heavily used

for diagnostic applications in process operations. Of these, at one extreme are model-based methods, using quantitative and qualitative simulation that does very little to constrain the allowable possibilities. By using behavioral models for the individual components in a process operation, model-based methods enable consideration of many kinds of possible behaviors. The advantage is that by manipulating models it is possible to discover input–output relationships that have not been pre-enumerated. The disadvantage is that descriptions of large numbers of behaviors can be generated, many of which will not be realistic or relevant under the circumstances and, therefore, the number of output considerations must be carefully managed. At the other extreme are table methods that fully constrain the possible input–output situations. Such systems do not allow for relation-path considerations at run time because they compile relation pathways into direct input–output relations. The advantage here is that table lookup is very direct and performance levels can be very high. The disadvantage is that correct performance is completely dependent on the ability to preenumerate all the relations. Digraphs and fault trees take intermediate positions in the trade-off between flexibility and efficiency.

1. Model-Based

The term *model-based* can be a source of confusion because descriptions of any aspects of reality can be considered to be models. Any KBS is model based in this sense. For some time, researchers in KBS approaches (Venkatasubramanian and Rich, 1988; Finch and Kramer, 1988; Kramer and Mah, 1994; McDowell and Davis, 1991, 1992) have been using model-based to refer to systems that rely on models of the processes that are the objects of the intent of the system. This section will avoid confusion by using the term *model* to refer to the type of model in which the device under consideration is described largely in terms of components, relations between components, and some sort of behavioral descriptions of components (Chandrasekaran, 1991). In other words, model-based is synonymous with *device-centered.* Figure 27 shows a diagram displaying relationships among components. The bubble shows a local model associated with one of the components that relates input–output relationships for flow, temperature, and composition.

A large number of model-based systems use either qualitative or quantitative simulation, such as FAULTFINDER (Kelly and Lees, 1986) or EAGOL (Roth, Woods, and Pople, 1992). These systems simulate normal behavior and compare the simulation results with observations, they simulate faults and compare simulation results with detected symptoms, or they interleave simulation with observation comparing the two to dynamically track normal and abnormal states. It is computationally very expensive to

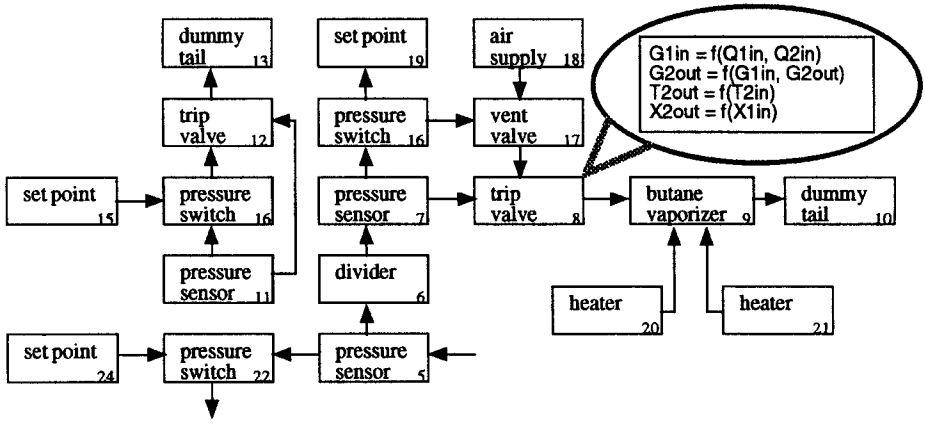


FIG. 27. Components and connections. [Drawn from J. S. Mullhi, M. L. Ang, and F. P. Lees, *Reliability Engineering and System Safety,* **23,** 31 (1988).

apply high-detail quantitative simulation to large and complex systems because of the large number of causal events that must be computed. Furthermore, quantitative simulations often do not capture malfunction situations adequately. Qualitative simulation, while capturing an appropriate level of fault description and saving computation compared with quantitative simulation, still typically results in generating too many malfunction candidates for large-scale systems, but has the advantage of flexibility, allowing causal relations to propagate forward and backward readily (Kuipers, 1994; Catino and Ungar, 1995; McDowell and Davis, 1991; Venkatasubramanian and Vaidhyanathan, 1994). One main source of computational expense in qualitative simulation arises because lack of precision leads to ambiguity in projecting the future, which leads to considering combinations of alternative future possibilities. Of course, local use of simulation is practical for large-scale systems when it is used sparingly in the focused pursuit of specific subgoals.

2. *Fault Trees and Digraphs*

Fault trees and digraphs were the earliest structures used for capturing causal relations in diagnostic KBS. A fault tree is a graph representing the causal dependencies of symptoms and faults. Confidence values are used to control and optimize the inference mechanism as well as to provide additional information for decision making. Early examples of the use of fault trees in diagnostic expert systems includes the work of Ulerich and Powers (1987) and Venkatasubramanian and Rich (1988). Because the relationship between faults and symptoms is a directed graph, not a tree (Kramer and Mah, 1994), a basic event in a fault tree may result from more than one event and

typically can contribute to more than one event. This creates a tangled graph, which may contain loops that make diagnostic inference complicated. Figure 28 provides an illustrative segment of a fault tree.

A digraph contains equivalent information to fault trees, but is written in terms of system variables; a variable is given as a node and the relationship between two variables is given as an edge. The digraph also can contain nodes that represent fault conditions rather than system variables—identical to fault trees. Positive and negative influences and their magnitudes, or conditional probabilities, are represented by annotations associated with the edges. Examples of digraphs include belief networks (Pearl, 1988; Kramer and Mah, 1994), possible cause–effect graphs (Wilcox and Himmelblau, 1994), and signed directed graphs (SDGs) (Mohindra and Clark, 1993). Diagnosis is carried out essentially by determining the maximal strongly connected subgraph. Special operators are used to handle loops. The inference mechanism involves enumerating combinations of the unmeasured nodes. For each unmeasured node, the system generates a set of patterns that correspond to all possible states. Execution time is exponential in the number of unmeasured nodes. However, the unmeasured nodes are necessary to maintain the diagnostic resolution. For a large and complex system with many unmeasured variables, diagnosis using only this approach is too time consuming. Figure 29 illustrates a digraph where the arcs represent directed causal connections and the nodes represent possible events.

3. *Tables*

Widely used in rule-based systems, input–output matching approaches are used in conventional expert systems going back to the earliest systems.

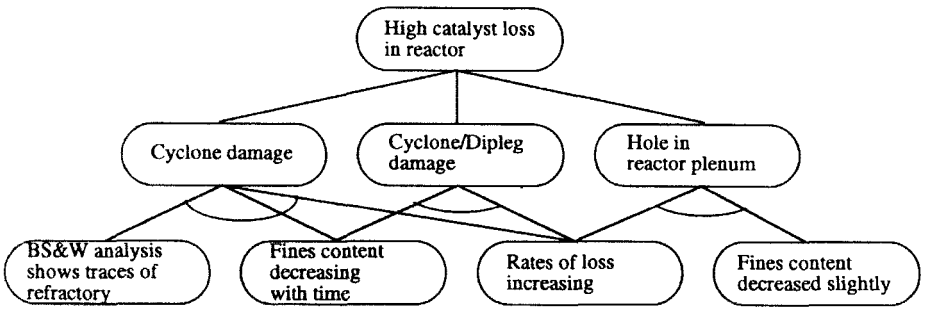


FIG. 28. Faul tree. [Drawn from G. Stephanopoulos, Ed., *Knowledge-Based Systems in Process Engineering*, Vol. 1, Cache Case Studies Series, CATDEX, Cache Corporation, Austin, TX (1988).

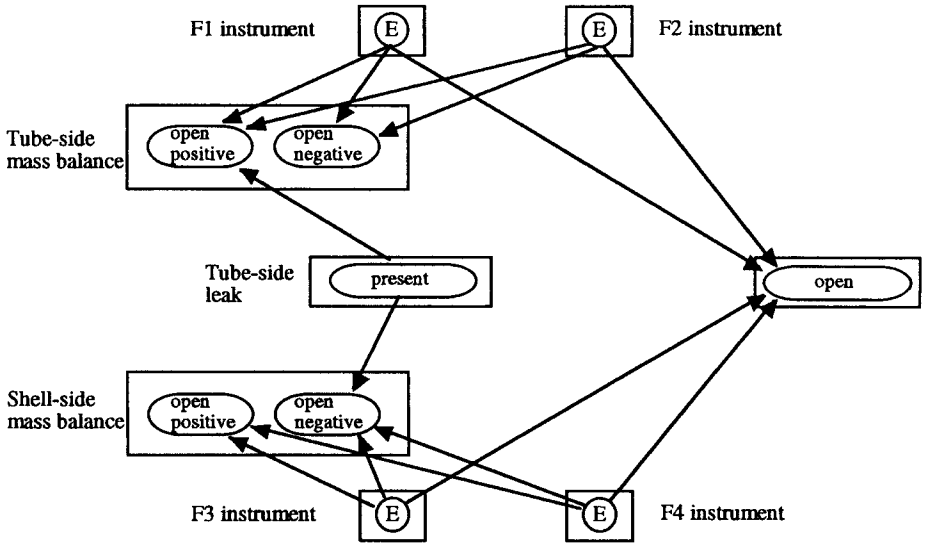


FIG. 29. Digraph. [Drawn from N. A. Wilcox and D. M. Himmelblau, *Comput. CHem. Eng.* **18,** 103 (1994).

There has been some interest in compiling digraphs and fault trees into rules for use in solving diagnostic problems. Kramer and Palowitch (1987) use a method that analyzes the digraph structure offline to derive simple diagnostic tests to be used online. Ragheb and Gvillo (1993) have developed a methodology for using fault-tree analysis techniques to represent knowledge, which is in turn translated to goal trees and then encoded as production rules. Computational load during run time is reduced considerably. For a large and complex system, however, compilation itself is computationally very expensive.

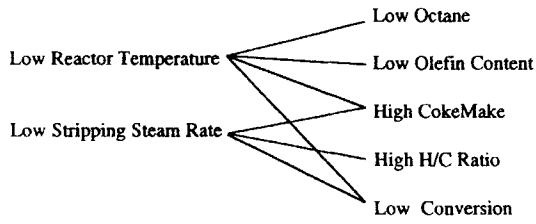
Figure 26, shown earlier, is a simple form of input mapping called table lookup. A more complicated inference mechanism is illustrated in Fig. 30. Here we see a simple example from a fluidized catalytic cracking unit in which multiple product quality attributes can be explained by multiple operating parameters (Ramesh *et al.,* 1992).

Overall, each of these KBS approaches has its place, depending on the needs of the problem and the characteristics of the process. However, the advantages, limitations, and roles of the various methods must be clarified so that problem-solving efficiency can be achieved along with the ability to reach correct conclusions despite challenging circumstances: knowledge that is incomplete; data that are incomplete, unreliable, or extremely plentiful; or situations that are untested or completely unanticipated.

Product Quality to be Explained

[Conversion: Low]
[Cokemake: High]
[H/C Ration: High]

Explanatory Relationships



Low Stripping Steam Rate is the better explanation given the current data

FIG. 30. A form of table lookup requiring assembly of best hypothesis.

VII. Managing Scale and Scope of Large-Scale Process Operations

KBSs can be viewed with increasing levels of commitment to problem solving. At the level described in the previous section, a KBS accomplishes symbolic–symbolic mappings between input and output variables analogous to the numeric–symbolic mappings of approaches such as neural networks and multivariate statistical interpreters. For each problem-solving task, the particular numeric–symbolic or symbolic–symbolic approach is based on the task and the knowledge and data available.

At another level, certain KBS approaches provide the mechanisms for decomposing complex interpretation problems into a set of smaller, distributed and localized interpretations. Decomposition into smaller, more constrained interpretation problems is necessary to maintain the performance of any one interpreter and it makes it possible to apply different interpretation approaches to subparts of the problem. It is well recognized that scale-up is a problem for all of the interpretation approaches described. With increases in the number of input variables, potential output conclusions, complexity of subprocess interactions, and the spatial and temporal distribution of effects, the rapidity, accuracy, and resolution of interpretations can deteriorate dramatically. Furthermore, difficulties in construction, verification, and maintenance can prohibit successful implementation.

A. Degradation of Interpretation Performance

Consider the following simulation-based study of the performance degradation of an Adaptive Resonance Theory numeric–symbolic interpreter for identifying malfunctions from input process data (Ali *et al.*, 1997). The data for this study were generated from a training simulator of a fluidized catalyst cracking unit (FCCU). The model simulates a stacked regenerator/disengager design. The regenerator, at the bottom, operates at a higher pressure. An external riser standpipe, a single-stage steam stripper standpipe, and a single-stage steam stripper complete the converter. A steam turbine powered centrifugal air blower supplies the regeneration air. The waste heat boiler develops steam from the surplus heat in the flue gas. There is also one main fractionator with a light cycle oil (LCO) stripper as part of the model.

The simulator models the FCCU, generating output from 110 sensors every 20 seconds. In all, 13 different malfunction situations were simulated and are available for analysis. There are two scenarios for each malfunction, slow and fast ramp. Table II provides a list and brief description of each malfunction. A typical training scenario for any fast ramp malfunction simulation had the landmarks listed in Table III. Similarly, a typical training scenario for any slow ramp malfunction simulation is shown in Table IV. For both the fast and slow ramp scenarios, there was data corresponding to 10 min of steady-state behavior prior to onset of the faulty situations.

With reference to Figs. 24 and 25, Fig. 31 illustrates a pattern representa-

TABLE II
List of Malfunctions Available for Analysis

I.D.	Malfunction description
kldft	KO drum level transmitter drifts.
mfdft	Main fractionator transmitter drifts.
absd	Air blower speed drop.
ftdft	Air blower flow transmitter drifts.
cinc	Carbon in the gas oil feed increases.
spar	Slurry pump performance degrades.
lpar	LCO pump performance degrades.
npar	Naphtha pump performance degrades.
hpar	HCO pump performance degrades.
sdft	Slurry flow sensor drifts.
ldft	LCO flow sensor drifts.
ndft	Naphtha flow sensor drifts.
hdft	HCO flow sensor drifts.

TABLE III
FAST RAMP TRAINING SCENARIO

Time	Landmarks
0:00	Simulation begins.
1:00	The faulty ramp starts.
3:00	The ramp is over and controllers take over.
10:00	The simulation ends.

tion space in ART for an evolving data set. The numbers represent patterns of data that occur over time and are compared to known pattern descriptions represented by the clusters, labeled *Normal, fault*$_1$, and *fault*$_2$.

As shown, the data patterns 1 and 2 are classified as Normal with high certainty, as they lie within the boundaries of the normal class. However, 3 and 4 are classified as normal with medium certainty, as they lie outside the normal region, but their similarities are closest to the normal cluster. Similarly, the data pattern represented by 5 is classified as fault$_2$ with medium certainty and the data patterns represented by 6, 7, and 8 are classified as fault$_2$, and so on. If the data are collected every 20 seconds as in the case study, the dynamic interpretation is tabulated as shown in Table V, with the labels in italics representing the correct class and appropriate certainty. An x means there was not an interpretation with this certainty.

To illustrate that increasing output dimensionality can lead to performance problems for a numeric–symbolic interpreter, the ART interpreter was trained with various subgroupings of the 13 malfunction scenarios. With 13 malfunction scenarios and one normal scenario, the maximum output dimension is 14.

The subgroups were determined based on a hierarchical decomposition strategy for a FCCU (Ramesh *et al.*, 1992). The FCCU was decomposed into four separate units: feed.system, catalyst.system, reactor/regenerator.system, and separation.system as shown in Fig. 32. Each of these units was further divided into more detailed functional, structural, or behavioral

TABLE IV
SLOW RAMP SCENARIO

Time	Landmarks
0:00	Simulation begins.
1:00	The faulty ramp starts.
46:00	The ramp is over and controllers take over.
1:00:00	The simulation ends.

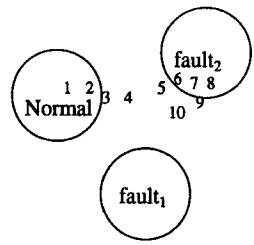


FIG. 31. ART pattern representation space for an evolving data set.

categories, going into three or, sometimes, four levels of detail. Each tip node corresponds to a specific malfunction such as feed temperature controller problem or stripper level problem. The particular segments of the decomposition that involve the 13 malfunctions are also shown in Fig. 32. As illustrated, the 13 malfunctions reside in three distinct groups.

Group 1: Loss of wet gas flow. This grouping of malfunctions is concerned with events leading to the loss of flow through the compressor which draws off the fractionator overhead vapors.

Group 2: Coke imbalance. This grouping considers malfunctions leading to a difference between the rate at which coke accumulates on the catalyst and the rate at which it is burned off. A coke imbalance is associated with a reduction of oxygen, which can be caused by a loss of combustion air or through an increase in the conradson carbon in the gas oil feed to the unit.

Group 3: Loss of pump around flow. This grouping comprises malfunctions associated with the circulation of fluids through and around the main fractionator. The loss of one or more pump flows (slurry, light cycle oil, naphtha, and heavy cycle oil) leads to the loss of some of the fractionator heat sink.

Table VI shows an example for detecting and isolating problems due to an air blower speed drop (fast ramp) *absd*. The behavior illustrated by this

TABLE V
DATA INTERPRETATION TABLE ONLINE

Time (min)	High certainty	Medium certainty
0.00–0.67	*Normal*	
1.00–1.33	x	*Normal*
1.67–1.67	x	$fault_2$
2.00–2.67	$fault_2$	
...	...	...

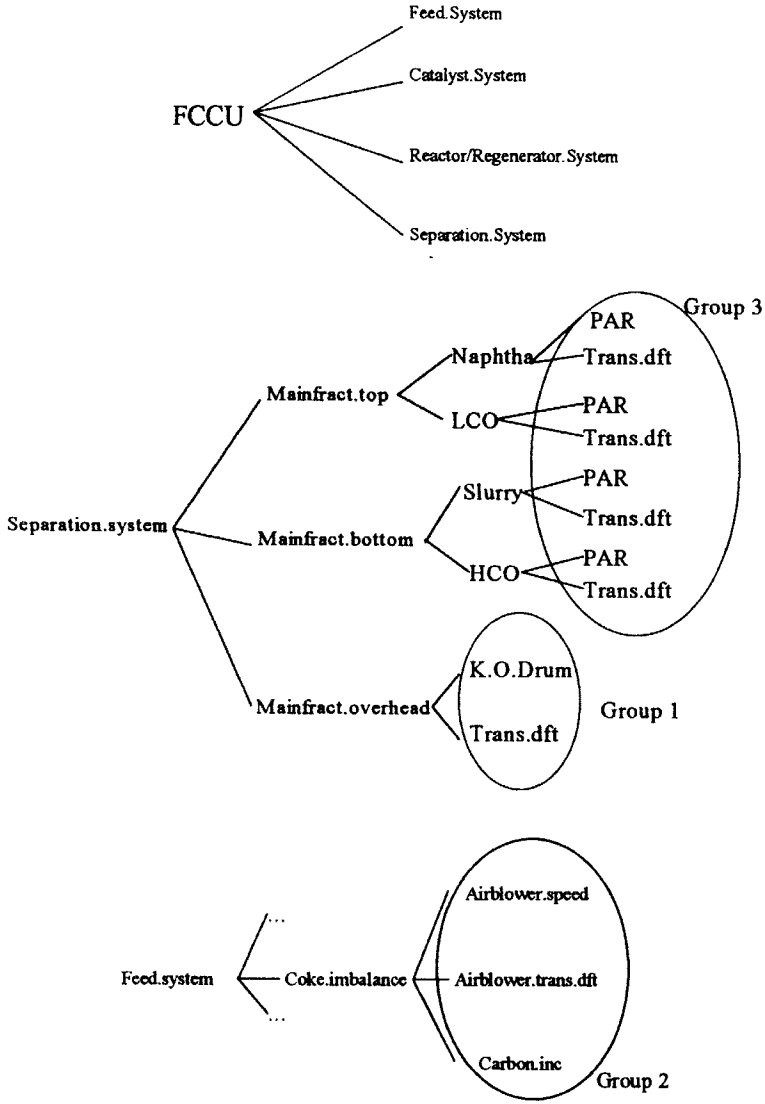


FIG. 32. Hierarchical decomposition to form malfunction groupings.

example was consistently observed for the other malfunction scenarios. When trained only with group 2 malfunction scenarios, an output dimension of 4, the air blower malfunction is both detected with high certainty and isolated with medium certainty within 4.67 min. When the interpreter is asked to provide extended coverage for both group 1 and group 2 malfunc-

TABLE VI
PERFORMANCE OF ART FOR AIR SPEED DROP (FAST RAMP) *absd*

Time	High certainty	Med certainty
Grouped training (Group 2) output dimensionality is 4.		
0.00–4.33	*Normal*	mixed
4.67–26.00	x	**absd**
Training with group 2 and 1 output dimensionality is 6.		
0.00–1.00	*Normal*	mixed
1.33–4.67	*mixed*	Normal
5.00–26.00	x	**absd**
Training with group 2 and 3 output dimensionality is 11.		
0.00–1.00	*Normal*	mixed
1.33–4.67	*mixed*	Normal/cinc
5.00–16.33	*x*	**absd**
16.67–26.00	*mixed*	ndft/ldft
Complete training output dimensionality is 14.		
0.00–2.33	*Normal*	mixed/cinc
2.67–4.33	*mixed*	ldft
4.67–16.00	x	**absd**
16.33–21.33	x	*mixed*
21.67–26.00	*mixed*	hdft

tions, raising the output dimension to 6, some mixed clusters are formed (clusters constructed from patterns from both normal and malfunction classes), leading to a mixed high certainty normal classification from 1.33 to 4.67 min and a deterioration in detection and isolation times to 5.00 min. However, when the dimension is extended to 11 with group 2 and group 3 malfunctions, performance significantly deteriorates. Over a period of 26 minutes, detection of changes from normal remains mixed; the correct interpretation with medium certainty is made between 5.00 and 16.33 minutes, but changes to an incorrect mixed conclusion about naphtha and LCO flow sensor drifts. When the dimension is raised to 14, further deterioration with mixed and incorrect medium-certainty conclusions are drawn.

The example clearly brings out the performance degradation as output scale increases. This particular example uses ART as the numeric–symbolic interpreter. Similar degradation in performance can be shown with other interpreters such as PCA and PLS (Ali *et al.*, 1997; Wold *et al.*, 1996).

The example brings out performance degradation associated with a numeric–symbolic interpreter. The principles apply to symbolic–symbolic in-

terpreters as well. Both digraph and fault tree methods have many problems when applied to physically complex systems (Wilcox and Himmelblau, 1994; Chen and Modarres, 1992). Kramer (1987) discussed scale-up problems with symptom pattern matching methods using quantitative models with non-Boolean reasoning. Catino and Ungar (1995) described the increase in time and memory required for qualitative simulations as the device complexity increases. In general, it is not possible to accurately predict when scale-up problems will become significant because performance is heavily dependent on the interactions occurring in the particular plant operation.

The complexity of large-scale process operations sometimes calls for applying, within a single application, a variety of methods to address different information-processing requirements for interpreting diverse behaviors of diverse components, to realize a range of different types of interpretations, or to provide redundancy to raise confidence in conclusions. The inclusion of components reflecting multiple methods and the demands of developing large-scale software essentially require some form of decomposition to manage and distribute decision making into a set of focused, but coordinated, problem-solving modules. While most numeric–symbolic interpretation methods are relatively effective at reducing input complexity, the scale of complex processes still requires constraining input variables into appropriate groupings to reduce the number of variables considered by any one interpreter. Similarly, a large number of possible output interpretations virtually requires that considerations be localized so that interpretations from any one interpreter are not confounded by too many possible conclusions. Thus, it is advantageous if the overall problem, and the knowledge needed to do it, can be decomposed into manageable modules, where each module is constructed by adopting the best available method for its local subproblem. A global perspective must also be maintained so that interpretations from localized modules can be coordinated and a global evaluation produced (Prasad *et al.*, 1998).

One key approach is to design a decision-support system made up of modules that mirror the modular structure of the process system. Interpretations then can be isolated on the basis of local behaviors, goals, and locally relevant input without generating long chains of inferences or hypothesizing complex combinations of behaviors. An important benefit of this organizing principle is that it facilitates knowledge acquisition for symbolic–symbolic interpreters or the training and configuration effort associated with numeric–symbolic interpreters. With knowledge about the process partitioned into manageable chunks and associated with named and identifiable process-system elements, the development of interpretation systems then becomes modular itself. Each modular component directs a small, manageable

effort focused on the behaviors and potential malfunctions of its corresponding system element. This same modular property facilitates debugging and maintenance of a plantwide decision-support system; debugging and maintenance are especially important design considerations for constructing large systems.

B. Hierarchical Modularization

Engineered processes typically reflect hierarchical whole–part designs (system–subsystem, devices–components). Any behavior in a given system always can be associated with one or more of its devices or device components. So it is convenient to associate process behaviors, various normal and abnormal situations, and so on with corresponding systems and/or devices at various levels, and it is convenient and natural to use the whole–part decomposition as a basis for modularizing a large-scale interpretation system. For example, typically what is wanted in diagnosis is a fault description that is as specific as possible. So if diagnostic reasoning establishes that a device is malfunctioning, it is usually desirable to know if the malfunction can be further localized to a part of the device, or even to a specific malfunction mode of a specific part. A reasonable way to modularize large-scale processes, therefore, is a hierarchy of categories, ordered by interpretation specificity, where most levels of the hierarchy mirror the whole–part organization of the process system and bottom levels represent specific components or specific component behaviors. Such a hierarchy of malfunction categories need not exhibit a strict tree structure, because a component may belong to more than one subsystem.

With care, these categories can be designed to respect strict set-inclusion semantics so that category–subcategory will correspond faithfully with set–subset. Besides providing clarity and discipline for knowledge acquisition and debugging, respecting set-inclusion semantics has distinct advantages for interpretation. What is required is that if a malfunction belongs to a class in the hierarchy, then it belongs to any superclass in the hierarchy. That is, for malfunction classes X and Y, and for the partial-order relation whereby elements of the malfunction hierarchy are ordered child-to-parent, we require that

$$\forall X \forall Y \text{ such that } X \propto Y, x \in X \rightarrow x \in Y. \tag{44}$$

This requirement is logically equivalent to the requirement that if a malfunction does not belong to a certain category, then it does not belong to any subcategory of it.

For example, if a valve is not malfunctioning, then it is not stuck

closed, and if the cooling system is not malfunctioning, then no component is malfunctioning. This may seem somewhat counterintuitive, because in general a device might continue to function properly, even though a component has malfunctioned; e.g., because of redundancy or compensation. Yet the requirement becomes reasonable if the convention is adopted that a component is not considered to be malfunctioning unless it impairs the functioning of the device of which it is a component. Alternatively, the convention may be that a device is considered to be functioning properly only if every component is functioning properly. In either case, if a malfunction does not belong to a category, then it does not belong to any subcategory. This property permits a diagnostic strategy where ruling out a high-level malfunction category rules out all of its subcategories. If it can be determined that there is no apparent problem with the cooling system, then problems with cooling-system components need not be considered.

A variety of hierarchical structures going back many years have been used to manage scale and achieve modularization. Finch and Kramer (1988) used an approach for diagnosis in which each hypothesis in the hierarchy corresponds to a control-loop system. Interdependencies among the systems are identified by postulating system malfunctions and determining which other systems are directly affected by a malfunction. The inference mechanism is similar to that of a sign-directed graph, in which the dependencies between systems are arcs. Even though a degree of modularization is achieved, the overall system suffers the same scaling problems as digraph systems because of the inference mechanism. Chen and Modarres (1992) developed a diagnostic and correction-planning system, FAX, which uses a goal-tree/success-tree hierarchical model. Malfunction categories are defined using the top plant goal or objective, and then decomposing the goal vertically downward to progressively more detailed subgoals. Table lookup and Bayes' theorem are used in the inference mechanism. Davis (1994); McDowell and Davis (1992); Ramesh *et al.* (1988); Prasad and Davis (1992); and Prasad *et al.* (1998) have extensively used an approach called *hierarchical classification* (HC) where the event or situation is classified as belonging to one or more categories, the categories being hierarchically organized according to levels of specificity. The hierarchical categories make use of combinations of system–subsystem, component–subcomponent, component–mode, and event–subevent decompositions customized for the problem of interest.

Together, these decomposition approaches argue for combining functional, goal, and structural perspectives of the plant, as needed, in constructing interpretation systems. With any of these hierarchical approaches, each node in the hierarchy can be construed to define a localized interpreter.

Interpretation requirements can be clearly defined and methods can be customized to meet the needs of the local decision. That is, a localized interpreter can be constructed to use a method that is most effective, given available knowledge and information, for inferring conclusions about the problem defined by the node—malfunction category rejected, goal achieved, subsystem functioning properly, and so on. Each such module will then provide information that influences the overall problem solving in a well-understood way. Figure 33 shows a portion of a hierarchical classification system for diagnostic decision support with a bubble showing a local module associated with one of the nodes (see Fig. 26).

In practice, high-level malfunction categories typically are monitored and evaluated continuously. Lower-level categories are evaluated only if high-level categories are established. The evaluation of malfunction situations may then proceed through the hierarchy from general to specific. For support of decision making, notification may be given to a higher level situation, but advice will be given based on the lowest level, which relates most directly to specific corrective actions that can be taken.

With a similar intent of decomposing problems to reduce dimensionality and to make interpretations more manageable, Wold, Kettaneh, and Tjessem (1996) and MacGregor *et al.* (1994) have reported on an approach called *blocking,* used in conjunction with multivariate PLS and PCA. The motivation for developing the technique of blocking stems from the problem that PLS and PCA models are difficult to interpret with many variables, loadings, and scores. Referred to as *hierarchical multiblock* PLS or PCA models, these hierarchical models group variables into meaningful groups or blocks. Blocking then leads to two levels where the upper level models

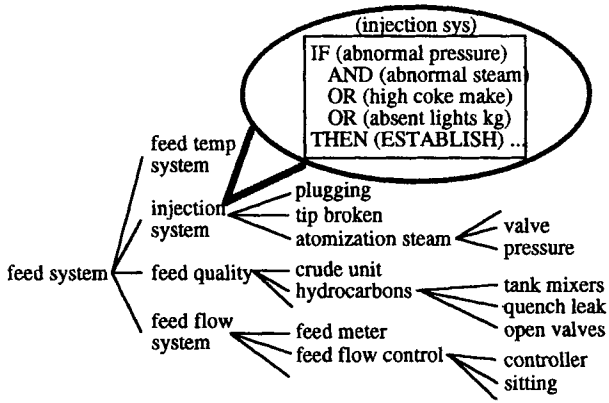


FIG. 33. Hierarchical classification. [Drawn from T. S. Ramesh, J. F. Davis, and G. J. Schwenzer, *Comput. Chem. Eng.* **16,** 109 (1992).]

the relation between blocks and the lower level models the details of each block.

As shown in Fig. 34, the approach involves breaking variables into blocks or groupings X_k and mapping the variables in each block into a set of block scores, r_k, and loadings, p_k. The block scores themselves are then aggregated, R, and become a variable block themselves from which a set of super scores, t, and loadings, w, can then be calculated. Although the algorithms for extracting the various features for hierarchical blocks have been developed and demonstrated, a systematic method of blocking variables is not yet available. It is clear that blocking based on knowledge of the process is necessary because it leads to logical groupings of variables. This observation suggests a future avenue of investigation of knowledge-based decomposition methods as described in preceding paragraphs. The strength of the KBS decompositions is that they consider the organization and structure of the process knowledge explicitly.

In general, there is no plug-in technology available for determining the best hierarchical decomposition of a plant. Prasad *et al.* (1998) have put forth some guidelines to assist in the development of taxonomies based on hierarchical classification. Given the vast variety of process plants, the design of an effective decomposition is not a trivial task. Moreover, the decomposition must be designed in concert with designing the localized interpreters. Yet achieving an effective modular design is crucial to the overall performance, ease of construction, and effective maintenance of a data interpretation system.

VIII. Comprehensive Examples

With reference to Fig. 4, this section presents several examples that demonstrate how various technologies can be assembled into data analysis and interpretation systems in practical application.

A. Comprehensive Example 1: Detection of Abnormal Situations

This example illustrates a system for monitoring and detection in a continuous polymer process (Piovoso *et al.*, 1992b). In the first stage of the process, several chemical reactions occur that produce a viscous polymer

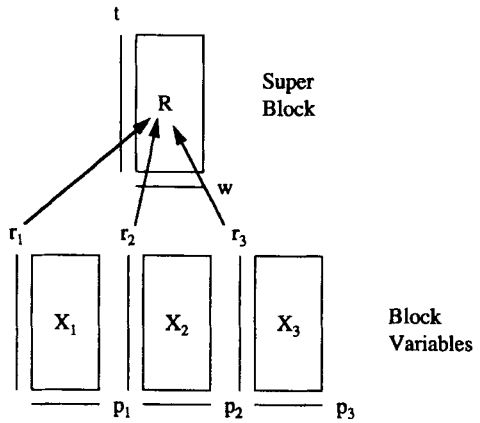


FIG. 34. A hierarchical multiblock principal component model.

product; in the second stage, the polymer is treated mechanically to prepare it for the third and final stage. Critical properties such as viscosity and density, if altered significantly, affect the final product, resulting in a loss of revenue and operability problems for the customer. Moreover, it is difficult to determine which of the three stages is responsible when the quality is not within acceptable limits. The situation is confounded by a lack of online sensors to measure continuously the critical properties. This makes it impossible to relate any specific changes to a particular stage. Indeed, property changes are detected by laboratory measurements that may have delays of 8 hours or more. The results of the analysis represent history; consequently, the current state of operations and the laboratory analysis of the process state are not aligned.

Such infrequent measurements make the control of the product quality difficult. At best, the operators have learned a set of heuristics that, if adhered to, usually produces a good product. However, unforeseen disturbances and undetected equipment degradations not accounted for by the heuristics still occur and affect the product. In addition, there are periods of operation when the final process step produces a degraded product in spite of near-perfect upstream operations.

Since stage 2 operates in support of stage 3 as well as to compensate for errors in the first stage, this example focuses on process monitoring and detection in stage 2. A simplified description of stage 2 is as follows. The product from the first stage is combined with a solvent in a series of mixers operated at carefully controlled speeds and temperatures. A sample of the mixture is taken at the exit of the final mixer for lab analysis. The material leaving the mixer enters a blender whose level and speed are maintained to impart certain properties to the final product. After exiting

the blender the material is then filtered to remove undissolved particulates before it is sent to the final stage of the process.

A significant amount of mechanical energy is necessary to move the highly viscous fluid through a complex transport network of pipes, pumps, and filters. Consequently, the time in service of the equipment is unpredictable. Even identical types of equipment placed in service at the same time may need replacement at widely differing times. Problems of incipient failure, pluggage, and unscheduled downtimes are an accepted, albeit undesirable, part of operation. To prolong continuous operations, pumps and filters are installed in pairs so that the load on one can be temporarily increased while the other is being serviced. Tight control of the process fluid is desirable because the equipment settings in the final stage of the process are preset to receive a uniform process fluid. As such, small deviations in the fluid properties may result in machine failure and nonsalable product.

Frequent maintenance on the equipment is not the only source of operating problems. Abrupt changes also occur due to the throughput demands in the final stage of the process. For example, if there is a decrease in demand due to downstream equipment failure, or a sudden increase due to the addition of new or reserviced equipment, the second stage must reduce or increase production as quickly as possible. It is more dramatic when throughput must be turned down because the process material properties change if not treated immediately, especially in the blender. These situations are frequent and unscheduled; thus, the second stage of this process moves around significantly and never quite reaches an equilibrium. Clearly, throughput is a dominant effect on the variability in the sensor values and process performance.

To address this situation, a data interpretation system was constructed to monitor and detect changes in the second stage that will significantly affect the product quality. It is here that critical properties are imparted to the process material. Intuitively, if the second stage can be monitored to anticipate shifts in normal process operation or to detect equipment failure, then corrective action can be taken to minimize these effects on the final product. One of the limitations of this approach is that disturbances that may affect the final product may not manifest themselves in the variables used to develop the reference model. The converse is also true—that disturbances in the monitored variables may not affect the final product. However, faced with few choices, the use of a reference model using the process data is a rational approach to monitor and to detect unusual process behavior, to improve process understanding, and to maintain continuous operation.

We can expect throughput to have a major effect on all the measure-

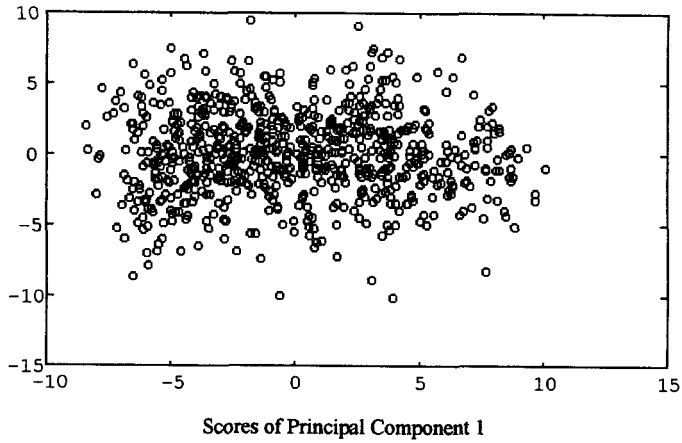


FIG. 35. Scores of Principal Component 1 vs scores of Principal Component 2; PCA model.

ments. A PCA analysis would, as a result, not only be overwhelmed by this single effect, but would obscure other information about the state of the process. An effective strategy then is to eliminate the throughput effect using a PLS analysis. Analysis of the residuals using PCA can then reveal other sources of variations critical to process operations.

To construct the reference model, the interpretation system required routine process data collected over a period of several months. Cross-validation was applied to detect and remove outliers. Only data corresponding to normal process operations (that is, when top-grade product is made) were used in the model development. As stated earlier, the system ultimately involved two analysis approaches, both reduced-order models that capture dominant directions of variability in the data. A PLS analysis using two loadings explained about 60% of the variance in the measurements. A subsequent PCA analysis on the residuals showed that five principal components explain 90% of the *residual* variability.

In this example, we focus on the PCA of the residuals. Figure 35 shows the observations of the residual data plotted in the score space of principal components 1 and 2. Although the observations are spread over a wide region of operating conditions and include a wide range of rate settings, Fig. 35 clearly shows how the data cluster when presented as principal components. The detection system exploits this cluster by identifying operations that fall within or outside of the cluster region based on a discriminant using the Mahalanobis distance as a measure of the goodness of fit (see Section V).

Figure 36 shows the performance of the interpretation system during a period of 30 hours. To detect an abnormal operation, the Mahalanobis

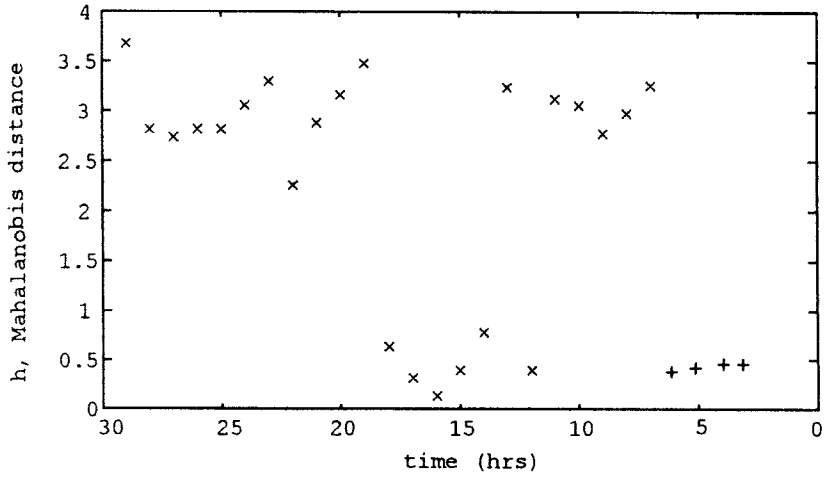


FIG. 36. Thirty hours of online operation. +, most recent 4 hours; ×, most distant history.

distance, h, expresses how close a given data pattern matches data used to form the principal component cluster in Fig. 35. The abscissa represents the most recent data to the left and the oldest data to the right. It was determined that a value greater than 1 implies that a given data pattern is not similar to the reference cluster. It is easy for operators to observe from figures such as this when the process is not performing well ($h > 1$) and to identify and correct the cause of the problem. In the first case (2–6 hours), a filter plugged and, in the second (12–18 hours), a pump failed. More details about this example can be found in Piovoso *et al.* (1992a).

B. COMPREHENSIVE EXAMPLE 2: DATA ANALYSIS OF BATCH OPERATION VARIABILITY

This example (Kosanovich *et al.*, 1995) builds on the previous example and illustrates how multivariate statistical techniques can be used in a variety of ways to understand and compare process behavior. The charge to the reactor is an aqueous solution that is first boiled in an evaporator until the water content is reduced to approximately 20% by weight. The evaporator's contents are then discharged into a reactor in which 10 to 20 pounds of polymer residue can be present from the processing of the previous batch.

This batch reactor is operated according to a combination of prespecified reactor and heat source pressure profiles and timed stages. The time to complete a batch is approximately 120 minutes. Key process checkpoints

(e.g., attaining a specific temperature within a given time) determine when one processing stage ends and the next one begins.

Heat is applied to the reactor to further concentrate the reactants and to supply the energy to activate the polymerization reactions. At the outset, the reactor temperature and pressure rise rapidly. Sensor measurements indicate the existence of a temperature gradient having as much as a 40°C difference between material at the top and at the bottom of the reactor. Shortly after the pressure reaches its setpoint, the entire mixture boils and the temperature gradient disappears. The solution is postulated to be well mixed at this time. The cumulative amount of water removed is one indication of the extent of polymerization.

The reactor pressure is reduced to 0 psig to flash off any remaining water after a desired temperature is reached. Simultaneous ramp up of the heat source to a new setpoint is also carried out. The duration spent at this second setpoint is monitored using CUSUM plots to ensure the batch reaches a desired final reactor temperature within the prescribed batch time. The heat source subsequently is removed and the material is allowed to continue reacting until the final desired temperature is reached. The last stage involves the removal of the finished polymer as evidenced by the rise in the reactor pressure. Each reactor is equipped with sensors that measure the relevant temperature, pressure, and the heat source variable values. These sensors are interfaced to a distributed control system that monitors and controls the processing steps.

In this example, data interpretations are based on Q-statistic limits. These are computed by assuming the data are normally distributed in the multivariate sense. The diagnostic limits are used to establish when a statistically significant shift has occurred. Charts based on these statistics and used in this manner are analogous to conventional SPC charts.

As in example 1, the explained variance (the total variance minus the residual variance) is calculated by comparing the true process data with estimates computed from a reference model. This explained variance can be computed as a function of the batch number, time, or variable number. A large explained variance indicates that the variability in the data is captured by the reference model and that correlations exist among the variables. The explained variance as a function of time can be very useful in differentiating among phenomena that occur in different stages of the process operations.

Process data for the same polymer recipe were analyzed for 50 nonconsecutive, sequential batches. As before, the data were preprocessed to remove outliers and sorted to reflect normal operation. The data are sampled at 1-minute intervals during production of each batch. Fillering and normalization of the data is done prior to analysis. The final polymer quality

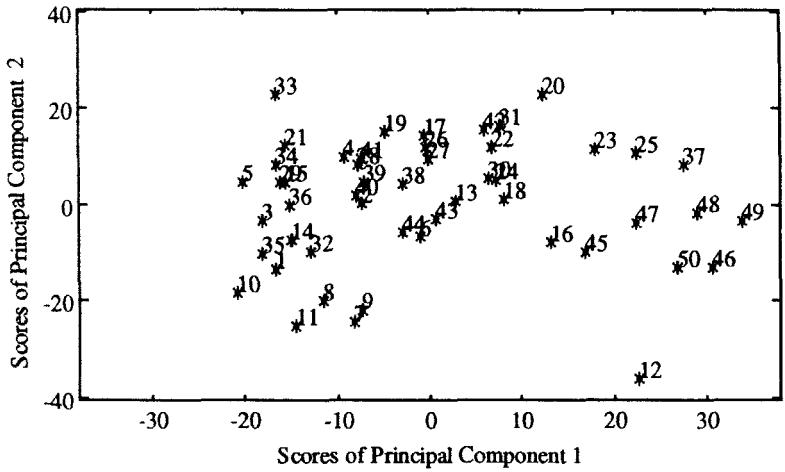


FIG. 37. Score plot of batch polymer reactor data (numbers indicate batch numbers).

data are obtained from laboratory measurements of polymer molecular weight and endgroups—one reading for each batch. As shown in Fig. 37, the PCA score plots for the reactor provide a summary of process performance from one batch to the next. All batches exhibiting similar time histories have scores that cluster in the same region of this particular principal component space.

Figure 38 shows the variance explained by the two principal component (PC) model as a percentage of each of the two indices: batch number and time. The lower set of bars in Fig. 38a are the explained variances for the first PC, while the upper set of bars reflects the additional contribution of the second PC. The lower line in Fig. 38b is the explained variance over time for the first PC and the upper line is the combination of PC 1 and 2. Figure 38a indicates, for example, that batch numbers 13 and 30 have very small explained variances, while batch numbers 12 and 33 have variances that are captured very well by the reference model after two PCs. It is impossible to conclude from this plot alone, however, that batches 13 and 30 are poorly represented by the reference model.

Additional insight is possible from Fig. 38b. Here we see that the magnitude of the explained variance accounted for by the second PC has noticeably increased after minute 70. This is consistent because, from process knowledge, it is known that removal of water is the primary event in the first part of the batch cycle, while polymerization dominates in the later part, explaining why the variance profile changes around the 70-minute point.

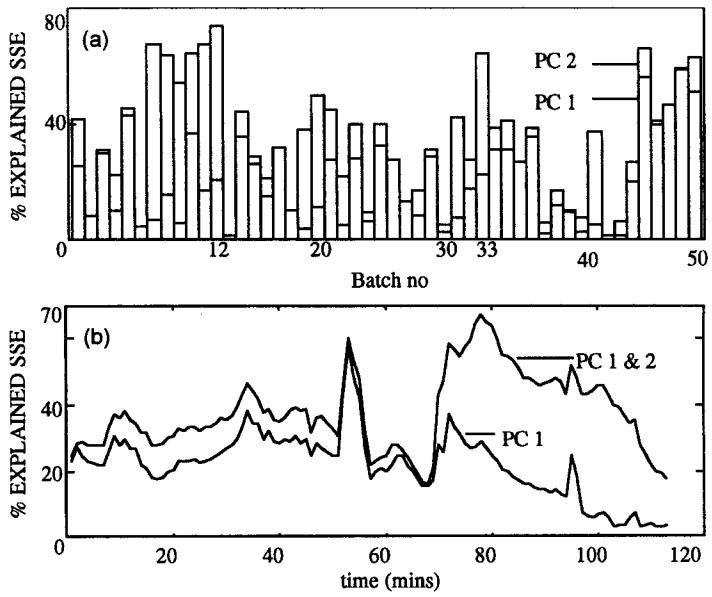


FIG. 38. Explained variance by batches (a) and over time (b) for batch polymer reactor data.

This interpretation of Fig. 38b leads to the conclusion that better insight into the variations can be obtained by separating the data into two time histories—the first covers time $k = 1$ to 53, and the second covers time $k = 54$ to 113 (all time units are in minutes). Incorporating this extended perspective, Fig. 39 provides score plots for two PC models where the score plots are now divided into these two main processing phases. Observe the

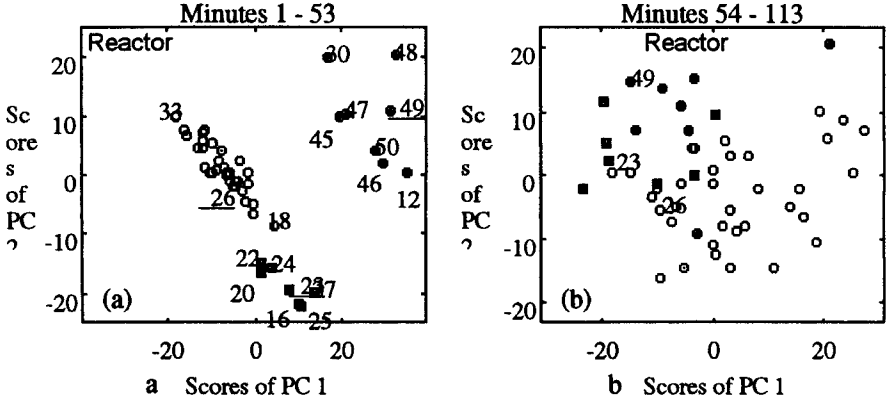


FIG. 39. Score plots for reactor: 1–53 minutes (a); 54–113 minutes (b).

presence of clusters in Fig. 39a for data from time $k = 1$ to 53, and the scatter in Fig. 39b for $k = 54$ to 113. A closer inspection of the operating conditions for these later batches indicates they were processed at a different heat transfer rate than the others. One possible conclusion is that the processing steps in the later stages are not influenced by the same factors that led to formation of clusters in the earlier stages. Conversely, it is possible that the processing steps in the later stage remove whatever led to the differentiation in the earlier stage.

Q-statistics can provide a multivariate way to view the batch processes. Computation of the Q-statistics as a function of batch number are shown in Fig. 40 for data taken from time $k = 1$ to 53. The Q-statistic for batches 30 and 37, for example, exceeds the 95% limit for *both* PCs, indicating that the reference model does not capture the variations in these batches. That is, the correlation structure of batches 30 and 37 are different than the majority of the batches. The Q-statistic as a function of time in Fig. 41 shows that deviations from the model subspace occur primarily in the first 35 minutes. Note that although the batches start out with a large Q-statistic, they end up within the 95% confidence limit.

The overall interpretation comes together when we recall that batches 13 and 30 had small explained variances. We note that the Q-statistic for batch 13 indicates that it is within the 95% limit for both PCs while the Q-statistic of batch 30 is not. The conclusion is that the variations in batch 13 are small random deviations about the average batch. In the case of batch 30, larger variations occur that are not well explained by the reference model. These variations are either large random fluctuations or variations that are orthogonal to the model subspace. Hence, the quality of batch 30, with a high probability, will not be within the specified limits.

C. Comprehensive Example 3: Diagnosis of Operating Problems in a Batch Polymer Reactor

This example illustrates how numeric–symbolic interpreters are combined with symbolic–symbolic interpretation for root cause diagnosis. It

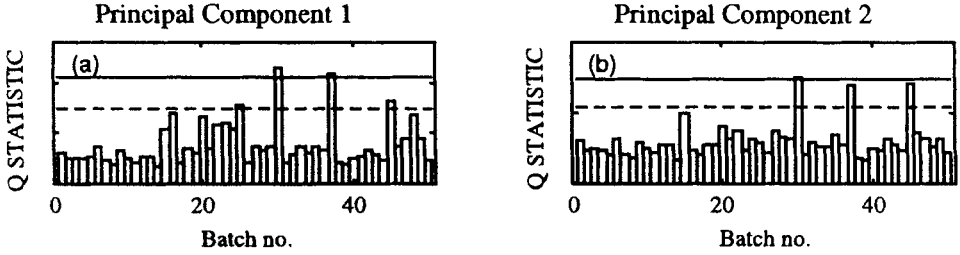


FIG. 40. Q-statistics by batch number for $k = 1$ to 53 minutes for reactor data.

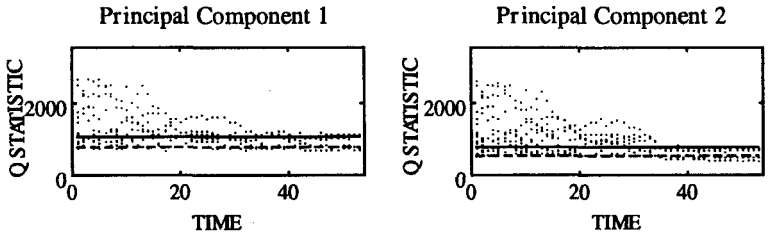


FIG. 41. Q-statistics over time for $k = 1$ to 53 minutes for reactor data.

also shows how hierarchical classification manages output dimensionality by decomposing the process to form a distributed set of data interpretation systems.

Consider again a batch polymerization process where the process is characterized by the sequential execution of a number of "steps" that take place in the two reactors. These are steps such as initial reactor charge, titration, reaction initiation, polymerization, and transfer. Because much of the critical product quality information is available only at the end of a batch cycle, the data interpretation system has been designed for diagnosis at the end of a cycle. At the end of a particular run, the data are analyzed and the identification of any problems is translated into corrective actions that are implemented for the next cycle. The interpretations of interest include root causes having to do with process problems (e.g., contamination or transfer problems), equipment malfunctions (e.g., valve problems or instrument failures), and step execution problems (e.g., titration too fast or too much catalyst added). The output dimension of the process is large with more than 300 possible root causes. Additional detail on the diagnostic system can be found in Sravana (1994).

The overall objective of the system is to map from three types of numeric input process data into, generally, one to three root causes out of the possible 300. The data available include numeric information from sensors, product-specific numeric information such as molecular weight and area under peak from gel permeation chromatography (GPC) analysis of the product, and additional information from the GPC in the form of variances in expected shapes of traces. The plant also uses univariate statistical methods for data analysis of numeric product information.

As is typical of process systems, the plant runs with a very high availability and product quality is normally maintained. For diagnostic data interpretation, there is very little data for developing any kind of numeric–symbolic interpreter that maps directly from the input data to the output diagnostic conclusions. It is possible, however, to map with a great deal of confidence from the input numeric data to a set of useful intermediate interpretations. With respect to the sensor data, there is considerable information for map-

ping between the input data and labels such as *high, low, increasing,* and *decreasing*. In this particular case study, all these interpretations are accomplished by limit checking.

In addition, the GPC trace, an example of which is shown in Fig. 42, reflects the composition signature of a given product and reflects the spectrum of molecular chains that are present. Analysis of the area, height, and location of each peak provides valuable quantitative information that is used as input to a CUSUM analysis. Numeric input data from the GPC is mapped into *high, normal,* and *low,* based on variance from established normal operating experience. Both the sensor and GPC interpretations are accomplished by individual numeric–symbolic interpreters using limit checking for each individual measurement.

In addition, operators can observe significant shape deviations in the GPC data such as *double peak* and *shoulder on low molecular weight side*. As shown in Fig. 42, the GPC trace can be subdivided into several regions. Each region is associated with an expected shape, e.g., Gaussian or flat. The features to be identified are variations in these expected shapes. These variations are complex in that they are characterized by skew to the right or left and extra distinct peaks. Figure 43 shows the expected shape in the Gaussian region and several examples of shape variation.

Development of a GPC shape interpreter is characterized by a large number of training examples of normal shapes and a few training examples of each of the abnormal shapes. Because of the existence, but limited availability, of abnormal examples, Adaptive Resonance Theory (see Sec-

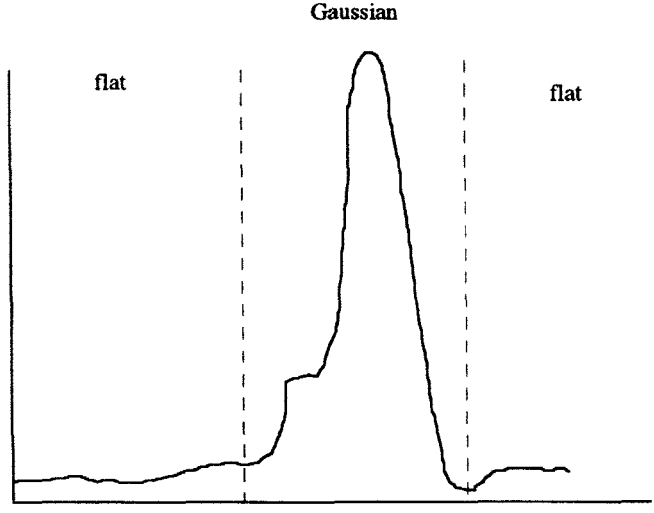


FIG. 42. A GPC trace.

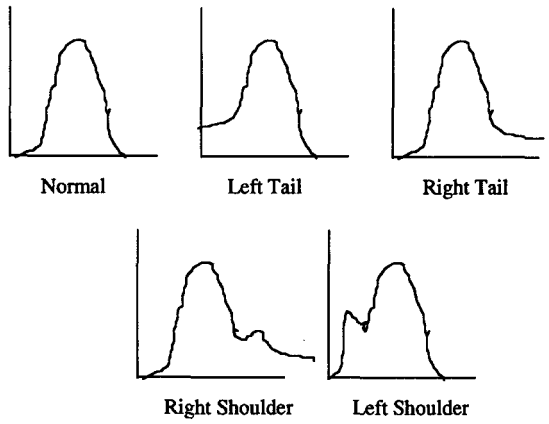


FIG. 43. Examples of Gaussian region shape deviations.

tion V) was selected for constructing a numeric–symbolic interpreter. Inputs to the interpreter were the numeric points that formed the GPC trace. The output of the interpreter were the labels *normal* and the various abnormal shapes. Figure 44 illustrates GPC pattern recognition using ART. Label assignment is determined by whether x falls or does not fall within the boundaries of defined fault classes that group patterns of data with similar characteristics. If an arbitrary pattern falls within a class boundary, then it is labeled that fault class with high certainty. If it falls outside, but near, a class boundary, then it is labeled the nearest fault class, but with medium certainty. The resulting interpretations in the form of labels such as *Gaussian Region—Left Tail—medium certainty* are used along with sensor interpretations as intermediate label inputs to a symbolic–symbolic interpreter.

Given the intermediate labels from the GPC pattern and sensor interpreters, a KBS is used to map them into the diagnostic labels of interest. However, because the output dimension of the problem is large (at over 300 possible malfunctions), a hierarchical classification system is used to decompose the process into a hierarchical set of malfunction categories that

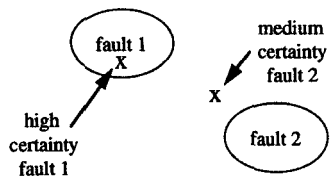


FIG. 44. Data interpretation in ART.

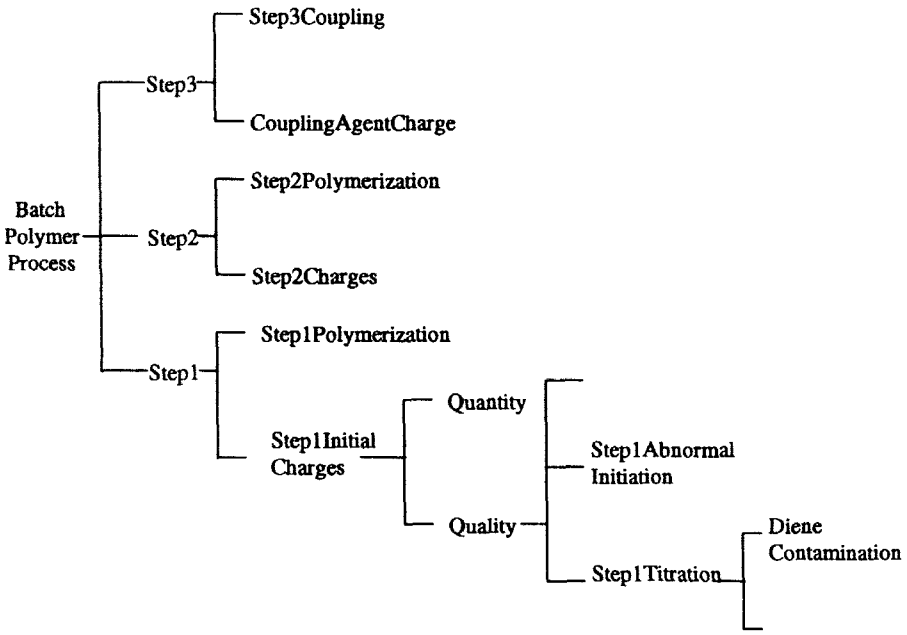


FIG. 45. Top-level organization of malfunction hypotheses for an existing batch polymer reactor with a focus on Step 1: Initial Charge problems.

provide a mechanism for generating and testing malfunction hypotheses in a distributed and localized manner.

As shown in Figure 45, the polymerization process can be organized at the top level according to major procedural steps, which in this case are called *Step1, Step2,* and *Step3* rections. At this level, problems with the process are associated with completion of the intended functions. Considering completion in more detail leads to consideration of problems with either the initial charges or the polymerization reaction itself. For example, in the case of Step1: Initial Charges, the next level of consideration is either the quantity or the quality of charge, and so forth. This decomposition continues as illustrated for diene contamination in Fig. 46 to tip nodes that deal with problems in specific equipment items, operator errors, and so forth.

Details of the hierarchical decomposition used to manage the scope of the problem are given in Sravana (1994). There are many considerations in developing a particular decomposition, but the objective is to decompose the process into smaller, more localized malfunction considerations.

The decomposition and localization of the problem effectively produces a distributed set of data interpretation problems, each with a particular set

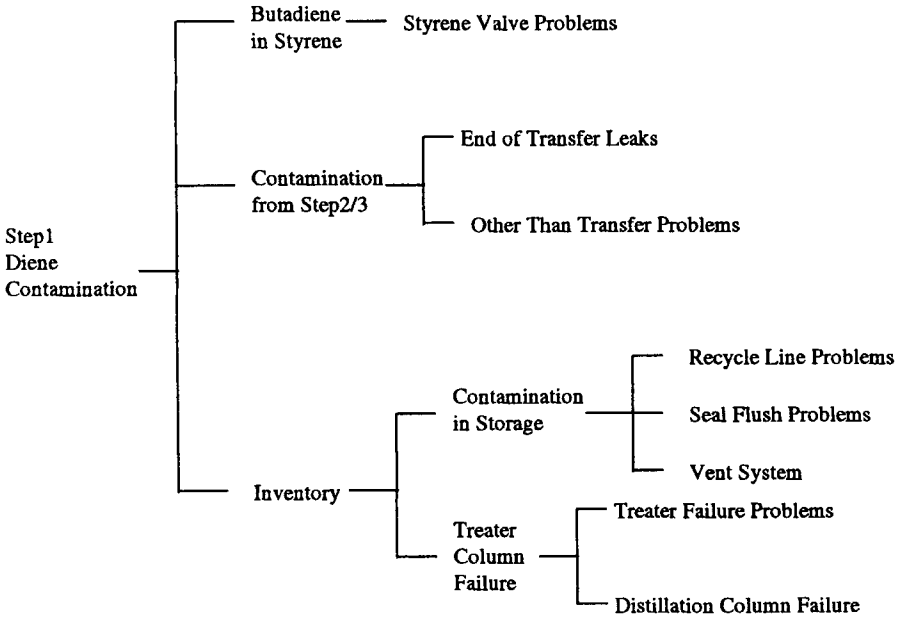


FIG. 46. Diene contamination branch.

of possible output conclusions and a set of input variables that provides information for the interpretation.

Each of the nodes represents an individual symbolic–symbolic interpretation problem of determining whether there is a malfunction associated with that particular element of the process. As illustrated in Fig. 47, the node Step1: Diene Contamination represents an interpretation problem in which there are two possible output conclusions—*yes* or *no*. There are three input variables, the values of which (intermediate interpretations) are determined by individual numeric–symbolic interpreters using process data, GPC data, and a GPC shape interpretation. At each node, mapping between the input variables and the output conclusions is accomplished by simple table lookup. To reach a detailed diagnostic conclusion, the objective is to pursue the diagnostic branches in greater and greater detail as diagnostic hypotheses indicated by the nodes are accepted or rejected using at each a local set of numeric–symbolic and symbolic–symbolic interpreters.

In this way, data interpretation is accomplished by a set of nested numeric–symbolic and symbolic–symbolic interpreters. Note that the hierarchical decomposition results in a distributed set of symbolic–symbolic interpretation problems represented by nodes. Each problem requires intermediate interpretations of numeric data as input to the symbolic–

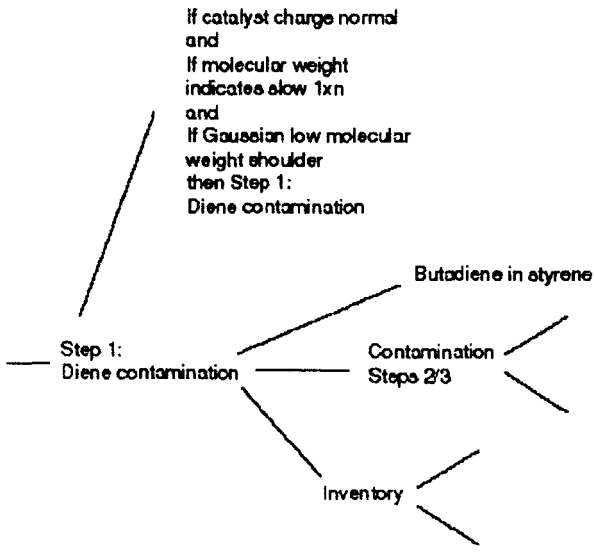


FIG. 47. Local diene contamination subproblem.

symbolic problem. The intermediate interpretations use individual numeric–symbolic interpreters tailored for the specific interpretation of interest.

IX. Summary

The widespread interest and success in Statistical Process Control over the past 15 years has introduced the value and importance of data analysis and interpretation in process operations. In response to much more stringent demands on production, quality, and product flexibility, multivariate statistical and intelligent system technologies and a multitude of associated integrated technologies have produced a new wave of more powerful approaches that are proving their economic worth. These approaches are not only providing considerably improved ways of analyzing data, but are also providing the mechanisms for interpretation that enhance how the essential human element can interact better with the process operation.

Figure 4 is a defining view of data analysis and interpretation illustrating four major families of technologies that must be considered for any given interpretation of numeric process data. With this view, the construction of any single data interpretation system is a complicated integration of technologies in itself. This is especially brought out in Fig. 2 and 5, in which

the scope of empirical analysis and interpretation techniques, respectively, become clear. However, when we further overlay this view of a single interpretation system with the fact that the number of interpretations possible is highly constrained by performance and that a complex process operation is likely to require many of these interpretation systems used locally, the problem is daunting. Nevertheless, progress on individual technologies and the relatively recent emphasis on integration have brought these complex analysis and interpretation systems into reality. Focused applications of analysis and interpretation systems for specifically defined process situations are coming into wide use at the present time. Comprehensive systems for entire process units are beginning to be considered.

The success of this chapter lies not in detailed listings, descriptions, or comparisons of individual approaches, nor in an exhaustive review of the literature, but in a perspective for integrating a very wide-ranging array of technologies and managing complexity in comprehensive analysis and interpretation systems. This integrated perspective is, in fact, the product of a considerable struggle by the four authors, who are representative of these widely varying technology perspectives. To this end, the chapter has taken a unique look at alternatives to quantitative behavioral model approaches from the point of view of interpretation. The emphasis on how to manipulate knowledge and data to clarify process behaviors and to assign interpretations based on experiences with the operation offers a practical view for linking the diverse technology areas.

Acknowledgments

The chapter is a compilation of a great deal of work in this area by the four authors and a large number of graduate students who have contributed over a period of more than 15 years, as well as the many contributions of a large technical community. We recognize and gratefully acknowledge these contributions, without which this chapter would not have been possible. We also specifically thank David Miller, a postdoctoral student at The Ohio State University, for his detailed review, and Lynda Mackey for her editorial assistance.

REFERENCES

Ali, Z., Ramachandran, S., Davis, J. F., and Bakshi, B., On-line state estimation in intelligent diagnostic decision support systems for large scale process operations, *AIChE Mtg.*, Los Angeles, CA, November (1997).

Antsaklis, P. J., and Passino, K. M., Eds., "An Introduction to Intelligent and Autonomous Control." Kluwer Academic Publishers, 1993.

Badiru, A. B., "Expert Systems Applications in Engineering and Manufacturing," Prentice Hall, 1992.

Bakshi, B. R., and Stephanopoulos, G., Wave-Net: a multi-resolution, hierarchical neural network with localized learning, *AIChE J.* **39**(1), 57–81 (1993).

Bakshi, B. R., and Utojo, U., Unification of neural and statistical methods that combine inputs by linear projection, *Comput. Chem. Eng.* **22**(12), 1859–1878 (1998).

Ball, G. H., and Hall, D. J., "Isodata, a novel method of data analysis and pattern classification," NTIS Report AD699616 (1965).

Basseville, M., and Benveniste, A., "Detection of Abrupt Changes in Signals and Dynamical Systems. Springer-Verlag, New York, 1986.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., "Classification and Regression Tres." Wadsworth, 1984.

Carpenter, G. A., and Grossberg, S., ART2: self-organization of stable category recognition codes for analog input patterns, *Appl. Opt.* **26,** 4919 (1987a).

Carpenter, G. A., and Grossberg, S., A massively parallel architecture for a self-organizing neural pattern recognition machine, *Comput. Vis. Graphics Image Process* **37,** 54 (1987b).

Carpenter, G. A., and Grossberg, S., The ART of adaptive pattern recognition by a self-organizing neural network, *Computer,* **21,** 77 (1988).

Carpenter, G. A., Grossberg, S., and Rosen, D. B., ART2-A: An adaptive resonance algorithm for rapid category learning and recognition, *Neural Network* **4,** 493 (1990).

Catino, C. A., and Ungar, L. H., Model-based approach to automated hazard identification of chemical plants, *AIChE J.* **41**(1), 97–109 (1995).

Chandrasekaran, B., Models versus rules, deep versus compiled, content versus form: Some distinctions in knowledge systems research, *IEEE Expert* **6**(2), 75–79 (1991).

Chen, L. W., and Modarres, M., Hierarchical decision process for fault administration, *Comput. Chem. Eng.* **16**(5), 425–448 (1992).

Chen, S., Cowan, C. F. N., and Grant, P. M., Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. Neur. Net.* **2**(2), 302–309 (1991).

Coifman, R. R., and Wickerhauser, M. V., Entropy-based algorithms for best basis selection, *IEEE Trans. Inform. Theory* **38**(2), 713–718 (1992).

Cybenko, G., Continuous valued neural networks with two hidden layer are sufficient, Technical Report, Department of Computer Science, Tufts University (1988).

Daubechies, I., Orthonormal bases of compactly supported wavelets, *Comm. Pure Applied Math.* **XLI,** 909–996 (1988).

Davis, J. F., On-line knowledge-based systems in process operations: The critical importance of structure for integration, *Proc. IFAC Symp. on Advanced Control of Chemical Processes,* ADCHEM '94, Kyoto, Japan (1994).

Davis, J. F., and Wang, C. M., Pattern-based interpretation of on-line process data, *in* "Neural Networks for Chemical Engineers" (A. Bulsari, ed.), Elsevier Science, 1995, pp. 443–470.

Davis, J. F., Bakshi, B., Kosanovich, K. A., and Piovoso, M. I., Process monitoring, data analysis and data interpretation, "Proceedings, Intelligent Systems in Process Engineering," AIChE Symposium Series, **92**(312), (1996a).

Davis, J. F., Stephanopoulos, G., and Venkatasubramanian, V., eds., "Proceedings, intelligent systems in process engineering, 1995," AIChE Symposium Series, 92 (312), (1996b).

De Veaux, R. D., Psichogios, D.C., and Ungar, L. H., A comparison of two nonparametric estimation schemes: MARS and neural networks, *Comput. Chem. Eng.* **17**(8), 819–837 (1993).

Dong, D., and McAvoy, T. J., Nonlinear principal component analysis—based on principal curves and neural networks, *Comput. Chem. Eng.* **20**(1), 65 (1996).

Donoho, D. L., Johnstone, I. M., Kerkyacharian, G., and Picard, D., Wavelet shrinkage: asymptopia? *J. R. Stat. Soc. B.* **57**(2), 301 (1995).

Duda, R. O., and Hart, P. E., "Pattern Classification and Scene Analysis." Wiley-Interscience, New York, 1973.

Finch, F. E., and Kramer, M. A., Narrowing diagnostic focus using functional decomposition, *AIChE J.* **34**(1), 25–36 (1988).

Frank, I. E, A nonlinear PLS model, *Chemom. Intell. Lab. Sys.* **8,** 109–119 (1990).

Friedman, J. H., Multivariate adaptive regression splines, *Ann. Statistics* **19**(1), 1–141 (1991).

Friedman, J. H., and Stuetzle, W., Projection pursuit regression, *J. Amer. Stat. Assoc.* **76,** 817–823 (1981).

Gallagher, N. C., Jr., and Wise, G. L., A theoretical analysis of the properties of median filters, *IEEE Trans. Acoustics, Speech, & Signal Proc.* **29,** 1136 (1981).

Geladi, P., and Kowalski, B., Partial least-squares regression: a tutorial, *Anal. Chim. Acta* **185,** 1–17 (1986).

Grossberg, S., "Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control," Reidel, Boston, 1982.

Grossberg, S., "The Adaptive Brain, I: Cognition, Learning, Reinforcement and Rhythm." Elsevier/North Holland, Amsterdam, 1987a.

Grossberg, S., "The Adaptive Brain, II: Vision, Speech, Language and Motor Control." Elsevier/North Holland, Amsterdam, 1987b.

Grossberg, S., "Neural Networks and Natural Intelligence," MIT Press, Cambridge, MA, 1988.

Hastie, T. J., and Tibshirani, R. J., "Generalized Additive Models." Chapman and Hall, New York, 1990.

Heinonen, P., and Neuvo, Y., FIR-median hybrid filters, *IEEE Trans. Acoustics and Signal Processing* **35**(6), 832 (1987).

Henrion, M., Breese, J. S., and Horvitz, E. J., Decision analysis and expert systems, *AI Mag.* **12**(4), 64–91 (1991).

Hornik, K., Stinchcombe, M., and White, H., Multilayer feedforward networks are universal approximators, *Neural Networks* **2**(5), 359–36 (1989).

Hoskins, J. C., and Himmelblau, D. M., Artificial neural network models of knowledge representation in chemical engineering, *Comput. Chem. Eng.* **12,** 881–890 (1988).

Hoskins, J. C., Kaliyur, K. M., and Himmelblau, D. M., Fault diagnosis in complex chemical plants using artificial neural networks, *AIChE J.* **37,** 137–141 (1991).

Hwang, J. N., Lay, M., Martin, R. D., and Schimert, J., Regression modeling in back-propagation and projection pursuit learning, *IEEE Trans. Neur. Networks* 5 (1994).

Iserman, R., Process fault detection based on modeling and estimation methods—a survey, *Automatica* **20,** 387–404 (1984).

Jain, A. K., and Dubes, R. C., "Algorithms for Clustering Data." Prentice-Hall, Englewood Cliffs, NJ, 1988.

Jackson, J. E., "A User's Guide to Principal Components." Wiley, New York, 1992.

Jolliffe, I. T., "Principal Component Analysis." Springer-Verlag, New York, 1986.

Josephson, J., and Josephson, S., eds., "Abductive Inference: Computation, Philosophy, Technology." Cambridge University Press, 1994.

Kavuri, S. N., and Venkatasubramanian, V., Using fuzzy clustering with ellipsoidal units in neural networks for robust fault classification, *Comput. Chem. Eng.* **17**(8), 765 (1993).

Kelly, B. E., and Lees, F. P., The propagation of faults in process plants: 1, modeling of fault propagation, *Reliability Eng. and Sys. Safety* **16,** 3 (1986).

Kohonen, T., "Self-Organizing and Associative Memory." Springer-Verlag, New York, 1989.

Kosanovich, K. A., Piovoso, M. J., and Dahl, K. S., Multi-way principal component analysis applied to an industrial batch process, *Ind. & Eng. Chem. Res.* **6,** 739 (1995).

Kramer, M. A., Malfunction diagnosis using quantitative models with non-Boolean reasoning in expert systems, *AIChE J.* **33,** 1130–140 (1987).

Kramer, M. A., Nonlinear principal component analysis using autoassociative neural networks, *AIChE J.* **37**(2), 233–243 (1991).

Kramer, M. A., and Mah, R. S. H., Model-based monitoring, *in* "Proc. Second Int. Conf. on Foundations of Computer Aided Process Operations." (D. Rippin, J. Hale, and J. Davis, eds.). CACHE, 1994.

Kramer, M. A., and Palowitch, B. L., Rule-based approach to fault diagnosis using the signed directed graph, *AIChE J.* **33**(7), 1067–1078 (1987).

Kresta, J. V., and MacGregor, J. F., Multivariate statistical monitoring of process operating performance, *Can. J. Chem. Eng.* **69,** 35–47 (1991).

Kuipers, B., "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge." MIT Press, Boston, 1994.

Leonard, J., and Kramer, M. A., Improvement of the back propagation algorithm for training neural networks, *Comput. Chem. Eng.* **14,** 337–341 (1990).

Leonard, J., and Kramer, M. A., Radial basis function networks for classifying process faults, *IEEE Control Systems,* April, p. 31 (1991).

MacGregor, J. F., and Kourti, T., Statistical process control of multivariate processes, *Cont. Eng. Prac.* **3,** 404–414 (1995).

MacGregor, J. F., Jaeckle, C., Kiparissides, C., and Koutoudi, M., Process monitoring and diagnosis by multiblock PLS method, *AIChE J.* **40**(5), 826–838 (1994).

MacQueen, J., Some methods for classification and analysis of multivariate data, *Proc. 5th Berkeley Symp. on Probability and Statistics,* Berkeley, CA (1967).

Mallat, S. G., A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7), 674–693 (1989).

Maren, A., Harston, C., and Rap, R., "Handbook of Neural Computing Applications." Academic Press, New York, (1990).

Mavrovouniotis, M. L., ed., "Artificial Intelligence in Process Engineering." Academic Press, Boston, 1990.

McDowell, J. K., and Davis, J. F., Diagnostically focused simulation: managing qualitative simulation, *AIChE J.* **37**(4), 569–580 (1991).

McDowell, J. K., and Davis, J. F., Problem solver integration based on generic task architectures, *in* "Intelligent Modeling, Diagnosis and Control of Manufacturing Processes" (B. B. Chu and S. Chen, eds.). World Scientific, Singapore, 1992, pp. 61–82.

McVey, S. R., Davis, J. F., and Venkatasubramanian, V., Intelligent systems in process operations, design and safety, *in* "A Perspective on Computers in Chemical Engineering." CACHE Corp., 1997.

Miller, W. T., III, Sutton, R. S., and Werbos, P. J., eds., "Neural Networks for Control." MIT Press, Boston, 1990.

Minsky, M., and Papert, S., "Perceptions: An Introduction to Computational Geometry." MIT Press, Cambridge, MA, 1969.

Mitra, S. K., and Kaiser, J. F., "Digital Signal Processing." Wiley, New York, 1993.

Mohindra, S., and Clark, P. A., A distributed fault diagnosis method based on digraph models: Steady-state analysis, *Comput. Chem. Eng.* **17,** 193 (1993).

Moody, J., "Fast learning in multi-resolution hierarchies," Research Report, Yale University, YALEU/DCS/RR-681 (1989).

Moody, J., and Darken, C. J., Fast learning in networks of locally-tuned processing units, *Neural Computation* **1,** 281–294 (1989).

Mullhi, J. S., Ang, M. L., and Lees, F. P., *Reliability Engr. and Sys. Safety* **23,** 31 (1988).

Myers, D. R., Hurley, C. E. and Davis, J. F. A diagnostic expert system for a sequential, PLC controlled operation, *in* "Artificial Intelligence Applications in Process Engineering." (M. Mavrovoumolis, (ed.). Academic Press, New York, 1990.

Naidu, S., Zafiriou, E., and McAvoy, T. J., Application of neural networks on the detection of sensor failure during the operation of a control system, *Proc. Amer. Control Conf.,* Pittsburgh, PA (1989).

Nason, G. P., Wavelet shrinkage using cross-validation, *J. R. Statist. Soc. B* **58**(2), 463 (1996).

Nomikos, P., and MacGregor, J. F., Monitoring of batch processes using multi-way PCA, *AIChE J.* **40**(5), 1361–1375 (1994).

Nounou, M. N., and Bakshi, B. R., On-line multiscale fillering of random and gross errors without process models, *AIChE J.,* **45**(6), 1041 (1999).

Oja, E., Unsupervised neural learning, *in* "Neural Networks for Chemical Engineers" (A. B. Bulsari, (ed.). Elsevier, Amsterdam, 1995, p. 21.

Pearl, J., "Probabilistic Reasoning in Intelligent Systems," Morgan Kaufmann, San Mateo, CA, 1988.

Piovoso, M. J., and Kosanovich, K. A., Applications of multivariate statistical methods to process monitoring and controller design, *Int. J. Control* **59**(3), 743–765 (1994).

Piovoso, M. J., Kosanovich, K. A., and Yuk, J. P., Process data chemometrics, *IEEE Trans. Instrumentation and Measurements* **41**(2), 262–268 (1992a).

Piovoso, M. J., Kosanovich, K. A., and Pearson, R. K., Monitoring process performance in real-time, *Proc. Amer. Control Conf.,* Chicago, IL, Vol. 3, pp. 2359–2363 (1992b).

Prasad, P. R., and Davis, J. F., A framework for knowledge-based diagnosis in process operations, *in* "An Introduction to Intelligent and Autonomous Control" (P. J. Antsaklis and K. M. Passino, eds.). Kluwer Academic Publishers, Boston, 1992, pp. 401–422.

Prasad, P. R., Davis, J. F., Jirapinyo, Y., Bhalodia, M., and Josephson, J. R., Structuring diagnostic knowledge for large-scale process systems, *Comput. Chem. Eng.,* **22,** 1897–1905 (1998).

Qin, S. J., and McAvoy, T. J., Nonlinear PLS modeling using neural networks, *Comput. Chem. Eng.* **16**(4), 379 (1992).

Quantrille, T. E., and Liu, Y. A., "Artificial Intelligence in Chemical Engineering." Academic Press, San Diego, 1991.

Ragheb, M., and Gvillo, D., Development of model-based fault identification systems on microcomputers, *International Society for Optical Engineering,* pp. 268–275. SPIE, Bellingham, WA, 1993.

Ramesh, T. S., Shum, S. K., and Davis, J. F., A structured framework for efficient problem solving in diagnostic expert systems, *Comput. Chem. Eng.* **12,** 891 (1988).

Ramesh, T. S., Davis, J. F., and Schwenzer, G. M., "Knowledge-based diagnostic systems for continuous process operations based upon the task framework," *Comput. Chem. Eng.* **16**(2), 109–127 (1992).

Rekalitis, G. V., and Spriggs, H. D., *Proceedings of the First International Conference on Foundations of Computer-Aided Operations,* Elsevier Science Publishers, New York, 1987.

Rich, E., and Knight, K., *Artificial Intelligence, 2nd edition,* McGraw-Hill, New York, 1991.

Ripley, B. D., "Neural networks and related methods for classification," *J. Royal Stat. Soc.* **56**(3), 409–456 (1994).

Rippin, D., Hale, J., and Davis, J. F., eds., *Proceedings Foundations of Computer-Aided Process Operations,* CACHE, 1994.

Roosen, C. B., and Hastie, T. J., Automatic smoothing spline projection pursuit. *J. Comput. Graph. Stat.,* **3,** 235 (1994).

Rosenblatt, F., *Principles of Neurodynamics: Perceptions and the Theory of Brain Mechanisms,*" Spartan, NY, 1962.

Roth, E. M., Woods, D. D., and Pople, H. E., "Cognitive simulation as a tool for cognitive task analysis," *Ergonomics* **35**(10), 1163–1198 (1992).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representation by error propagation," *in* "Parallel Distributed Processing, Vol. 1., Foundations" (D. E. Rumelhart and J. L. McClelland, eds.). MIT Press 1986.

Shapiro, S. C., Eckroth, D., et al., eds., "*Encyclopedia of Artificial Intelligence.*" John Wiley & Sons, New York, 1987.

Simon, J. C., "*Patterns and Operators: The Foundations of Data Representation.*" North Oxford Academic Publishers Ltd., London, 1986.

Sravana, K. K., "Diagnostic Knowledge-Based systems for batch chemical processes: hypothesis queuing and evaluation," PhD Thesis, The Ohio State University (1994).

Stephanopoulos, G., ed., "*Knowledge-based systems in process engineering.*" CACHE Case Studies Series-I, CATDEX, CACHE Corp., Austin, TX, 1988.

Stephanopoulos, G., and Davis, J. F., eds., CACHE monograph series, *in* "Artificial Intelligence in Process Systems Engineering." Vols. I, II and III, CACHE (1990).

Stephanopoulos, G., and Davis, J. F., eds., CACHE monograph series, *in* "Artificial Intelligence in Process Systems Engineering." Vol. IV, CACHE (1993).

Stephanopoulos, G., and Han, C., Intelligent systems in process engineering. *Proc. PSE 94,* Korea (1994).

Strang, G., Wavelets and dilation equations: a brief introduction. *SIAM Review,* **31**(4), 614–627 (1989).

Tan, S., and Mavrovouniotis, M. L., Reducing data dimensionality through optimizing neural networks inputs, *AIChE J.* **41**(6), 1471–1480 (1995).

Tou, J. T., and Gonzalez, R. C., "*Pattern Recognition Principles.*" Addison-Wesley, Reading, MA, 1974.

Ulerich, N. H., and Powers, G. J., On-Line hazard aversion and fault detection: Multiple loop control example. *AIChE Fall National Meeting,* New York, 1987.

Ungar, L. H., Powell, B. A., and Kamens, S. N., Adaptive networks for fault diagnosis and process control, *Comput. Chem. Eng.* **14,** 561–572 (1990).

Vedam, H., Venkatasubramanian, V., and Bhalodia, M., A B-spline base method for data compression, process monitoring and diagnosis. *Comput. Chem. Eng.* **22**(13), S827–S830 (1998).

Venkatasubramanian, V., and Rich, S., An object-oriented two-tier architecture for integrating compiled and deep level knowledge for process diagnosis. *Comput. Chem. Eng.* **12**(9), 903 (1988).

Venkatasubramanian, V., and Vaidhyanathan, R., A knowledge-based framework for automating HAZOP analysis, *AIChE J.* **40**(3), 496 (1994).

Whiteley, J. R., and Davis, J. F., Qualitative interpretation of sensor patterns. *IEEE Expert,* **8**(2) 54–63 (1992a).

Whiteley, J. R., and Davis, J. F., Knowledge-based interpretation of sensor patterns, special issue of *Comput. Chem. Eng.* **16**(4), 329–346 (1992b).

Whiteley, J. R., and Davis, J. F., A similarity-based approach to interpretation of sensor data using adaptive resonance theory. *Comput. Chem. Eng.* **18**(7), 637–661 (1994).

Whiteley, J. R., Davis, J. F., Ahmet, M., and Ahalt, S. C., Application of adaptive Resonance theory for qualitative interpretation of sensor data, *Man, Machine and Cybernetics* **26**(7), July (1996).

Widrow, B., and Hoff, M. E., Adaptive switching circuits, *1960 WESCON Convention Record, Part IV,* 96–104 (1960).

Wilcox, N. A., and Himmelblau, D. M., The possible cause and effect graphs (PCEG) model for fault diagnosis—I methodology, *Comput. Chem. Eng.* **18**(2) 103–116 (1994).

Wold, H., "Soft Modeling: The Basic Design and Some Extensions, Systems Under Indirect Observations" (K. J. Horeskog and H. Wold, eds.). Elsevier Science, North Holland, Amsterdam, 1982.

Wold, S., Ruhe, A., Wold, H., and Dunn, W., The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses, *SIAM J. Sci. and Statistical Computing* **5,** 735–743 (1984).

Wold, S., Esbensen, K., and Geladi, P., Principal component analysis, *Chemometrics and Intelligent Laboratory Systems* **2,** 37–52 (1987a).

Wold, S., Geladi, P., and Ohman, J., Multi-way principal components and PLS analysis, *J. Chemometrics* **1,** 41–56 (1987b).

Wold, S., Kettaneh-Wold N., and Skagerberg, B., Nonlinear PLS modeling, *Chemom. Intell. Lab. Sys.* **7,** 53–65 (1989).

Wold, S., Nonlinear PLS modeling II: spline inner relation, *Chemom. Intell. Lab. Sys.,* **14,** 71–84 (1992).

Wold, S., Kettaneh, N., and Tjessem, K., "Hierarchical multiblock PLS and PC models for easier model interpretation and as an alternative to variable selection," *J. Chemometrics* **10,** 463 (1996).