Chapter 7 LINEAR PRINCIPAL COMPONENT ANALYSIS

7.1. Introduction to linear principal component analysis (LPCA)

Principal component analysis is among the most popular methods for extracting information from data, which has been applied in a wide range of disciplines. In chemical process operation and control, PCA is used to solve several tasks including rectification (Kramer and Mah, 1994), gross-error detection (Tong and Crowe, 1995), disturbance detection and isolation (Ku et al., 1995), statistical process monitoring (Kresta et al., 1991; Wise et al., 1990), and fault diagnosis (MacGregor et al., 1994; Dunia et al., 1996). PCA is popular for process monitoring since it allows extension of the principles of univariate statistical process monitoring (SPM) to monitoring of multivariate processes (Jackson, 1980; Kresta et al., 1991).

Conventional PCA is best for analyzing a two-dimensional matrix of data collected from a steady-state process, containing linear relationships between the variables. Since these conditions are often not satisfied in practice, several extensions of PCA have been developed. Multiway PCA allows the analysis of a multidimensional matrix (Nomikos and MacGregor, 1994). Hierarchical or multiblock PCA permits easier modeling and interpretation of a large matrix by decomposing it into smaller matrices or blocks (Wold et al., 1996; MacGregor et al., 1994). Dynamic PCA extracts time-dependent relationships in the measurements by augmenting the data matrix by time-lagged variables (Kresta et al., 1991; Ku et al., 1995). Nonlinear PCA (Kramer, 1991; Hastie and Stuetzle, 1989; Dong and McAvoy, 1996; Tan and Mavrovouniotis, 1995) extends PCA to extracting nonlinear relationships between the variables. On-line adaptive PCA updates the model parameters continuously by exponential smoothing (Wold, 1994).

The superficial dimensionality of data, or the number of individual observations constituting one measurement vector, is often much greater than the intrinsic dimensionality, the number of independent variables underlying the significant nonrandom variations in the observations (Kramer, 1991). The reduction of the data set from its superficial to intrinsic dimensions is the focus of principal component analysis.

Due to correlation, a few principal components are usually sufficient to capture the data variance (Dunia, 1999). Two kinds of abnormal conditions can be distinguished using PCA. These are:

- failure of sensor correlations: In this situation the PCA model is no longer valid and the Euclidean norm of the residual vector increases significantly.
- Excessive normal variance: The variables used to define the operating variability are out of the normal range, as suggested by the historical data.

7.2. Introduction to Multiscale PCA (MSPCA)

Modeling by PCA and its extensions is done at a single scale, that is, the model relates data represented on basis functions with the same time-frequency localization at all locations. For example, PCA of a time series of measurements is a single-scale model since it relates variables only at the scale of the sampling interval. Such a single-scale modeling approach is only appropriate if the data contains contributions at just one scale. Unfortunately, data from almost all practical processes are multiscale in nature due to:

- Events occurring at different locations and with different localization in time and frequency.
- Stochastic processes whose energy or power spectrum changes with time and/or frequency.
- Variables measured at different sampling rates or containing missing data

Consequently, conventional PCA is not ideally suited for modeling of most process data. Techniques have been developed for PCA of some types of multiscale data such as missing data, but the single-scale approach forces data at all scales to be represented at the finest scale, resulting in increased computational requirements.

Another shortcoming of conventional PCA and its extensions is that its ability to reduce the error by eliminating some components is limited, since an embedded error of magnitude proportional to the number of selected components will always contaminate the PCA model (Malinowski, 1991). This limited ability of PCA to remove the error deteriorates the quality of the underlying model captured by the retained components, and adversely affects the performance of PCA in a variety of applications. For example, in process monitoring by PCA, due to the presence of errors, detection of small deviations may not be possible and that of larger deviations may be delayed. Similarly, contamination by the embedded error also deteriorates the quality of the gross-error detection and estimation of missing data. Consequently, the performance of PCA may be improved by methods that allow better separation of the errors from the underlying signal.

A popular approach for improving the separation between the errors and the underlying signal is to pretreat the measurements for each variable by an appropriate filter. Linear filters represent the data at a single scale, and suffer form the disadvantages of single-scale PCA. Nonlinear filters are multiscale in nature, and cause less distortion of the retained features, but perform best for piecewise constant or slowly varying signals, and are often restricted to off-line use. The recent development of wavelet-based methods (Donoho et al., 1995) overcomes the disadvantages of other nonlinear filters, and can be used on-line for all types of signals (Nounou and Bakshi, 1998). Despite these advances in filtering methods, preprocessing of the measured variables is still not a good idea, since it usually

destroys the multivariate nature of the process data, which is essential for multivariate SPM and other operation tasks (MacGregor, 1994).

For reaping the benefits of reducing errors by filtering to improve process monitoring, it is necessary to use an integrated approach to both these tasks. For this reason the approach developed by Bakshi (1998) is used who combined the ability of PCA to extract the relationship between the variables and decorrelate the cross-correlation with the ability of wavelets to extract features in the measurements and approximately decorrelate the autocorrelation. This multiscale approach for modeling by PCA can also be generalized to transform other single-scale empirical modeling methods to multiscale modeling. Interesting enough, multiscale modeling has received surprisingly little attention, despite the fact that most existing modeling methods are inherently single scale in nature, whereas most data contain contributions at multiple scales.

The reconstructed signal in the time domain is generated from the large wavelet coefficients and therefore MSPCA integrates the task of monitoring with that of extracting the signal features representing abnormal operation, with minimum distortion and time delay. Consequently, there is no need for a separate step for prefiltering the measured variables (Bakshi, 1998)

7.3. Methodology of MSPCA

The MSPCA methodology consists of decomposing each variable on a selected family of wavelets according to the methods discussed in Chapter 6. The PCA model is then determined independently for the coefficients at each scale. All the scales (approximations and details) are then combined to yield the model for all scales together.

The approach of computing the PCA of the wavelet coefficients instead of the time-domain data, and its application to process monitoring has also been suggested by Kosanovich and Piovoso (1997). Their approach preprocesses the data by the univariate FMH filter and then transforms it to the wavelet domain before applying PCA to the coefficients. This approach does not fully exploit the benefits of multiscale modeling, and the univariate filtering is not integrated with the PCA. Furthermore, monitoring a process based only on its wavelet decomposition will result in too many false alarms after a process returns to normal operation.

MSPCA combines the ability of PCA to extract the cross-correlation or relationship between the variables with that of orthonormal wavelets to separate deterministic features from stochastic processes and approximately decorrelate the autocorrelation among the measurements. The steps in the MSPCA methodology are shown in Figure 6.21 in Chapter 6 and the following algorithm:

1. for each column in the data matrix,
2. compute wavelet decomposition
3. apply level dependent wavelet thresholding
4. end
5. for each scale that contains important information,
6. compute covariance matrix of wavelet coefficients at selected scale
7. compute PCA loadings and scores of wavelet coefficients
8. select appropriate number of loadings
9. end
10. for all scales together, repeat steps 6 to 8
11. reconstruct approximate data matrix from the selected and thresholded scores at each scale
12. end

Steps 11 and 12 only serve to evaluate the linear principal component model and do not serve a purpose in the monitoring scheme. Steps 11 and 12 are also evaluated for the nonlinear principal component model discussed in Chapter 9, which will form part of the monitoring methodology.

To combine the benefits of PCA and wavelets, the measurements for each variable (column) are decomposed to the column's wavelet coefficients using the same orthonormal wavelet for each variable. This results in transformation of the data matrix, $\mathbf{X}$, into the matrix, $\mathbf{WX}$, where $\mathbf{W}$ is an $n \times n$ orthonormal wavelet transformation operator containing the filter coefficients,

$$\mathbf{W} = \begin{bmatrix} h_{L,1} & h_{L,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & h_{L,N} \\ g_{L,1} & g_{L,2} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & g_{L,N} \\ g_{L-1,1} & \cdot & \cdot & \cdot & g_{L-1,\frac{N}{2}} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & g_{L-1,\frac{N}{2}-1} & \cdot & \cdot & \cdot & g_{L-1,N} \\ \cdot & \cdot & & & \cdot & \cdot & & & & \cdot \\ \cdot & & \cdot & & \cdot & \cdot & \cdot & & & \cdot \\ \cdot & & & \cdot & \cdot & \cdot & & \cdot & & \cdot \\ g_{1,1} & g_{1,2} & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & & 0 \\ \cdot & & & & \cdot & \cdot & & & \cdot & \cdot \\ 0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & g_{1,N-1} & g_{1,N} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_L \\ \mathbf{G}_L \\ \mathbf{G}_{L-1} \\ \cdot \\ \cdot \\ \mathbf{G}_m \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{G}_1 \end{bmatrix}, \qquad (7.1)$$

where $\mathbf{G}_m$ is the $2^{\log_2 n - m} \times n$ matrix containing wavelet filter coefficients corresponding to scale $m = 1, 2, \ldots, L$, and $\mathbf{H}_L$ is the matrix of scaling-function filter coefficients at the coarsest scale discussed in Chapter 6. The matrix, $\mathbf{WX}$, is the same size as the

original matrix, $\mathbf{X}$, but due to the wavelet decomposition, the deterministic component in each variable in $\mathbf{X}$ is concentrated in a relatively small number of coefficients in $\mathbf{WX}$, while the stochastic component in each variable is approximately decorrelated in $\mathbf{WX}$, and is spread over all components according to its power spectrum.

The covariance of the wavelet transformed matrix, and equivalently of the original data matrix, may be written in terms of the contribution at multiple scales as

$$(WX)^T(WX) = (\mathbf{H}_L\mathbf{X})^T(\mathbf{H}_L\mathbf{X}) + (\mathbf{G}_L\mathbf{X})^T(\mathbf{G}_L\mathbf{X}) + \ldots + (\mathbf{G}_m\mathbf{X})^T(\mathbf{G}_m\mathbf{X}) + \ldots + (\mathbf{G}_1\mathbf{X})^T(\mathbf{G}_1\mathbf{X}) \tag{7.2}$$

To exploit the multiscale properties of the data, the PCA of the covariance matrix of the coefficients at each scale is computed independently of the other scales. The resulting scores at each scale are not cross-correlated due to PCA, and their autocorrelation is approximately decorrelated due to the wavelet decomposition. Depending on the nature of the application, a smaller subset of the principal-component scores and wavelet coefficients may be selected at each scale. The number of principal components to be retained at each scale are not changed due to the wavelet decomposition since it does not affect the underlying relationship between the variables at any scale. Consequently, existing methods such as cross-validation may be applied to the data matrix in the time domain or to all the wavelet coefficients to select the relevant number of components. This is done to ensure that only those principal components associated with noise are discarded. A cross-validation method for selecting the relevant number of components will be discussed in Section 7.10. Applying separate thresholds at each scale as discussed in Chapter 6 allows MSPCA to be more sensitive to scale-varying signal features such as autocorrelated measurements. Thresholding of the coefficients at each scale identifies the region of the time-frequency space and scales where there is significant contribution from the deterministic features in the signal and also helps in denoising the signal.

The covariance matrix for computing the loadings and scores for all scales together is computed by combining the covariance matrices of all the approximations and details.

This MSPCA modeling method represents one way of using the PCA models at multiple scales, and other approaches may be devised, depending on the application.

Instead of the MSPCA methodology described earlier, some of the benefits of the wavelet representation may be reaped by just transforming the measured data on a selected wavelet basis and computing the PCA of $\mathbf{WX}$ instead of $\mathbf{X}$. PCA of $\mathbf{WX}$ will make it easier to separate deterministic features in a stochastic process, but this approach will be restricted to off-line use and will not fully exploit the benefits of the multiscale representation,

since it will implicitly assume that the nature of the data does not change with scale. This assumption will cause too many false alarms for autocorrelated measurements, as compared to the MSPCA approach that accounts for the scale-dependent power spectrum. False alarms will also be created for process monitoring based on the scores of $\mathbf{WX}$ after a process returns to normal operation.

7.4. Principle of LPCA

Principal Component Analysis (PCA) is a multivariate technique in which a number of related variables are transformed to (hopefully) a smaller set of uncorrelated variables.

PCA transforms the data matrix in a statistically optimal manner by diagonalizing the covariance matrix by extracting the cross-correlation or relationship between the variables in the data matrix. If the measured variables are linearly related and contaminated by errors, the first few components capture the relationship between the variables, and the remaining components are composed only of the error. Thus, eliminating the less important components reduces the contribution of errors in the measured data and represents it in a compact manner. Applications of PCA rely on its ability to reduce the dimensionality of the data matrix while capturing the underlying variation and relationship between the variables.

7.5. Characteristic roots and vectors

The method of principal components is based on a key result from matrix algebra: A $p \times p$ symmetric, nonsingular matrix, such as the covariance matrix $\mathbf{S}$, may be reduced to a diagonal matrix $\mathbf{L}$ by premultiplying and postmultiplying it by a particular orthonormal matrix $\mathbf{U}$ such that

$$\mathbf{U'SU} = \mathbf{L} \tag{7.3}$$

The diagonal elements of $\mathbf{L}$, $l_1, l_2, \ldots, l_p$ are called the characteristic roots, latent roots or eigenvalues of $\mathbf{S}$. The columns of $\mathbf{U}$, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p$ are called the characteristic vectors or eigenvectors of $\mathbf{S}$. The characteristic roots may be obtained from the solution of the following detrimental equation, called the characteristic equation:

$$|\mathbf{S} - l\mathbf{I}| = \mathbf{0} \tag{7.4}$$

where $\mathbf{I}$ is the identity matrix. This equation produces a pth degree polynomial in l from which the values $l_1, l_2, \ldots, l_p$ are obtained.

7.6. The method of principal components

The starting point for PCA is the sample covariance matrix $\mathbf{S}$ (or the correlation matrix). For a p-variable problem,

$$\mathbf{S} = \begin{bmatrix} s_1^2 & s_{12} & \cdots & s_{1p} \\ s_{12} & s_2^2 & \cdots & s_{2p} \\ \vdots & \vdots & & \vdots \\ s_{1p} & s_{2p} & \cdots & s_p^2 \end{bmatrix} \tag{7.5}$$

Where s_i^2 is the variance of the ith variable, x_i, and s_{ij} is the covariance between the ith and jth variables. If the covariances are not equal to zero, it indicates that a linear relationship exists between these two variables, the strength of that relationship being represented by the correlation coefficient, $r_{ij} = s_{ij}/(s_i s_j)$.

The principal component transformation will transform the p correlated variables $x_1, x_2, \ldots, x_p$ into p new uncorrelated variables $z_1, z_2, \ldots, z_p$. The coordinate axes of these new variables are described by the characteristic vector $\mathbf{u}_i$ which make up the matrix $\mathbf{U}$ of the direction cosines used in the transformation:

$$\mathbf{z} = \mathbf{U}'[\mathbf{x} - \bar{\mathbf{x}}] \tag{7.6}$$

Here $\mathbf{x}$ and $\bar{\mathbf{x}}$ are $p \times 1$ vectors of observations on the original variables and their means.

The transformed variables are called the principal components of $\mathbf{x}$ or pc's for short. The ith principal component is

$$\mathbf{z}_i = \mathbf{u}_i'[\mathbf{x} - \bar{\mathbf{x}}] \tag{7.5}$$

and will have mean zero and variance l, the ith characteristic root. To distinguish between the transformed variables and the transformed observations, the transformed variables will be called principal components and the individual transformed observations will be called z-scores. The distinction is made here with regard to z-scores because another normalization of these scores exists.

7.7. Some properties of principal components

7.7.1. TRANSFORMATIONS

If one wishes to transform a set of variables $\mathbf{x}$ by a linear transformation $\mathbf{z}_i = \mathbf{u}_i'[\mathbf{x} - \bar{\mathbf{x}}]$ whether $\mathbf{U}$ is orthonormal or not, the covariance matrix of the new variables, $\mathbf{S}_z$, can be determined directly from the covariance matrix of the original observations, $\mathbf{S}$, by the relationship

$$\mathbf{S}_z = \mathbf{U}'\mathbf{S}\mathbf{U} \tag{7.8}$$

However, the fact that $\mathbf{U}$ is orthonormal is not a sufficient condition for the transformed variables to be uncorrelated. Only this characteristic vector solution will produce an $\mathbf{S}_z$ that is a diagonal matrix like $\mathbf{L}$ producing new variables that are uncorrelated.

7.7.2. INTERPRETATION OF PRINCIPAL COMPONENTS

The interpretation of principal components will be explained by means of an example. For the purpose of illustration, linear principal component analysis was applied to the first two variables from the industrial data. The coefficients of the first vector were found to be 0.7236 and 0.6902. As observed, they are nearly equal and both positive, indicating that the first pc, z_1, is a weighted average of both variables. This is related to variability that x_1 and x_2 have in common; in the absence of correlated errors of measurement, this would be assumed to represent process variability. The coefficients of the second vector were -0.6902 and 0.7236. They are also nearly equal except for sign indicating that the second pc, z_2, represent differences in the measurements for the two variables that would probably represent testing and measurement variability.

7.7.3. GENERALIZED MEASURES AND COMPONENTS OF VARIABILITY

In keeping with the goal of multivariate analysis of summarizing results with as few numbers as possible, there are two single-number quantities for measuring the overall variability of a set of multivariate data. These are

1. The determinant of the covariance matrix, $|\mathbf{S}|$. This is called the generalized variance. The square root of this quantity is proportional to the area or volume generated by a set of data.

2. The sum of the variances of the variables:

$$s_1^2 + s_2^2 + \ldots + s_p^2 = Tr(\mathbf{S}) \quad (trace\ of\ \mathbf{S}) \tag{7.9}$$

Conceivably, there are other measures of generalized variability that may have certain desirable properties but these two are the ones that have found general acceptance among practitioners.

A useful property of PCA is that the sum of the original variances is equal to the sum of the characteristic roots. For the two variable case,

$$\begin{aligned} s_1^2 + s_2^2 &= 0.7986 + 0.7343 = 1.5329 \\ &= 1.4465 + 0.0864 = l_1 + l_2 \end{aligned}$$

This identity is particularly useful because it shows that the characteristic roots, which are the variances of the principal components, may be treated as variance components. The ratio of each characteristic root to the total will indicate the proportion of the total variability accounted for by each pc. For z_1, 1.4456/1.5329 = 0.944 and for z_2, 0.0864/1.5329 = 0.056. This says that roughly 94% of the total variability of these data (as represented by $Tr(\mathbf{S})$) is associated with, accounted for or "explained by" the variability of the process and 6% due to the variability related to testing and measurement. Since the characteristic roots are sample estimates, these proportions are also sample estimates.

7.5.4. CORRELATION OF PRINCIPAL COMPONENTS AND ORIGINAL VARIABLES

It is also possible to determine the correlation of each pc with each of the original variables, which may be useful for diagnostic purposes. The correlation of the ith pc, z_i, and the jth original variable, x_j, is equal to

$$r_{zx} = \frac{u_{ji}\sqrt{l_i}}{s_j} \tag{7.10}$$

For instance, the correlation between z_1 and x_1 is

$$\frac{u_{11}\sqrt{l_1}}{s_1} = \frac{0.7236\sqrt{1.4465}}{\sqrt{0.7986}} = 0.974$$

The first pc is more highly correlated with the original variables than the second. This is to be expected because the first pc accounts for more variability than the second. Note that the sum of squares of each row is equal to 1.0.

7.8. Scaling of Data

7.8.1. INTRODUCTION

There are two ways of scaling principal components, one by rescaling the original variables, which will be discussed here, and the other by rescaling the characteristic vectors, which won't be discussed here. There is no significant reason for choosing the one method above the other and in the end remains a personal choice. To me, scaling the original variables made more sense.

The results obtained by scaling the original data will depend on the method employed. Once the method of scaling is selected, the PCA operations will proceed, for the most part, as described earlier but there ,will be some modifications unique to each method. The main effect of this choice will be on the matrix from which the characteristic vectors are obtained.

Specifically, the following three methods will be considered:

1. No scaling at all. The final variate vector is $\mathbf{x}$.
2. Scaling the data such that each variable has zero mean (i.e., in terms of deviation from the mean). The final variate vector is $\mathbf{x}-\bar{\mathbf{x}}$.
3. Scaling the data such that each variable is in standard units. (i.e., has zero mean and unit standard deviation). Each variable is expressed as $(x_i-\bar{x}_i)/s_i$.

As stated above, the choice of scale will determine the dispersion matrix used to obtain the characteristic vectors. If no scaling is employed, the resultant matrix will be the product or second moment matrix; if the mean is subtracted, it will be the covariance matrix; if the data are in standard units, it will be a correlation matrix.

7.8.2. DATA AS DEVIATIONS FROM THE MEAN: COVARIANCE MATRICES

Here it is not necessary to actually subtract the variable means from the data; the operations required to obtain the covariance matrix will take care of it.

If one were to subtract the means from the data and use these deviations as a data set, say $\mathbf{x_d}=\mathbf{x}-\bar{\mathbf{x}}$ with the resulting $n\times p$ data matrix $\mathbf{X}_d$, the covariance matrix would be $\mathbf{X}_d'\mathbf{X}_d/(n-1)$. The characteristic vectors $\mathbf{U}, \mathbf{V}$ and $\mathbf{W}$ would stay the same. For large problems in terms of sample size or large number of digits for the original data, this option may be preferable, numerically, to obtaining $\mathbf{S}$ from the raw data directly.

There are many occasions when one cannot use the covariance matrix. There are two reasons for this:

1. The original variables are in different units. In this case, the operations involving the trace of the covariance matrix have no meaning. For instance, if a variable is expressed in centimeters, its variance is 100 times what it would be if it were expressed in millimeters (variance of variable in cm divided by variance of variable in mm). The variable would now exert considerable more influence on the shaping of the pc's since PCA is concerned with explaining variability. When the units are different, the solution is to make the variances the same (i.e., use standard units), which makes the covariance matrix into a correlation matrix.

2. Even if the original variables are in the same units, the variances may differ widely, often because they are related to their means. If this gives undue weight to certain variables, the correlation matrix should be employed here also (unless, possibly, taking logs of the variables or the use of some other variance-stabilizing transformation will suffice).

Nevertheless, when the variables are in the same units and do have the same amount of variability, there are some advantages in using covariance matrices. This is particularly true in physical applications where PCA is used in building physical models. Using the covariance matrix should also help with diagnostics since the $\mathbf{V}$-vectors are in the original units of the variables.

It is important to note that there is no one-to-one correspondence between the pc's obtained from a correlation matrix and those obtained from a covariance matrix. The more heterogeneous variances are, the larger the difference will be between the two sets of vectors. If the covariance matrix has $(p-k)$ zero roots, then the correlation matrix will also have $(p-k)$ zero roots. However, if the covariance matrix has $(p-k)$ equal roots, the correlation matrix will not necessarily have the same number.

7.9. Using Principal Components in Quality Control

7.9.1. TYPE I ERRORS

When one uses two or more control charts (i.e. time-series plot of the squared prediction error, SPE) simultaneously, some problems arise with the type I error. This is the probability of a sample result being outside the control limits when the process is at the mean or the standard established for that process. If one would consider first the two control charts for x_1 and x_2 which is variable one and variable two of the training

data (see Paragraph 3.11 of Chapter 3), the probability that each of them will be in control if the process is on standard is 0.95. If these two variables were uncorrelated (which they are not in this case), the probability that both of them would be in control is 0.95^2 = 0.9025 so the effective Type I error is roughly $\alpha = 0.10$, not 0.05. For 8 uncorrelated variables, the Type 1 error would be $1-(0.95^8) = 0.37$. Thus if one was attempting to control 8 independent variables, at least one or more of these variables would indicate an out-of-control condition over one-third of the time.

The problem becomes more complicated when the variables are correlated as they are here. If they were perfectly correlated, the Type I error would remain 0.05. However, anything less than that, such as the present case, would leave one with some involved computations to find out what the Type I error really was. The use of principal component control charts resolves some of this problem because the pc's are uncorrelated; hence, the Type I error may be computed directly. This may still leave one with a sinking feeling about looking for trouble that does not exist.

7.9.2. GOALS OF MULTIVARIATE QUALITY CONTROL

Any multivariate quality control procedure, whether or not PCA is employed, should fulfill four conditions.

1. A single answer should be available to answer the question: "Is the process in control?"
2. An overall Type I error should be specified.
3. The procedure should take into account the relationships among the variables.
4. Procedures should be available to answer the question: "If the process is out-of-control, what is the problem?"

Condition 4 is much more difficult than the other three, particularly as the number of variables increases since it needs expert information which is more than just the mathematical or statistical information needed in the first three conditions. There is usually no easy way to this, although the use of PCA may help.

7.10. Selecting the number of principal components

7.10.1. INTRODUCTION

One of the greatest uses of PCA is its potential ability to adequately represent a $p-$variable data set in $k < p$ dimensions. The question becomes: "What is k?"

Obviously, the larger k is, the better the fit of the PCA model; the smaller k is, the more simple the model will be. Somewhere, there is an optimal value of k; what is it? To determine k, there must be a criterion for optimality.

One method for determining the optimum number of pc's is the cross-validation approach by Wold (1976, 1978), Eastment and Krzanowski (1982) and Krzanowski (1983, 1987). This approach is recommended when the initial intention of a study is to construct a PCA model with which future sets of data will be evaluated as in this case.

In PCA it consists of randomly dividing the sample into g groups. The first group is deleted from the sample and a PCA is performed on the remaining sample. The vectors obtained from that reduced sample are used to obtain pc's and Q-statistics (explained in more detail in Chapter 10) for the deleted group. That group is returned to the sample, the next group is deleted, and the procedure is repeated g times. The grand average of the Q-statistic, divided by p, is called the PRESS-statistic (PREdiction Sum of Squares). Its primary use in PCA is as a stopping rule. It differs from other stopping rules in that it is based on the Q-statistic rather than the characteristic roots. Krzanowski pointed out that it is possible to have different data sets produce the same covariance or correlation matrix but would probably produce different PRESS-statistics although their characteristic roots would be the same. Cross-validation also differs from other stopping rules in requiring the original data while other procedures work directly from the covariance or correlation matrix. Although the procedure described here is not a significance test, it is more quantitative than most other stopping rules.

7.10.2. A SIMPLE CROSS-VALIDATION PROCEDURE

The principle of cross-validation as a stopping rule will be illustrated using dataset 1 (refer to Chapter 6 paragraph 6.9.2.4.), where $p = 8$ and $n = 645$. The transformed data matrix will be denoted by the 645×8 matrix $\mathbf{X}$. For this case the data set will be divided into $g = 5$ groups of 20 observations each so that the first group will be observations 1-20, the second group 21-40, and so on. The procedure is as follows:

1. Delete the first group from the sample. Perform a PCA on the remaining observations (i.e., 21-645). Obtain all eight vectors. This example used a correlation matrix, the data will be in standard units.

2. For the deleted sample, obtain all eight z-scores for each observation using the vectors obtained in step 1.

3. Using, in turn, the first pc, the first two pc's, and so on, obtain the predicted values of the deleted sample. $\bar{\mathbf{x}}$ will be equal to zero.

4. For each observation in the deleted sample, obtain Q. For the first observation, $Q = 1.054$ for one pc, $Q = 0.639$ for two pc's, and so on.

5. Return the deleted group to the sample and remove the second group. Repeat steps 1-4. Do the same for the other three groups. This concluded, there will now be 645 values of Q for a one-pc model, another 645 for a two-pc model, and so on.

6. For each pc model, add up the 645 Q-statistics and divide each sum by $np = 5160$. These are called PRESS-statistics and be designated by PRESS(1), PRESS(2), and so on. It will also be necessary to obtain PRESS(0), the sum of squares of the original data, again assuming a mean of zero.

7. To determine whether the addition of another pc, say the k th pc, to the model is warranted, form the statistic

$$W = \frac{[PRESS(k-1) - PRESS(k)]/D_M}{PRESS(k)/D_R} \tag{7.11}$$

where

$$D_M = n + p - 2k \tag{7.12}$$

$$D_R = p(n-1) - \sum_{i=1}^{k}(n + p - 2i) \tag{7.13}$$

If $W > 1$, then retain the k th pc in the model and test the $(k+1)$ st. for example, to test whether the first pc should be included, one would form

$$W = \frac{[PRESS(0) - PRESS(1)]/651}{PRESS(1)/4501} = \frac{0.0077}{0.00066303} = 11.62$$

and the first pc would be included in the model. For the process data the process terminated with the inclusion of the third principal component.

In practice, if one had a large number of variables and was confident that only a small number of pc's would be retained, a different strategy might be employed, in which each characteristic vector is obtained and tested sequentially before obtaining the next and, in that way, only one unwanted vector is obtained.

Table 7.1. PRESS values for selecting the number of pc's to retain

k	$PRESS(k)$	D_M	D_R	W
0	8.0000			
1	2.9843	651	4501	11.6200
2	2.2097	649	3852	2.0806
3	1.1594	647	3205	4.4873
4	0.9813	645	2560	0.7204<1
5	0.9634	643	1917	0.0555
6	0.8834	641	1276	0.1804
7	0.8811	639	637	0.0025

It is possible that if one were to continue this process beyond the first occurrence where $W < 1$, later values of k might produce one or more occurrences of $W > 1$. This may be due to the presence of outliers.

7.10.3. ENHANCEMENTS

In the previous section $\bar{\mathbf{x}}$ was assumed to be zero since it was equal to zero for the entire example. However, it may be that the mean is not equal to zero and the cross-validation technique for it is more complicated, involving the deletion of variables. Furthermore, both $\mathbf{U}$ and $\mathbf{z}$ are considered estimates and, as we now know, may be estimated simultaneously using singular value decomposition. This, of course, is not possible here because the $\mathbf{z}$-scores are obtained for the observations not included in the sample from which $\mathbf{U}$ is obtained. The solution to this problem is to use all n observations in each subsample but randomly delete elements from each data vector. The good way to do this is to randomly order the observations and use a cyclic deletion pattern given by Wold (1987). The estimation procedure will, of necessity, require SVD but the SVD algorithm employed must be able to handle missing data.

7.11. Application

7.11.1. SOFTWARE SETUP

Figure 7.1. displays the LPCA interface which can be used to apply LPCA to the data. This interface can either be displayed from the main interface or by using the Next button from the wavelet analysis interface.

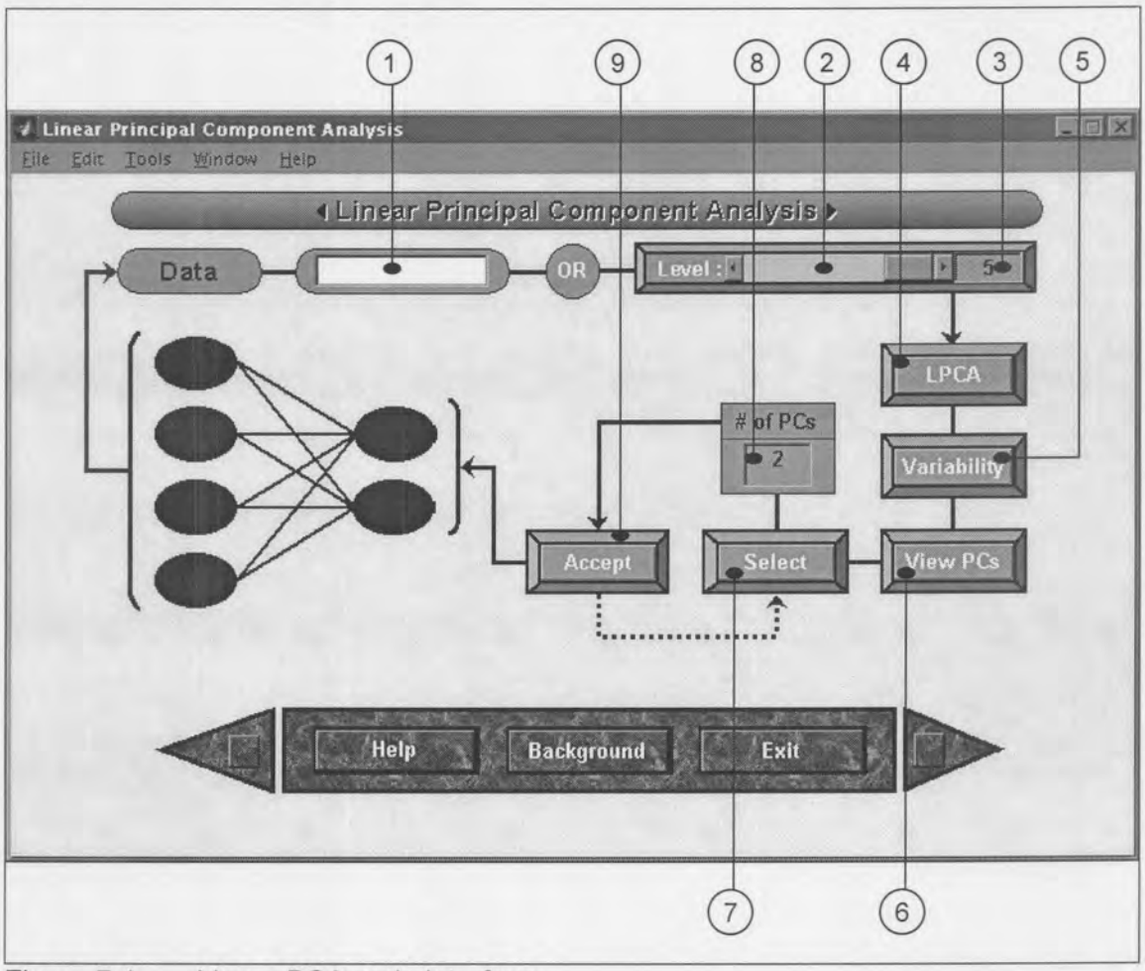

Figure 7.1. Linear PCA main interface

Figure 7.1 Tags:

1. Name of the variable containing the data to which linear PCA needs to be applied. If this space is left blank the default variable from the database will be used. The variable must contain more than one column of data.

2. Level selection slider. This slider is used if the default data from the database is used. Since LPCA is applied to each level separately, the level to which LPCA needs to be applied can be selected using this slider. It will automatically detect the number of levels contained in the database.

3. Level number display.

4. LPCA application button. Using this button will apply PCA to the named variable or the specified level in the database.

5. Variability interface used to view the variability of the principal components. This button will open Figure 7.2.

6. Principal component viewer interface used to view each principal component separately. This button will open Figure 7.3.

7. Number of principal components selection interface. This button will open Figure 7.4.

8. Specify the final number of principal components to select. This choice is based on the results obtained from the principal components selection interface in Figure 7.4.

9. Accept the number of principal components specified in 8. This reduces the number of principal components to the number specified and saves the results to the database.

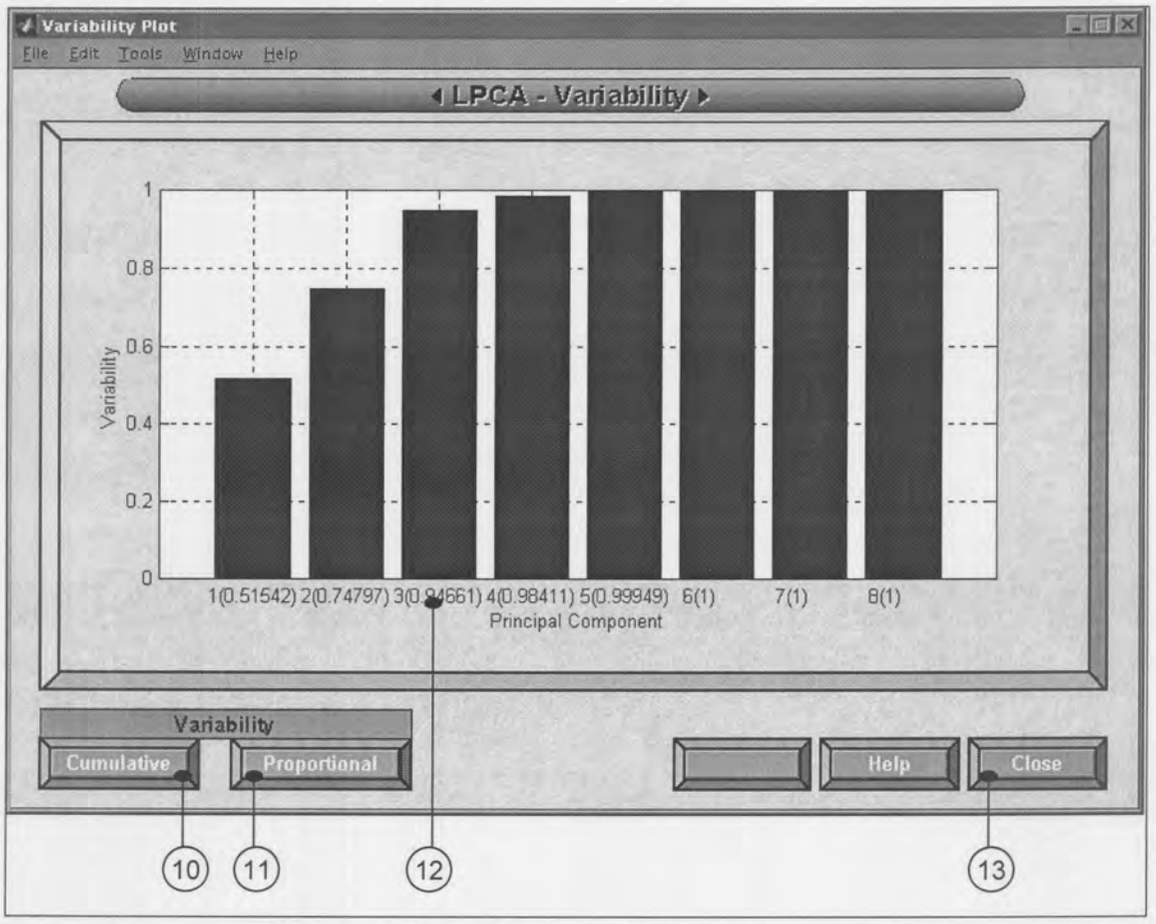

Figure 7.2. Cumulative variability plot

Figure 7.2 Tags:

10. Cumulative variability. This button displays Figure 7.2.

11. Proportional variability. This button displays Figure 7.3.

12. Display of the individual contribution of each variable to the total variability.

13. Close the current interface and return to the LPCA interface.

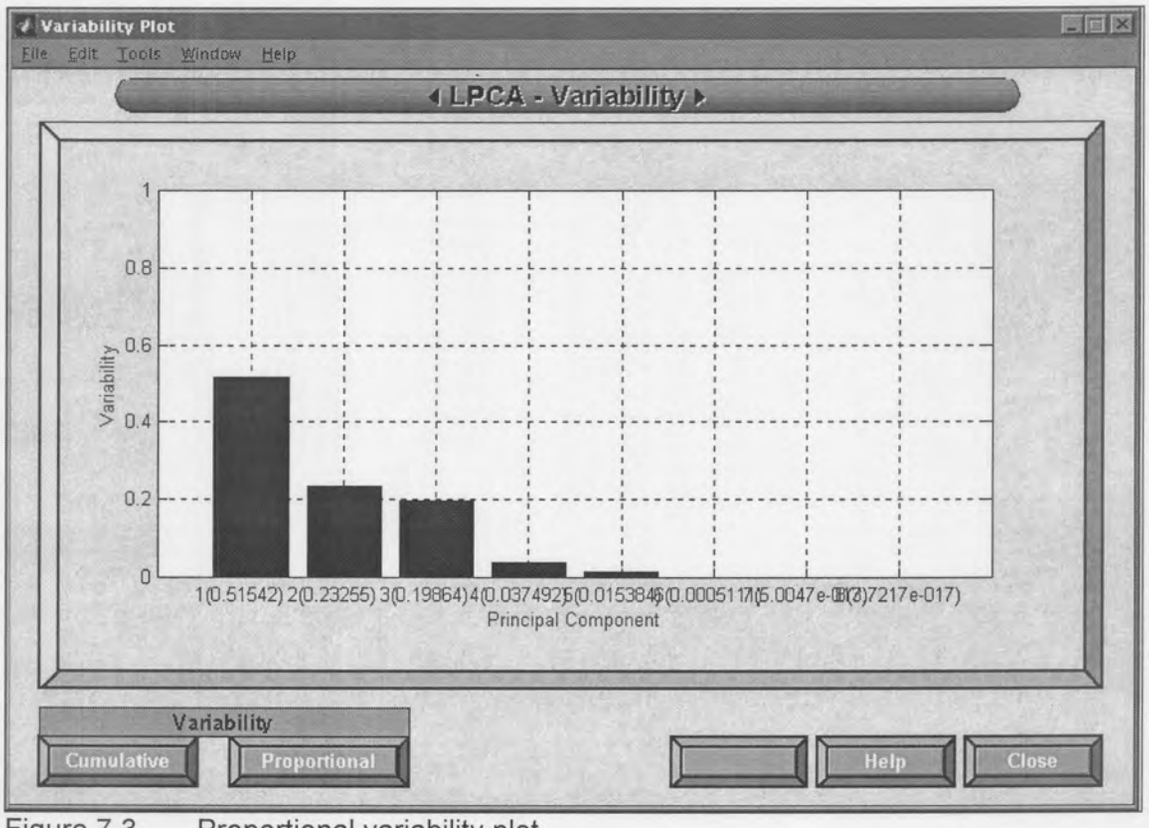

Figure 7.3. Proportional variability plot

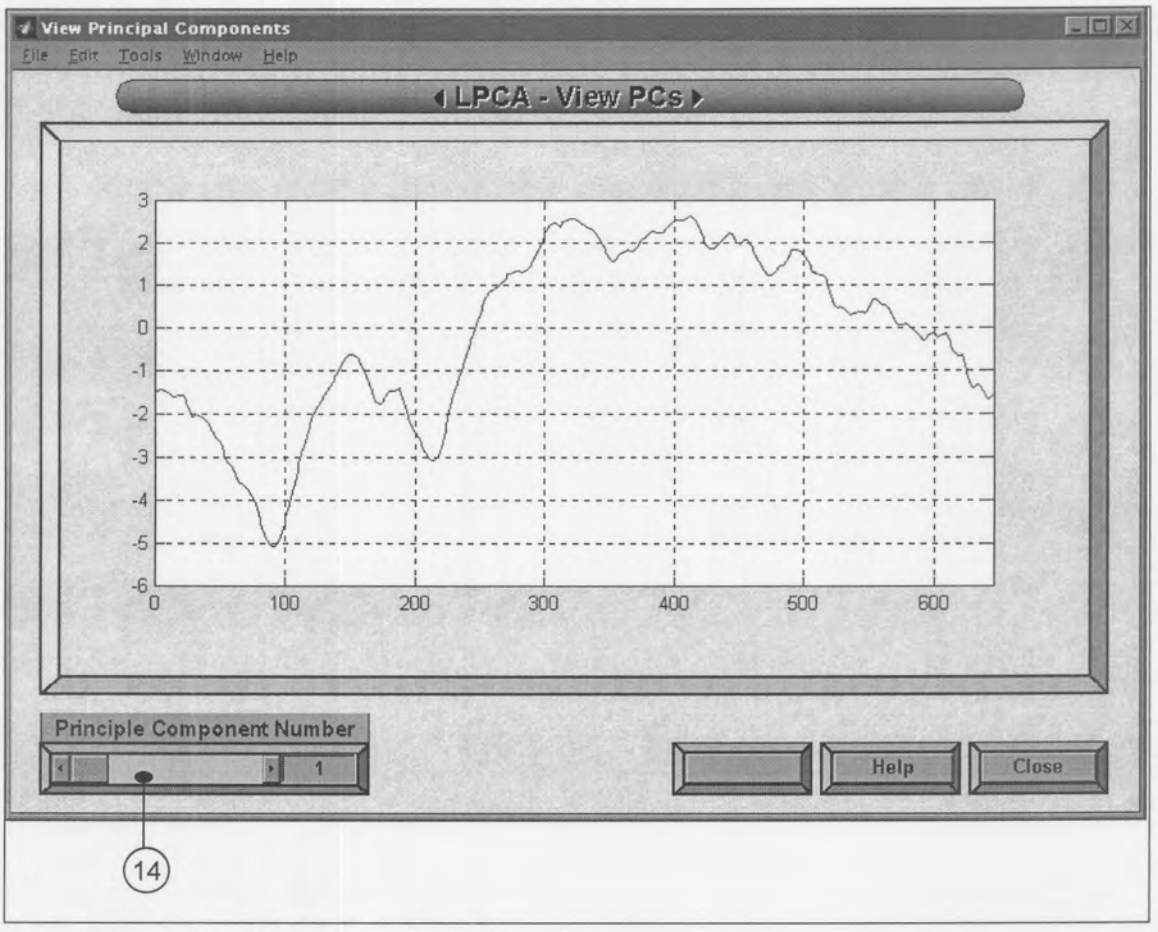

Figure 7.4. Principal Component plot

Figure 7.4 Tags:

14. Principal component number to display.

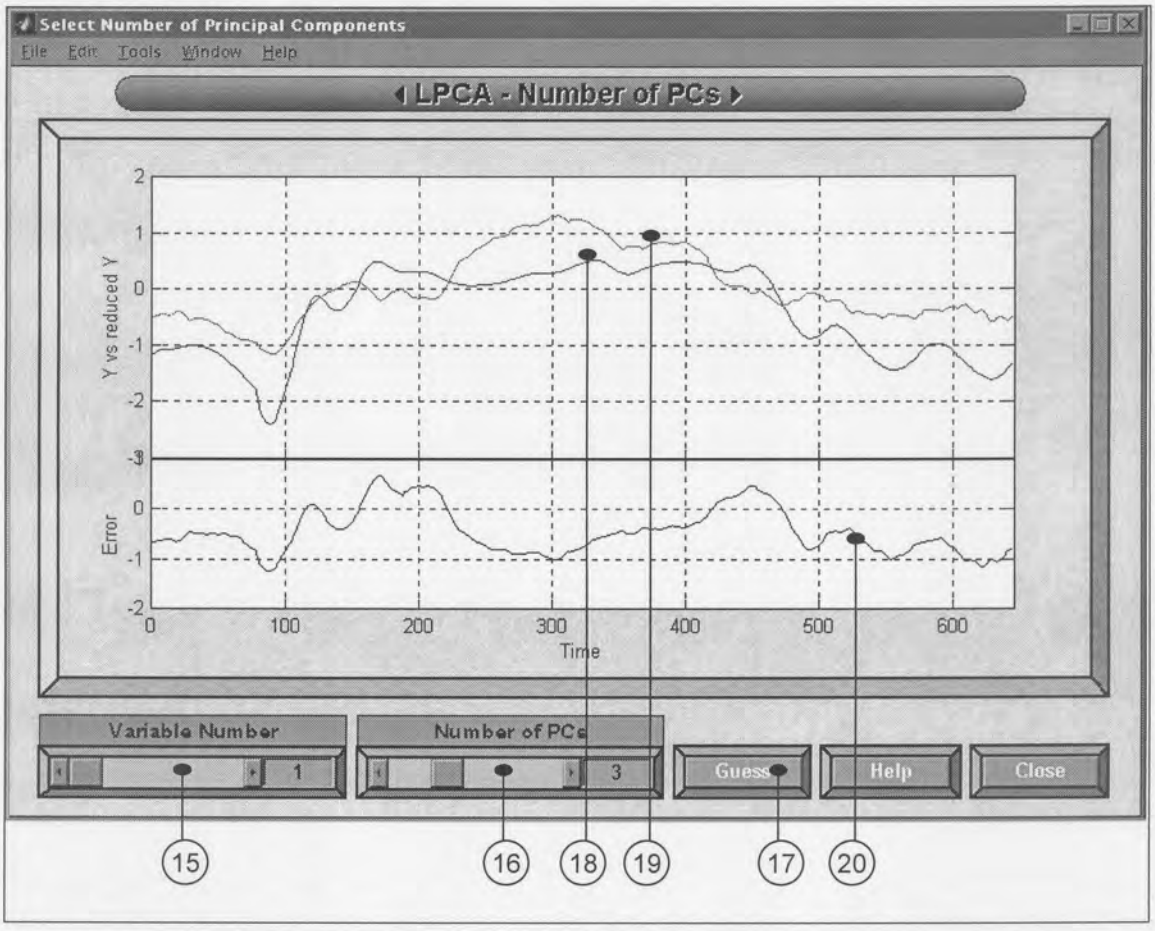

Figure 7.5. Principal component selection interface

Figure 7.5 Tags:

15. Select the original variable number from the current level to display.

16. Reconstruct the original variable in 15 using the number of principal components specified here.

17. As an option you can choose the number of principal components to retain using the Krzanowski cross-validation method based on PRESS values discussed in Section 7.8.

18. Original variable

19. Reconstruction of the original variable

20. Error plot between the original and reconstructed variable using the number of principal components specified.

7.11.2. EXPERIMENTAL

Once the data matrix of the combined non-thresholded details and approximations (dataset 2) and matrix of approximations (dataset 1) from Chapter 6 had been obtained, the next step was to remove any data points which did not correspond to nominal process operation. By visual inspection ten coefficients from dataset 1 and 30 from dataset 2 (refer to Chapter 6 paragraph 6.9.2.4.) were identified as not being representative of normal operation. Dataset 2 contained 40 columns and Dataset 1 eight columns. Linear PCA was applied to the resultant data sets. The cross-validation method using Krzanowski's PRESS-statistic (Krzanowski, 1987) was used to select the appropriate number of principal components. This technique indicated that three and four linear principal components were adequate to explain the underlying variability in dataset 1 and dataset 2 respectively . For dataset 2 this meant four out of a possible 40 and for dataset 1, three out of a possible eight, indicating a high degree of correlation. A total of 94.47% and 93.5% of the total explained variance was captured by the two sets of principal components respectively.

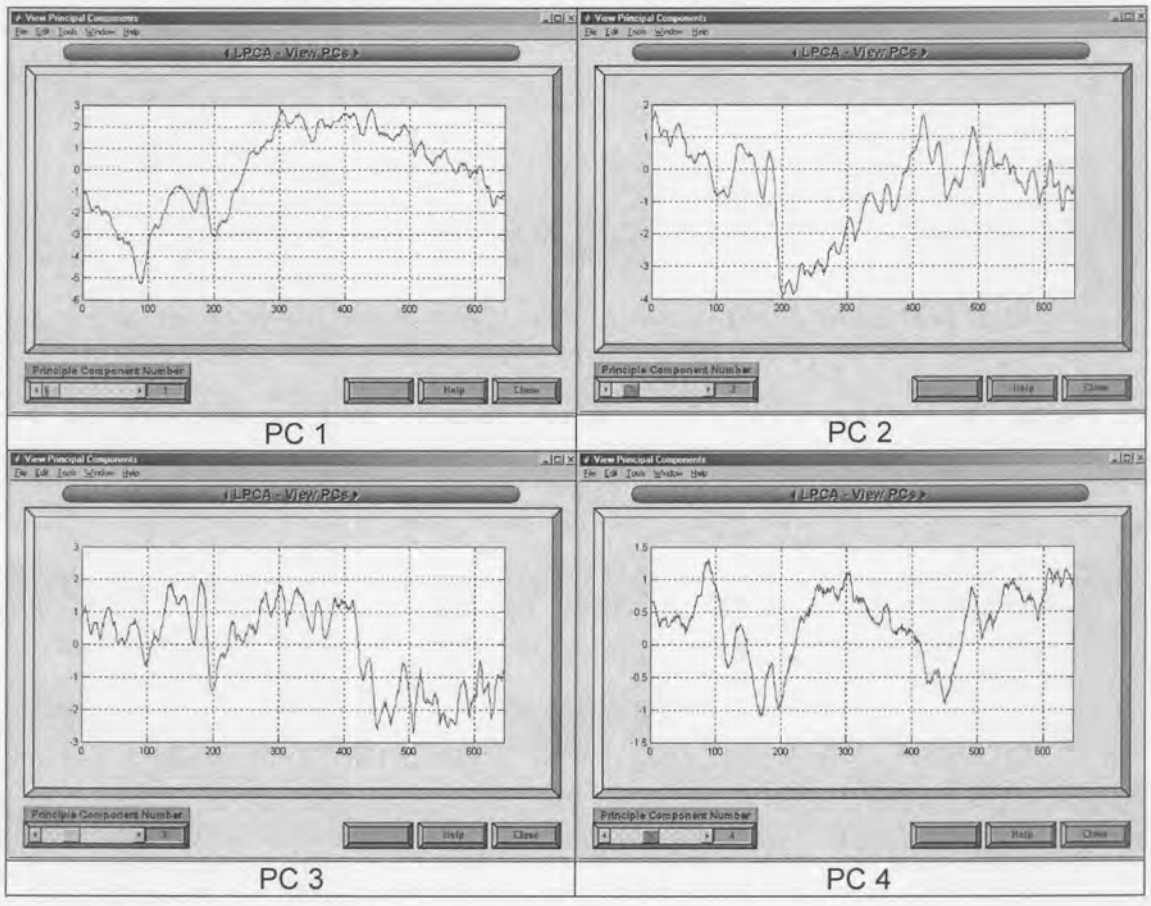

Figure 7.6. First four principal components of dataset 1.

Figure 7.6 gives the first four linear principal components of dataset 1 of which the first three were retained. Table 7.2 gives a summary of the cumulative variability of the eight and first eight principal components of dataset 1 and dataset 2 respectively.

Table 7.2. Cumulative variability for pc's of dataset 1 and dataset 2

# Principal Components	Cumulative Variability For Dataset 1	Cumulative Variability For Dataset 2
1	0.50957	0.48375
2	0.73978	0.70285
3	0.94467	0.89768
4	0.98321	0.93503
5	0.99796	0.96529
6	0.99934	0.97937
7	0.99994	0.98790
8	1	0.99089

Chapter 8 INPUT TRAINING NEURAL NETWORKS

8.1. Introduction

Many industrial processes such as the case under consideration exhibit significant nonlinear behavior. In these cases the application of PCA is not strictly appropriate. A non-linear PCA methodology is proposed to take account of the non-linearities inherent within the process data.

The non-linear PCA method proposed in this study is based upon the input-training neural network approach (Tan and Mavrovouniotis, 1995). The advantage of this approach is that it enables both the second-order and higher-order correlations to be extracted separately. This is achieved by first applying linear PCA as discussed in Chapter 7 to compress the data prior to implementing non-linear compression. By adopting this procedure, a more parsimonious description of process behaviour is achieved. The methodology is investigated for non-linear process performance monitoring in Chapter 9.

In chapter 7 the concept of multiscale linear principal component analysis (MSLPCA) was introduced as means of dimensionality reduction. In this chapter it will be extended to nonlinear principal component analysis (NLPCA) and in chapter 9 this will be combined with the LPCA from chapter 7 and multiscaling from chapter 6 to form multiscale nonlinear principal component analysis (MSNLPCA). Since NLPCA uses a type of neural network referred to as an input training neural network this concept will first be introduced separately before being applied to MSNLPCA.

Apart from input training neural networks, dimensionality reduction can also be performed by autoassociative neural networks, which are feedforward neural nets trained to perform the identity mapping between network inputs and outputs. Although IT-nets are an improvement over autoassociative neural networks (AANN), AANN's will be briefly discussed in order to realize the similarities and differences between the two.

Figure 8.1 gives a schematic of an AANN. With AANN's dimensionality reduction is achieved through a bottleneck, that is a hidden layer with a small number of nodes. Most previous work focused on single-hidden-layer networks (Ackley et al., 1985; Cottrel et al., 1987; Abbas and Fahmy, 1993). Kramer (1991) pointed out that the single-hidden-layer architecture was unable to model nonlinear relationships between observed variables and latent variables, and consequently offered no significant improvement over conventional PCA. He then established a three-hidden-layer architecture for autoassociative networks to capture nonlinear correlations. The three-hidden-layer autoassociative networks can be used to perform various data screening

tasks, such as data noise filtering, missing measurement replacement, and gross error detection and correction (Kramer, 1992).

An autoassociative network is composed of a mapping subnet and a demapping subnet, each of which is a single-hidden-layer network by itself. Dong and McAvoy (1993) proposed to train the two subnets separately. The method they proposed involves three steps:

(1) find principal curves by successively applying the algorithm of Hastie and Stuetzle (1989) to observed data and residuals;

(2) train a network that maps the original data to principal curves;

(3) train another network that maps principal curves to the original data.

Autoassociative networks are typically trained through backpropagation. In general, the performance of backpropagation deteriorates as the number of hidden layers gets larger (Hertz et al., 1991). The poor performance of backpropagation in training the mapping subnet is attributable to the large number of layers. In the process of backpropagation learning, modifications of weights are based on errors propagated backward from the output layer. After several layers of error propagation, the searching direction for the weights in the mapping subnet may deviate from the direction that minimizes the output error function. This effect becomes even more pronounced for an autoassociative network due to its bottleneck layer.

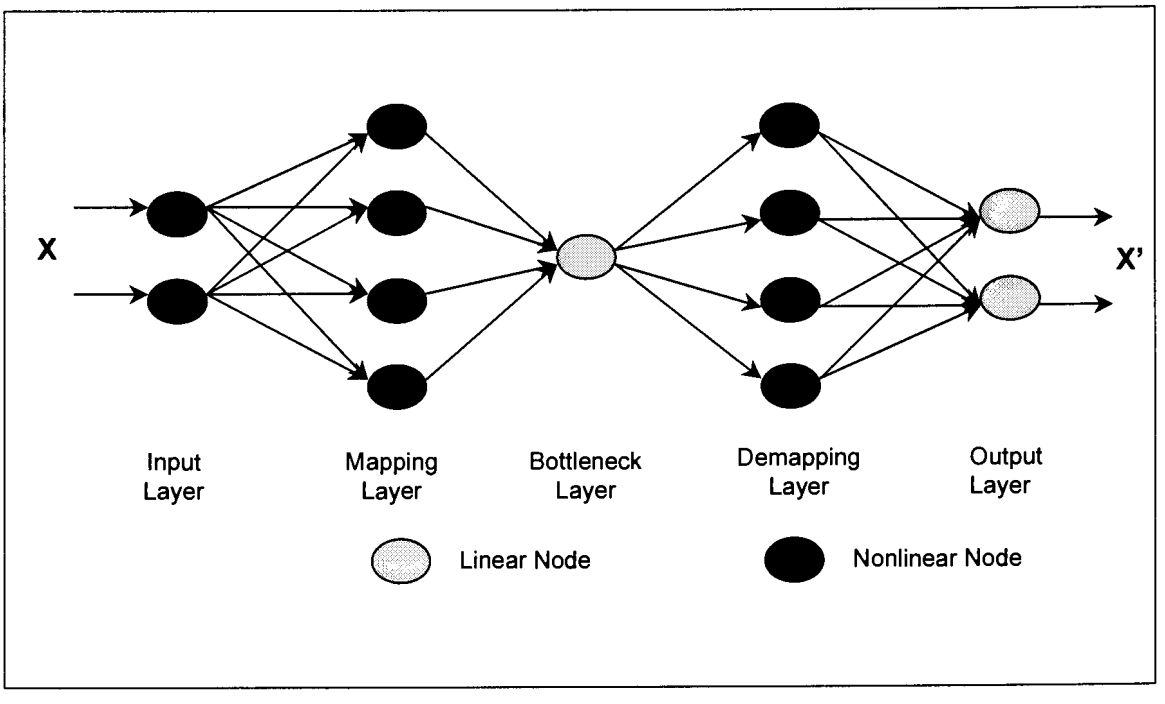

Figure 8.1. A 2-4-1-4-2 autoassociative network

Thus, as an alternative, the new method proposed by Tan and Mavrovouniotis (1995) was used. In their work they also used neural networks as nonlinear models for observed variables and latent variables and additionally used a concept called input training (IT). With this method, only one single-hidden-layer network is needed for dimensionality reduction of a given data set. The method proposed by them however, uses backpropagation to train the network, which is not an optimized training method and tends to take long to converge to the performance goal. In order to overcome this, this work extends the backpropagation training algorithm and combines it with the Levenberg-Marquardt (LM) training algorithm in an effort to facilitate enhanced speed and better convergence. This new enhanced training algorithm forms a significant contribution to the process of NLPCA.

In section 8.2. a background to Backpropagation is given and in section 8.3. background is provided to the Levenberg-Marquardt training algorithm. Section 8.4 explains the concept of input training. Section 8.5. extends section 8.3 and 8.4 to develop an enhanced training algorithm for input training neural networks (IT-nets).

8.2. The Backpropagation algorithm

8.2.1. GENERAL BACKPROPAGATION

There are many variations of the backpropagation algorithm. The simplest implementation of backpropagation learning updates the network weights and biases in the direction in which the performance function decreases most rapidly - the negative of the gradient.

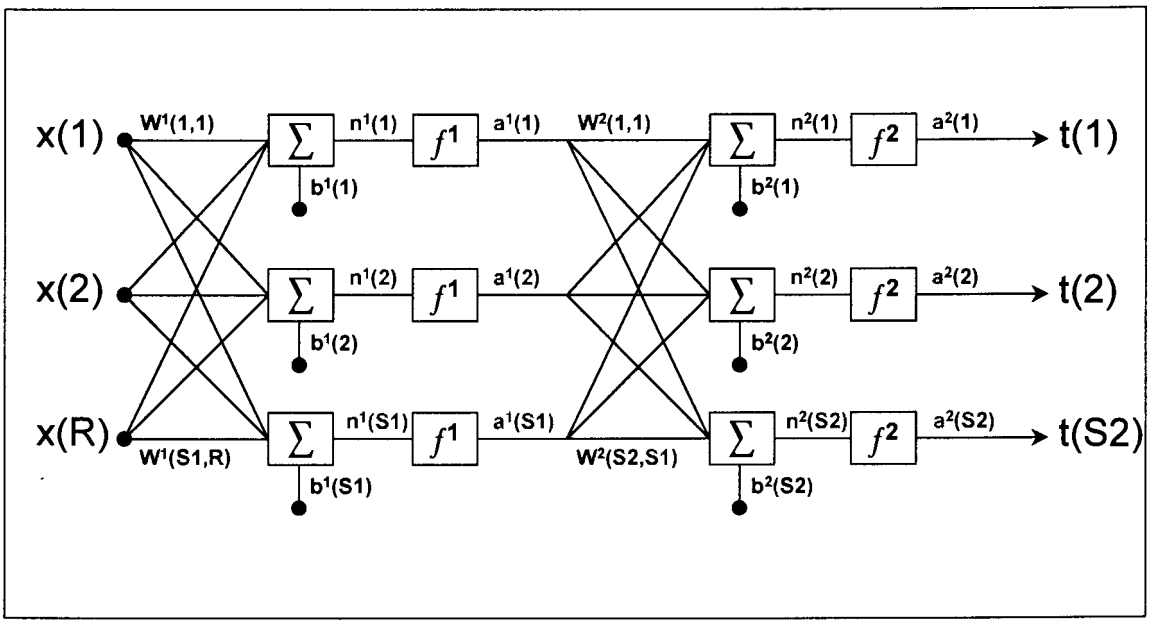

Figure 8.2. two-layer feedforward network

Consider a multilayer feedforward network, such as the two-layer network of Figure 8.2. The net input to unit i in layer $k+1$ is

$$n^{k+1}(i) = \sum_{j=1}^{Sk} w^{k+1}(i,j)a^k(j) + b^{k+1}(i) \tag{8.1}$$

The output of unit i will be

$$a^{k+1}(i) = f^{k+1}(n^{k+1}(i)) \tag{8.2}$$

For an M layer network the system equations in matrix form are given by

$$\mathbf{a}^0 = \mathbf{x} \tag{8.3}$$

$$\mathbf{a}^{k+1} = \mathbf{f}^{k+1}(W^{k+1}\mathbf{a}^{k+1} + \mathbf{b}^{k+1}) \quad k = 0, 1, \ldots, M-1 \tag{8.4}$$

The task of the network is to learn associations between a specific set of input-output pairs $\{(x_1,t_1),(x_2,t_2),\ldots,(x_Q,t_Q)\}$.

The performance index for the network is

$$E = \frac{1}{2}\sum_{q=1}^{Q}(\mathbf{t}_q - \mathbf{a}_q^M)^T(\mathbf{t}_q - \mathbf{a}_q^M) = \frac{1}{2}\sum_{q=1}^{Q}\mathbf{e}_q^T\mathbf{e}_q \tag{8.5}$$

where $\mathbf{a}_q^M$ is the output of the network when the qth input, $\mathbf{x}_q$, is presented and $\mathbf{e}_q$ is the error for the qth input. For standard backpropagation algorithm, as in this case, we use an approximate steepest descent rule. The performance index is approximated by

$$\hat{E} = \frac{1}{2}\mathbf{e}_q^T\mathbf{e}_q \tag{8.6}$$

where the total sum of squares is replaced by the squared errors for a single input/output pair. The approximate steepest (gradient) descent algorithm is then

$$\Delta w^k(i,j) = -\alpha\frac{\partial\hat{E}}{\partial w^k(i,j)} \tag{8.7}$$

$$\Delta b^k(i) = -\alpha\frac{\partial\hat{E}}{\partial b^k(i)} \tag{8.8}$$

where α is the learning rate. Define

$$\delta^k(i) \equiv \frac{\partial \hat{E}}{\partial n^k(i)} \tag{8.9}$$

as the sensitivity of the performance index to changes in the net input of unit i in layer k. Now it can be shown, using (1), (6), and (9), that

$$\frac{\partial \hat{E}}{\partial w^k(i,j)} = \frac{\partial \hat{E}}{\partial n^k(i)} \frac{\partial n^k(i)}{\partial w^k(i,j)} = \delta^k(i) a^{k-1}(j) \tag{8.10}$$

$$\frac{\partial \hat{E}}{\partial b^k(i)} = \frac{\partial \hat{E}}{\partial n^k(i)} \frac{\partial n^k(i)}{\partial b^k(i)} = \delta^k(i) \tag{8.11}$$

It can also be shown that the sensitivities satisfy the following recurrence relation

$$\boldsymbol{\delta}^k = \dot{F}^k(\mathbf{n}^k) W^{k+1T} \boldsymbol{\delta}^{k+1} \tag{8.12}$$

where

$$\dot{F}^k(\mathbf{n}^k) = \begin{bmatrix} \dot{f}^k(n^k(1)) & 0 & \cdots & 0 \\ 0 & \dot{f}^k(n^k(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^k(n^k(Sk)) \end{bmatrix} \tag{8.13}$$

and

$$\dot{f}^k(n) = \frac{df^k(n)}{dn} \tag{8.14}$$

The recurrence relation is initialized at the final layer

$$\boldsymbol{\delta}^M = -F^M(\mathbf{n}^M)(\mathbf{t}_q - \mathbf{a}_q) \tag{8.15}$$

The overall learning algorithm now proceeds as follows:

1. Propagate the input forward using equations 8.3 and 8.4;
2. Propagate the sensitivities back using equations 8.15 and 8.12;
3. Update the weights and offsets using equations 8.7, 8.8, 8.10, and 8.11.

There are two different ways in which this gradient descent algorithm can be implemented: incremental mode and batch mode. In the batch mode which is applied here, all of the inputs are applied to the network before the weights are updated.

The training function has one learning parameter associated with it - the learning rate α shown in equations 8.7 and 8.8. With standard steepest descent as applied here, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. The larger the learning rate, the bigger the step. If the learning rate is made too large the algorithm may oscillate and become unstable. If the learning rate is set too small, the algorithm will take a long time to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

8.2.2. BACKPROPAGATION APPLIED TO IT-NETS

Let t_{pk} be the value of the *k*th observed variable in the *p*th training sample and z_{pk} the corresponding IT-net approximation. Then the objective function to be minimized in network training is

$$E = \sum_{p} \sum_{k} (z_{pk} - t_{pk})^2 \tag{8.16}$$

The steepest descent direction for optimizing network inputs x_{pi} is given by

$$\Delta x_{pi} = -\frac{\partial E}{\partial x_{pi}} = \sum_{k} (t_{pk} - z_{pk}) \frac{\partial z_{pk}}{\partial x_{pi}} \tag{8.17}$$

Assuming that input and output nodes use the identity activation function, like in this study, while hidden nodes use a sigmoidal function $(f = \sigma)$, the network output is given by

$$z_{pk} = \sum_{j} w_{kj} \sigma(b_j + \sum_{i} v_{ji} x_{pi}) \tag{8.18}$$

where $\sigma(.)$ is a sigmoidal function, b_j is the bias of the *j*th hidden node, and V_{ji} and W_{kj} are network weights. Hence, the steepest descent direction for training network inputs is

$$\Delta x_{pi} = \sum_{j} v_{ji} \delta_{pj} \tag{8.19}$$

where δ_{pj} is the propagated error at the hidden layer and has been defined as

$$\delta_{pj} = \sigma'(b_j + \sum_i v_{ji} x_{pi})(\sum_k w_{kj}(t_{pk} - z_{pk})) \tag{8.20}$$

Note that the steepest descent direction for training network weights between the input layer and the hidden layer is

$$\Delta v_{ji} = \sum_p x_{pi} \delta_{pj} \tag{8.21}$$

Therefore, the extra computation required for training the inputs is negligible compared with training the rest of the network. In the preceding derivation, it is assumed that only hidden nodes use sigmoidal functions and that input and output nodes are linear since this is the setup applied in this study. The same derivation can be carried out for networks with sigmoidal output and/or input nodes and Equation 8.19 still holds but with different δ_{pj}.

8.3. Levenberg-Marquardt

While backpropagation is a steepest descent algorithm, the Levenberg-Marquardt algorithm is an approximation to Newton's method. The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. The Jacobian matrix can be computed through a standard backpropagation technique (Hagan and Menhaj, 1994) that is much less complex than computing the Hessian matrix. Suppose that we have a function $E(\mathbf{x})$ which we want to minimize with respect to the parameter vector $\mathbf{x}$, then Newton's method would be

$$\Delta \mathbf{x} = -[\nabla^2 E(\mathbf{x})]^{-1} \nabla E(\mathbf{x}) \tag{8.22}$$

where $\nabla^2 E(\mathbf{x})$ is the Hessian matrix and $\nabla E(\mathbf{x})$ is the gradient. If we assume that $E(\mathbf{x})$ is a sum of squares function

$$E(\mathbf{x}) = \sum_{i=1}^{N} e_i^2(\mathbf{x}) \tag{8.23}$$

then it can be shown that

$$\nabla E(\mathbf{x}) = J^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{8.24}$$

$$\nabla^2 E(\mathbf{x}) = J^T(\mathbf{x})J(\mathbf{x}) + S(\mathbf{x}) \tag{8.25}$$

where $J(\mathbf{x})$ is the Jacobian matrix, which contains first derivatives of the network errors with respect to the weights and biases, and $\mathbf{e}$ is a vector of network errors.

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_1(\mathbf{x})}{\partial x_1} & \frac{\partial e_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial e_2(\mathbf{x})}{\partial x_1} & \frac{\partial e_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\mathbf{x})}{\partial x_1} & \frac{\partial e_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{8.26}$$

and

$$S(\mathbf{x}) = \sum_{i=1}^{N} e_i(\mathbf{x}) \nabla^2 e_i(\mathbf{x}) \tag{8.27}$$

For the Gauss-Newton method it is assumed that $S(\mathbf{x}) \approx 0$, and the updated equation 8.22 becomes

$$\Delta \mathbf{x} = [J^T(\mathbf{x}) J(\mathbf{x})]^{-1} J^T(\mathbf{x}) \mathbf{e}(\mathbf{x})] \tag{8.28}$$

The Levenberg-Marquardt modification to the Gauss-Newton method is

$$\Delta \mathbf{x} = [J^T(\mathbf{x}) J(\mathbf{x}) + \mu I]^{-1} J^T(\mathbf{x}) \mathbf{e}(\mathbf{x}) \tag{8.29}$$

The parameter μ is multiplied by some factor (β) whenever a step would result in an increased $E(\mathbf{x})$. When a step reduces $E(\mathbf{x})$, μ is divided by some factor ζ. In this case a value of $\mu = 0.001$ was selected as an initial value, with $\zeta = 0.1$ and $\beta = 10$. Thus, it is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm. Notice that when μ is large the algorithm becomes steepest descent (with step $1/\mu$), while for small μ the algorithm becomes Gauss-Newton. The Levenberg-Marquardt algorithm can be considered a trust-region modification to Gauss-Newton.

The key step in this algorithm is the computation of the Jacobian matrix. For the neural network mapping problem the terms in the Jacobian matrix can be computed by a simple modification to the backpropagation algorithm. The performance index for the mapping problem is given by equation 8.16. It is easy to see that this is equivalent in form to equation 8.23, where

$$x = [w^1(1,1) w^1(1,2) \ldots w^1(S1,R) b^1(1) \ldots b^1(S1) w^2(1,1) \ldots b^M(SM)]^T \tag{8.30}$$

and $N = Q \times SM$

Standard backpropagation calculates terms like

$$\frac{\partial \hat{E}}{\partial w^k(i,j)} = \frac{\partial \sum_{m=1}^{SM} e_q^2(m)}{\partial w^k(i,j)} \tag{8.31}$$

For the elements of the Jacobian matrix that are needed for the Marquardt algorithm we need to calculate terms like

$$\frac{\partial e_q(m)}{\partial w^k(i,j)} \tag{8.32}$$

These terms can be calculated using the standard backpropagation algorithm with one modification at the final layer

$$\Delta^M = -\dot{F}^M(\mathbf{n}^M) \tag{8.33}$$

Note that each column of the matrix in equation 8.33 is a sensitivity vector which must be backpropagated through the network to produce one row of the Jacobian.

The Marquardt modification to the backpropagation algorithm thus proceeds as follows:

1. Present all inputs to the network and compute the corresponding network outputs (using equations 8.3 and 8.4), and errors $\mathbf{e}_q = \mathbf{t}_q - \mathbf{a}_q^M$. Compute the sum of squares of errors over all inputs ($E(\mathbf{x})$).

2. Compute the Jacobian matrix (using equations 8.33, 8.12, 8.10, 8.11, and 8.26).

3. Solve equation 8.29 to obtain $\Delta\mathbf{x}$.

4. Recompute the sum of squares of errors using $\mathbf{x} + \Delta\mathbf{x}$. If this new sum of squares is smaller than that computed in step 1, then reduce μ by β, let $\mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$, and go back to step 1. If the sum of squares is not reduced, then increase μ by β and go back to step 3.

5. The algorithm is assumed to have converged when the norm of the gradient (equation 8.24) is less than some predetermined value, or when the sum of squares has been reduced to some error goal.

The training parameters as used in Matlab for the Levenberg-Marquardt training algorithm are:

- Maximum number of epochs to train (1000);
- Epochs between showing progress or updating display (25);
- Performance goal which is the sum-squared error goal (0.02);
- Minimum performance gradient (1e-6);
- Maximum validation failures (10);
- An initial value for μ (0.001);
- A value by which μ is multiplied whenever the performance function is reduced by a step (10);
- A value by which μ is multiplied whenever a step would increase the performance function (0.1);
- A maximum value of μ so that if μ becomes larger than this maximum value, the algorithm is stopped (1e10);
- Learning rate for input training (0.1).

The values in brackets are the default values.

8.4. Concept of Input Training

Instead of training a whole three-hidden-layer autoassociative network, only its demapping subnet can be trained. Training such a subnet is meaningful and can be done by extending the backpropagation algorithm, and in this case also combining it with the Levenberg-Marquardt training algorithm.

The difference between training a demapping subnet and training an ordinary feedforward network is that the inputs to the subnet are not given. Not only the internal network parameters but also the input values need to be changed to reproduce the given data as accurately as possible. When network inputs are adjusted, each output sample should be uniquely associated with one input vector. Figure 8.3 shows a 2-4-5 input training network with input adjustment used for reducing the dimensionality of a data set from five to two. Each input vector $(x_{p1}\ \ x_{p2})^T$ is adjusted to minimize only the error of its corresponding output vector $(z_{p1}\ \ z_{p2}\ \ldots\ z_{p5})^T$ while internal network parameters are trained using all output samples.

After the demapping subnet and its inputs are properly trained, we obtain a reduced matrix and a demapping model in the form of a neural network. Thus all requirements for data dimensionality reduction can be fulfilled through training a single-hidden-layer network and its input simultaneously. The concept of input training (IT) gives an alternative to the autoassociative network architecture for reducing data dimensionality. It is this architecture that is referred to as an IT-net. Two characteristics are basic for an IT-net: the input layer has fewer nodes than any other layer, and inputs are adjusted according to corresponding outputs.

Note that the term input in the context of input training slightly differs from what is used for traditional neural networks, where inputs are always given. It is not unusual, however, to adjust inputs to a model while its parameters are being modified to minimize the output error. Examples of this model-fitting strategy include the polynomial PCA and factor analysis (FA). Input training is an application of the same strategy in neural networks. Training an IT-net with one input node and no hidden layer is equivalent to PCA.

IT-nets are basically feedforward networks. With one hidden layer of sigmoidal nodes, a feedforward network can approximate any nonlinear function to an arbitrary accuracy given sufficient hidden nodes (Cybenco, 1989). Let $\phi_k(\lambda_1, \ldots, \lambda_f)$, $k = 1, \ldots, n$, denote nonlinear mappings by an IT-net. When the output error for a given data vector, $(t_{p1}, \ldots, t_{pn})^T$, is minimized at an input vector, $(x_{p1}, \ldots, x_{pf})^T$, we have:

$$\left[\frac{\partial}{\partial \lambda_i}\sum_k (\phi_k - t_{pk})^2\right]_{\lambda_i = x_{pi}} = 0, \quad i = 1, \ldots, f \tag{8.34}$$

which can be rearranged into

$$\sum_k (z_{pk} - t_{pk})\left[\frac{\partial \phi_k}{\partial \lambda_i}\right]_{\lambda_i = x_{pi}} = 0, \quad i = 1, \ldots, f \tag{8.35}$$

where $z_{pk} = \phi_k(x_{p1}, \ldots, x_{pf})$.

If the IT-net has exactly one input node, the functions $\phi_k(\lambda)$, $k = 1, \ldots, n$, represents a smooth curve in n-dimensional space. Equation 8.35 indicates that the vector of output errors, $z_p - t_p$, is orthogonal to the tangent of the curve at $\lambda = x_p$ when the sum of the square errors is minimized through input training. Therefore, the result of training a single-input-node IT-net with a hidden layer of sigmoidal nodes is equivalent to a principal curve as defined by Hastie as Stuetzle (1989). Similarly, training a two-input-node IT-net will result in a principal surface, which is a vector of continuous functions

driven by two parameters and minimizes the orthogonal deviation of the data from the surface (Hastie and Stuetzle, 1989).

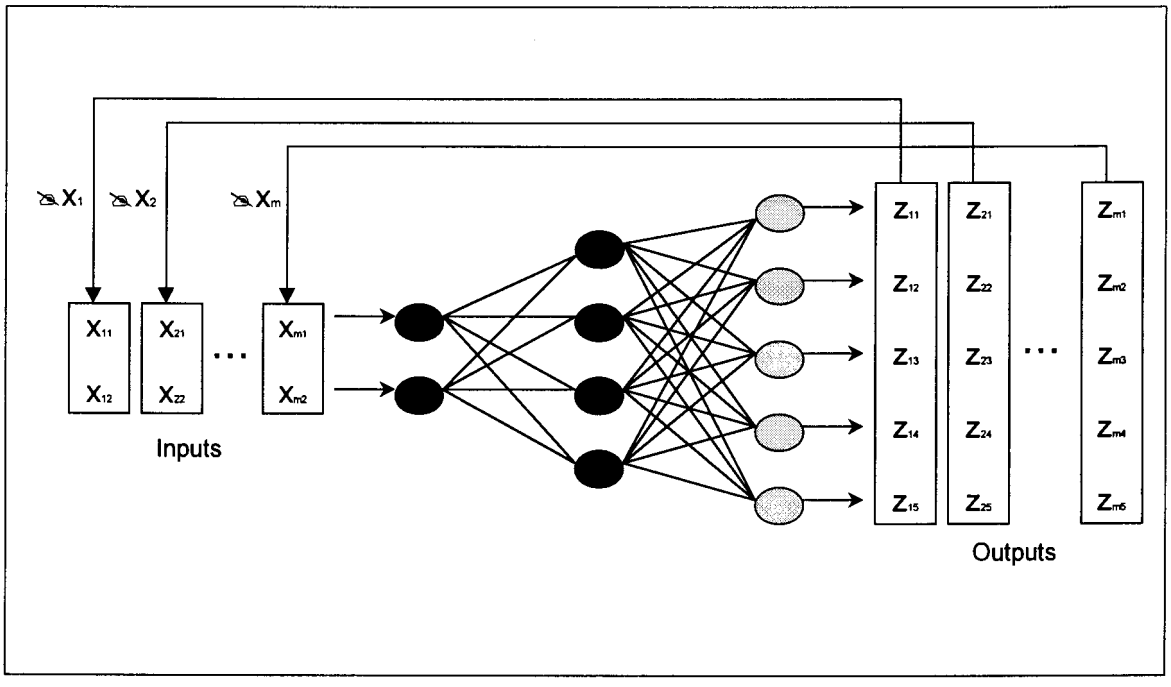

Figure 8.3. Concept of input training

8.5. Training IT-Nets

For IT-nets with hidden layers, a direct iterative procedure for network inputs is not available. The backpropagation and Levenberg-Marquardt training method discussed in Section 8.2 and 8.3 are applied to network inputs. Similar to network weights, network inputs are modified using errors backpropagated from the output layer. Thus, the steepest descent direction for minimizing the output errors through adjustment of network inputs is used.

8.6. Input Training

For input training we start with the Levenberg-Marquardt algorithm and proceed to the end. When step 5 from section 8.4 is completed, we jump to the general backpropagation algorithm with a view adjustments. We first calculate $\mathbf{g}_k$, the gradient, and multiply it by α_k, the learning rate, according to equation 16. Since this is

equivalent to δ_{pj}, and the new weights have been calculated from the LM-algorithm, we can implement equations 8.17 and 8.18 to update the inputs.

When starting the training procedure from the IT-Net interface (Figure 9.1 discussed in Chapter 9), the training status will be displayed every *show* iterations of the algorithm. The other parameters determine when the training is stopped. The training will stop if the number of iterations exceeds the number of *epochs*, if the performance function drops below *goal*, if the magnitude of the gradient is less than the minimum gradient, or if the training time is longer than *time* seconds.

8.7. Testing and Using IT-Nets

A trained IT-net can be tested through cross validation. In this sense, testing and using an IT-net involve the same computing task. We will need to describe only testing. Because network inputs are unknown for testing samples, the appropriate way to test a trained network is to adjust network inputs while freezing all internal network parameters (weights and biases). That is, testing an IT-net still requires a searching procedure that optimizes each input pattern to yield a good approximation for its corresponding output sample.

Note that optimization of inputs for testing is much less time-consuming than training a whole IT-net. First, testing can be done for each individual sample and it involves much fewer searching variables than training the whole network. In addition, since the inputs of training data are available, we can apply a nearest neighbor algorithm to obtain good initial guesses for the inputs of testing samples. Finally, replacing the small fixed learning rate with a 1-D search significantly improves the speed of optimizing network inputs.

As an alternative to the preceding testing method, after training an IT-net we might proceed to train another network that maps the observed data to the reduced data. An autoassociative network could then be constructed by combining the mapping network and the IT-net. The major motivation for doing this is that using a trained autoassociative network would then be as simple as feedforward calculation. However, the result of this method of construction of an autoassociative network is often disappointing due to the following factors:

(1) Training errors are introduced twice

(2) Two network training rounds are independent of each other and there is no effective mechanism to ensure the quality of the whole autoassociative network.

In this study this alternative mapping method proved to be effective and gave acceptable results.

Chapter 9 NONLINEAR PCA

9.1. Introduction

In practice, both linear and non-linear correlations exist between process variables. The presence of linear correlations within the data impacts upon the non-linear PCA algorithm in terms of its ability to extract a parsimonious description of the underlying characteristics of the process. The presence of linear correlations between the variables results in the need for higher dimensionality to define the underlying non-linear structure.

Linear PCA is effectively a rotation. Since the rotation is carried out in linear space, the non-linear structures will be encapsulated within the principal component scores sub-space if the dimensionality of the sub-space is sufficiently large. Thus, the process of extracting linear and non-linear correlations from the data can be performed separately. In this chapter a non-linear PCA approach is proposed which combines the advantages of both linear PCA and the previous non-linear PCA algorithms. This is done by extending the principles of Input Training as discussed in Chapter 8 to nonlinear principal component analysis.

9.2. Nonlinear Principal Component Analysis

So how does NLPCA differ from LPCA? As we have seen, linear principal components analysis (LPCA) is a projection-based statistical tool traditionally used for dimensionality reduction. Consider an m-dimensional data set $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$ whose variance-covariance matrix has eigenvalue-eigenvector pairs $(\lambda_1, \mathbf{p}_1), (\lambda_2, \mathbf{p}_2), \ldots, (\lambda_m, \mathbf{p}_m)$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq 0$. The linear principal component decomposition of $\mathbf{X}$ can be represented as:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E} = \sum_{i=1}^{l} \mathbf{t}_i \mathbf{p}_i + \mathbf{E} \quad (l < m)$$

where $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_l]$ is defined to be the matrix of principal component scores, $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_l]$ is the matrix of principal component loadings and $\mathbf{E}$ is the residual matrix in the sense of minimum Euclidean norm. Non-linear PCA is an extension of linear PCA. Whilst PCA identifies linear correlations between process variables, non-linear PCA can extract both linear (second-order statistics) and non-linear (higher-order statistics) correlations. This generalisation is achieved by projecting the process

variables down onto curves or surfaces instead of lines or planes using the same objective function, i.e. minimising the mean-square error $\mathbf{E}\{\| \mathbf{X} - \mathbf{X} \|^2\}$. The data $\mathbf{X}$ can be expressed in terms of k non-linear principal components, where $k << m$,

$$\mathbf{X} = F(\mathbf{T}) + \mathbf{E} \tag{9.1}$$

where $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k]$ is the matrix of non-linear principal component scores, F is the non-linear function equivalent to the loadings in linear PCA and $\mathbf{E}$ is the matrix of residuals.

The IT-net discussed in Chapter 8 can effectively represent non-linear systems which include both additive and multiplicative types of non-linearities through the simultaneous calculation of the latent variables. A potential disadvantage of this approach is that the complexity of training the IT-net increases exponentially with the dimensionality of the training dataset (Bakshi, 1998).

Two steps form the basis of the algorithm. In the first step, linear PCA is applied to the original observations as in Chapter 7, resulting in a new set of uncorrelated ordinates. By retaining sufficient data variability, the underlying non-linear structure is not compromised and only those linear principal components associated with noise are discarded. By reducing the dimensionality of the data, the non-linear structure becomes more apparent. In the second step the IT-net is used to extract the latent non-linear structure in the transformed dataset. To overcome the problem of local minima, the training of the IT-net is repeated several times with different initial conditions for each network architecture to ensure that the global minimum is found. The proposed non-linear PCA representation can be defined as follows:

$$\mathbf{X} = F(\mathbf{T}) \cdot \mathbf{P}^T + \mathbf{E} \tag{9.2}$$

where $\mathbf{T}(k < l < m)$ is the matrix of non-linear principal component scores which are identified from the input layer of the IT-net at the second stage and $P(m \times l)$ is the matrix of linear principal loadings calculated from the first stage of the algorithm. $F(.)$ represents the input-training network function and $\mathbf{E}$ is the matrix of model residuals. Equation 9.2 gives the non-linear principal scores $\mathbf{T}$. However, for a new observation, to calculate the corresponding non-linear principal component score requires the implementation of a time consuming non-linear optimisation algorithm and is thus not appropriate for on-line application. An alternative and more straightforward approach is to develop a model between the process observations $\mathbf{X}$ and the non-linear principal scores $\mathbf{T}$ using a feed-forward neural network. Finally, the function relating the non-linear principal component scores to the process observations is defined as

$$\mathbf{T} = G(\mathbf{X} \cdot \mathbf{P}) \tag{9.3}$$

The function $G(\cdot)$ is the feed-forward neural network model with the linear PCA transformed data set $\mathbf{X} \cdot \mathbf{P}$ as the input layer and the non-linear principal scores $\mathbf{T}$ as the output layer. Once models based on Equations 9.2 and 9.3 are built, the task of developing an on-line monitoring and fault detection scheme is straightforward requiring minimal computational effort.

9.3. Application

9.3.1. SOFTWARE SETUP

The NLPCA setup interface can be accessed by using the Next button from the LPCA interface or can accessed directly from the Main interface. In Figure 9.1 to Figure 9.4 the NLMSPCA model is created. The use of the interfaces are straightforward.

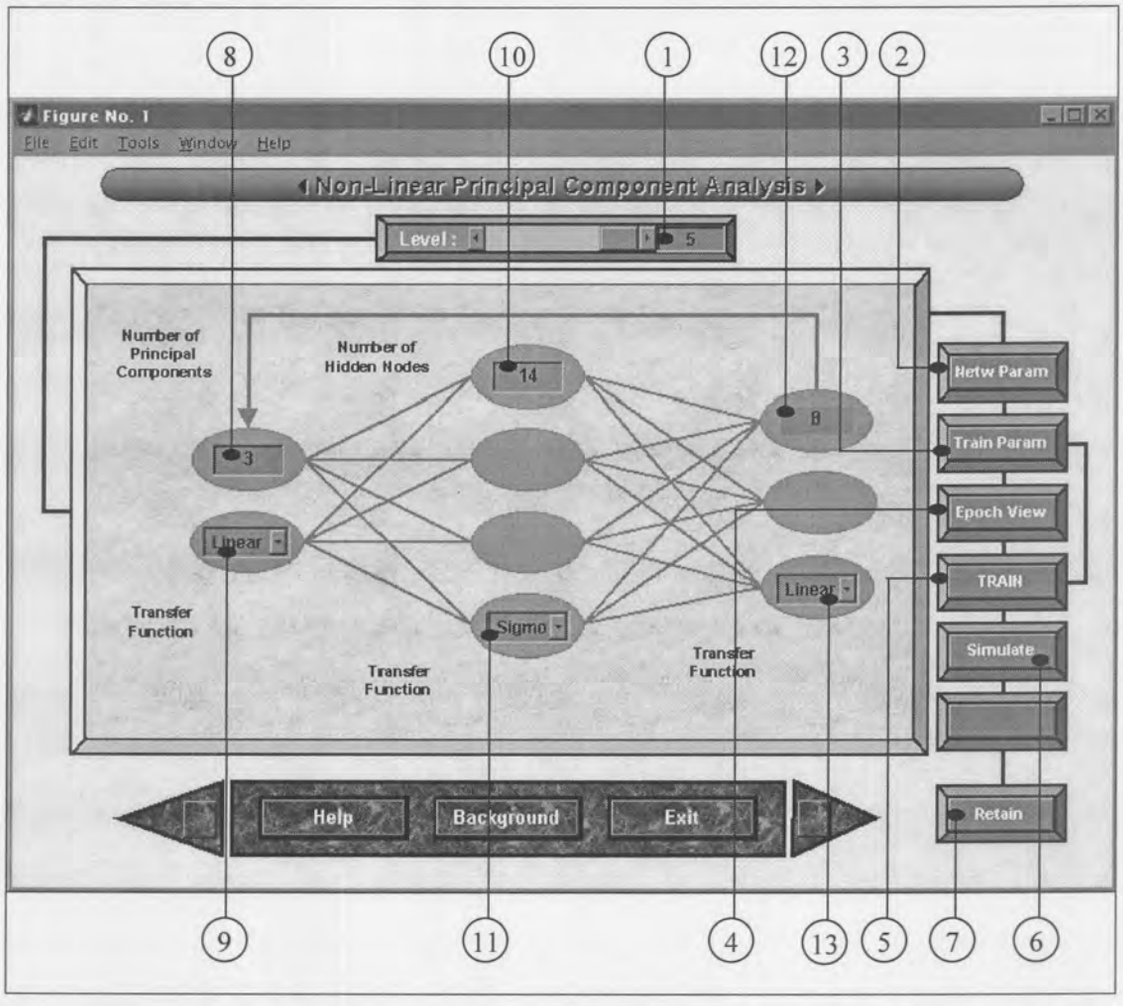

Figure 9.1. IT-Net parameter setup interface

Figure 9.1 Tags:

1. Dataset used to construct the NLMSPCA model. In this case it will either be dataset 1 or dataset 2.

2. Display the Network Parameter window (Figure 9.1). This allows the user to set up the input training neural network structure.

3. Display the Training Parameters window (Figure 9.2). This allows the user to set up the training parameters for the input training neural network.

4. Display the Epoch Viewer. This allows the user to view the Mean Squared Error (MSE) graphically as the training of the IT-Net progresses in order to get an idea of how fast and how well the training is progressing.

5. Train the IT-Net. After setting up Figure 9.1 and Figure 9.2 and after selecting Figure 9.3 this button is used to start the training and the progress will be displayed via Figure 9.3. Depending on the network structure and the error goal this can take a long time to complete. Training of the IT-Net will stop as soon as one of the stopping criteria is reached or when terminated by the user.

6. Simulate the network. This will display Figure 9.4.

7. Retain the network. If the user is satisfied with the trained network the network structure and parameters can be saved to the database.

8. Number of nonlinear principal components. By default the same number of nonlinear as linear principal components are chosen. However, the user has the option to change the number of nonlinear principal components to retain. Recall that the linear principal component scores are used as initial input to the IT-Net.

9. Input node/level transfer function. By default a linear transfer function is used and should remain so.

10. Number of hidden nodes. This is actually the only network structure parameter, except for perhaps the number of nonlinear principal components, that will be changed. There is no set rule for the optimum number of hidden nodes and therefore the user will have to play around with this parameter together with the training parameters until acceptable results are obtained. The more hidden nodes, the longer the training will take.

11. Transfer function of the hidden node. By default the sigmoidal transfer function is used and it should not be necessary to use any other transfer function. Other options include the linear and tangent transfer functions.

12. Number of original variables. This will be automatically displayed according to the number of variables in the database. Recall that the process variables are used as output for the IT-Net.

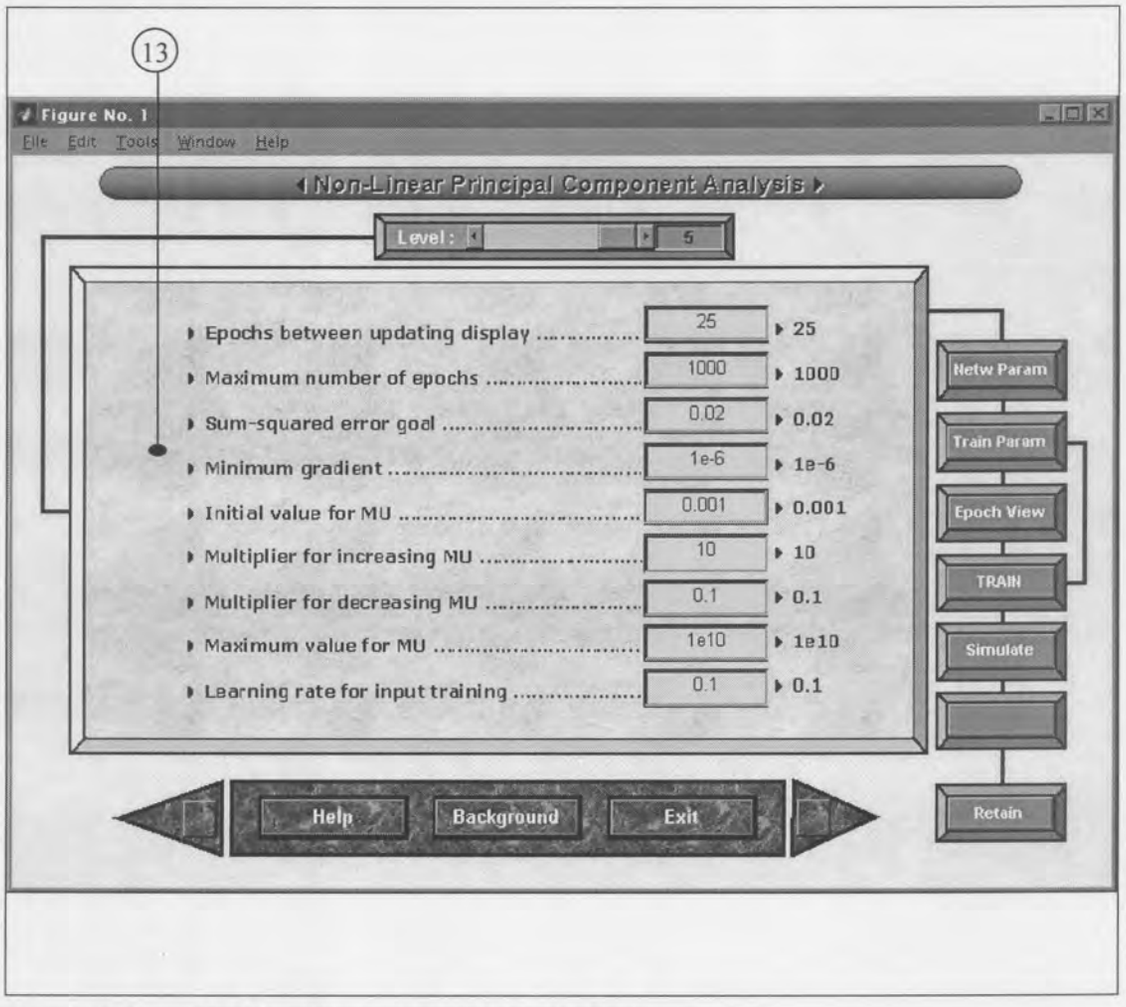

Figure 9.2. IT-Net training parameter interface

Figure 9.2 displays default values for the training parameters next to the edit boxes. These values have proven to work in most cases. If the error goal is not met after training the first parameter that can be altered is to increase the maximum number of epochs that will also increase the training time. After the training has stopped, a message will be displayed in the Matlab workspace indicating the stopping criteria used to stop the training. If, for example, training was terminated because the maximum value for mu was exceeded and the user is not yet satisfied with the performance, this will be an indication that the maximum value of mu needs to be increased. At this stage no other value except 0.1 for the learning rate will work.

Figure 9.2 Tags:

13. Training parameters display window.

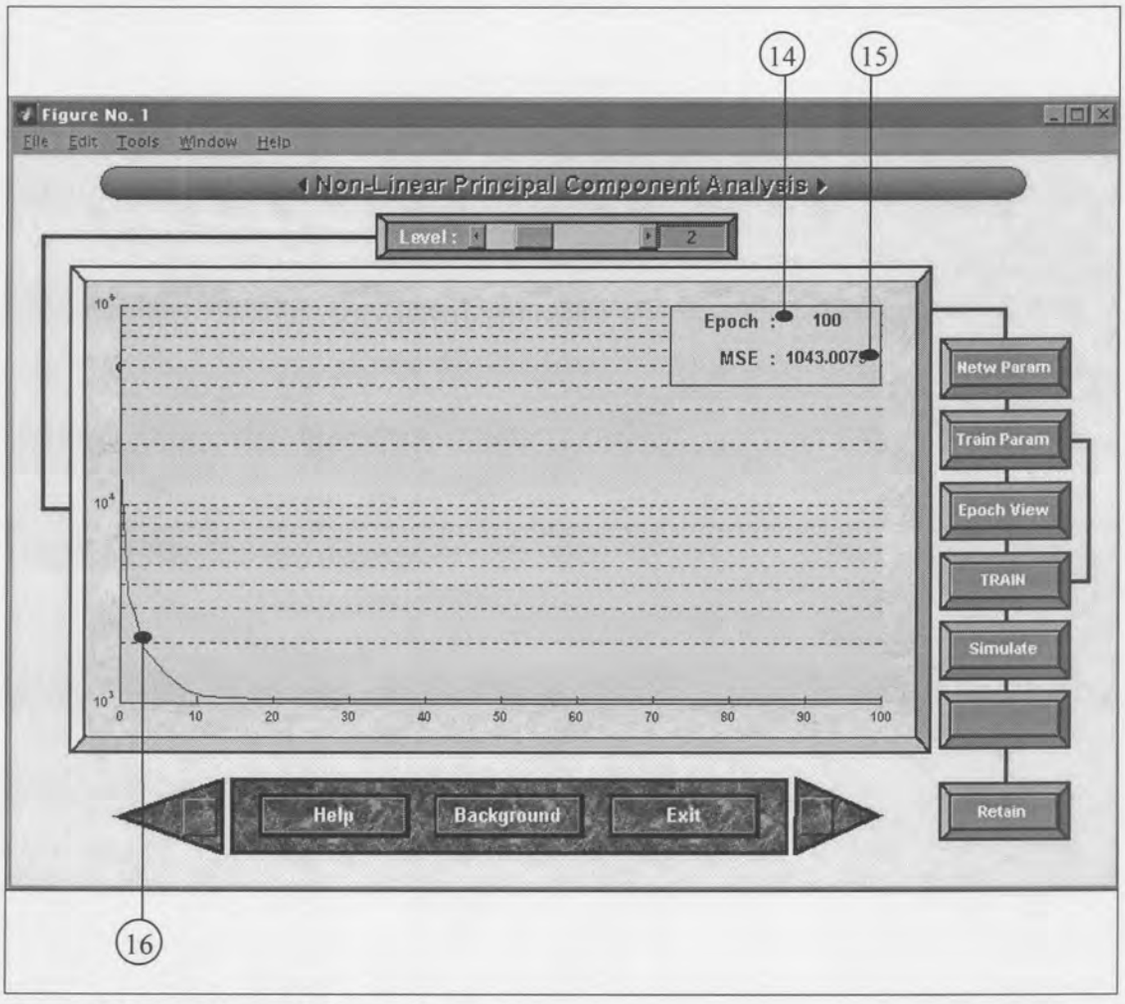

Figure 9.3. Epoch view interface

Figure 9.3 Tags:

14. Number of epochs completed so far.

15. The mean squared error of the network so far. This is also displayed via the graph. This window is updated on intervals specified in the Training Parameters window (Figure 9.3, Epochs between updating display).

16. MSE plot.

After the training has stopped the interface in Figure 9.4 can be used to simulate the network. The newly generated nonlinear principal components are used as input and the original process variables are used as output. This allows the user to visually inspect the performance of the network.

Figure 9.4 Tags:

17. Data name. By default the original data representing normal operation in the database is used. The user can however test the network performance by using

new unseen data. The name of the variable containing this data can be entered here. The variable must reside in the Matlab workspace with each column representing a separate process variable.

18. Process Variable number. Each process variable can be viewed separately.

19. Original versus simulated variable. The original process variable and the same process variable generated from the IT-Net through simulation are displayed to give a visual comparison between the original and trained data. This normally gives an indication of the level of network performance, especially in the case of new (validation) data.

20. Error plot between the original and simulated process variable.

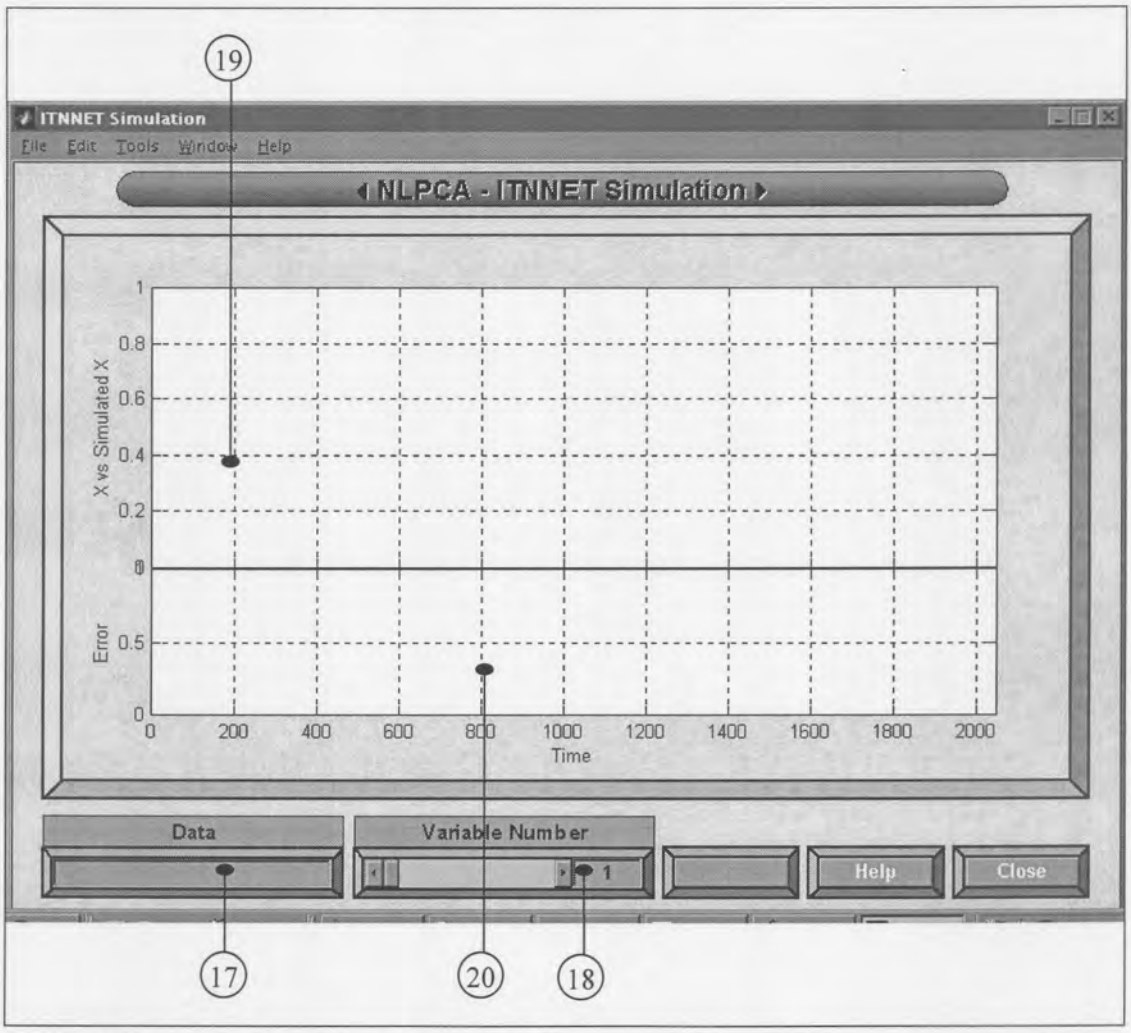

Figure 9.4. IT-Net simulation interface

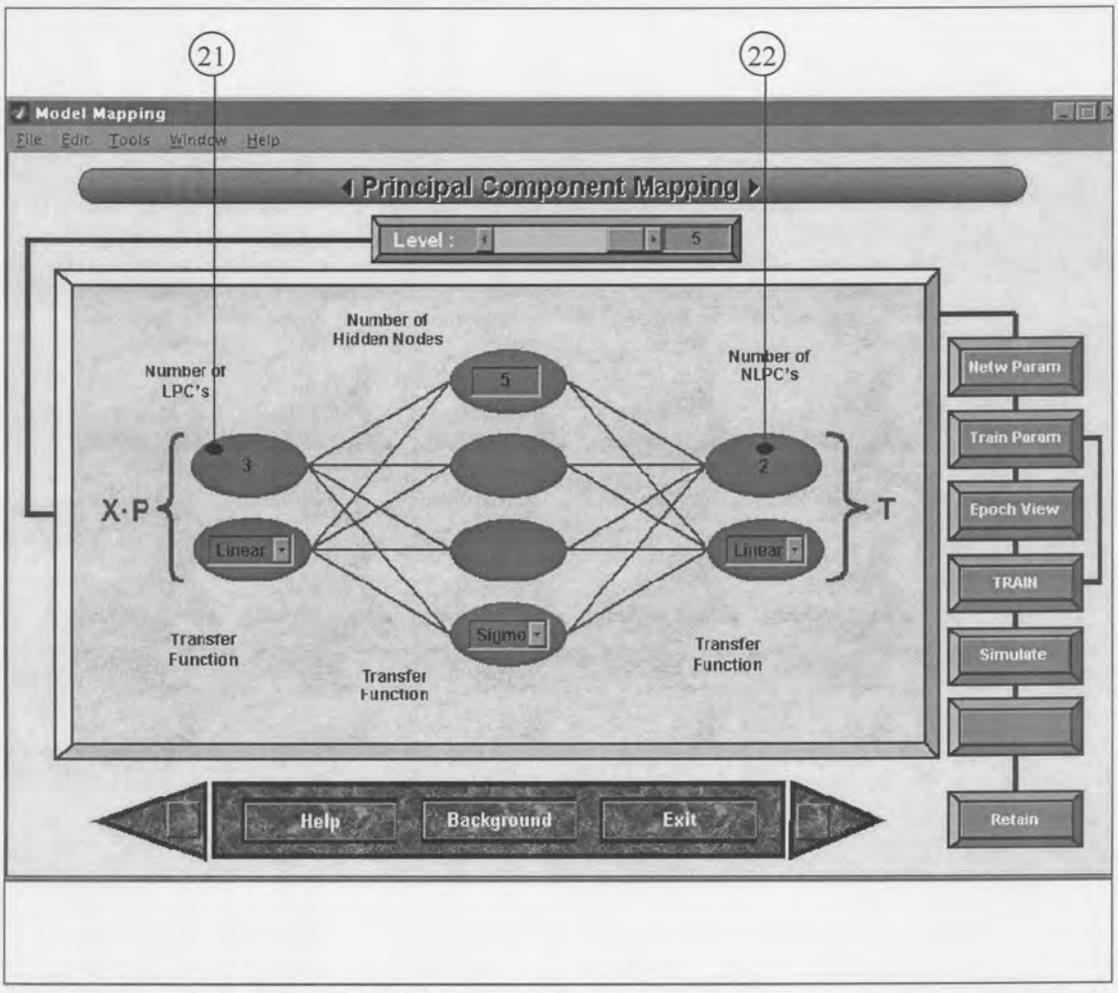

Figure 9.5. Neural network mapping interface

After the IT-Net has been trained the interface in Figure 9.5 can be used to generate a mapping model that generates a model between the linear and nonlinear principal component scores that can be connected to the IT-Net model. This interface is displayed by using the next button from the NLPCA interface (Figure 9.1) or can be accessed directly from the Main interface. Its setup and use are identical to Figure 9.1 to Figure 9.4 except for its inputs and outputs and for the fact that this is just a normal feedforward neural network that is trained using the Levenberg Marquardt training algorithm.

Figure 9.5 Tags:

21. Number of linear principal components. This is automatically displayed according to the number of linear principal components recorded in the database.

22. Number of nonlinear principal components. This is automatically displayed according to the number of nonlinear principal components recorded in the database.

9.3.2. EXPERIMENTAL

Non-linear PCA was applied to dataset 1 and dataset 2 using the IT-net methodology to extract the nonlinear correlations between the process variables and to build the nonlinear multiscale PCA representation.

Here, as in the case of linear principal component analysis, three principal components were retained in the model for dataset 1 capturing 97.72% of the data variability and again four for dataset 2 capturing 98.53% of the data variability making this a 3-4 principal component-model. The discarded principal components were attributed to process noise. Different network initialisation and neural network structures were used to train the neural networks to address the local minima problem. Finally, an IT-net with a 4-13-40 structure and a 40-18-4 feedforward neural network were built to model the projection of the data onto the nonlinear sub-space and its inverse, respectively for dataset 2 and a 3-12-8 structure and 8-14-3 feedforward neural network for dataset 1. The training parameters are summarised in Table 9.1. The shadowed blocks indicate the criteria on which the training was terminated for each case.

Table 9.1. Training parameters for IT-Net and mapping model

	IT-Net		**Mapping**	
Training parameter	**Dataset 1**	**Dataset 2**	**Dataset 1**	**Dataset 2**
Epochs between updating display	25	25	25	25
Maximum number of Epochs	2000	4000	1000	1000
Sum squared error goal	0.02	0.02	0.01	0.01
Sum squared error achieved	*0.029*	*0.032*	*0.01*	*0.01*
Minimum gradient	1e-6	1e-6	1e-6	1e-6
Initial value for MU	0.001	0.001	0.001	0.001
Multiplier for increasing MU	10	10	10	10
Multiplier for decreasing MU	0.1	0.1	0.1	0.1
Maximum value of MU	1e10	1e10	1e10	1e10
Learning rate for Input Training	0.1	0.1	-	-

Using the combined backpropagation and Levenberg-Marquardt training algorithm resulted in an average improved convergence rate of 350%. The results showed that the final nonlinear PCA representation captures 89.5% of the variability in dataset 2 and 93.1% in dataset 1. For the validation data it is 89.9% and 94.4% respectively. Action and warning limits for the bivariate score and SPE plots were then derived.

Chapter 10 PROCESS MONITORING

10.1. Introduction

To be effective, a plant-wide control monitoring and performance assessment system should have the following properties:

(i) automated background operation, including scheduled remote collection of control loop data and data integrity checks,

(ii) theoretically sound, efficient, and automated computational procedures,

(iii) decision support (for example, problem reporting by exception),

(iv) technical support, and

(v) a suitable user interface.

Together, these properties form the basis of a comprehensive control performance monitoring and assessment system, as shown in Figure 10.1.

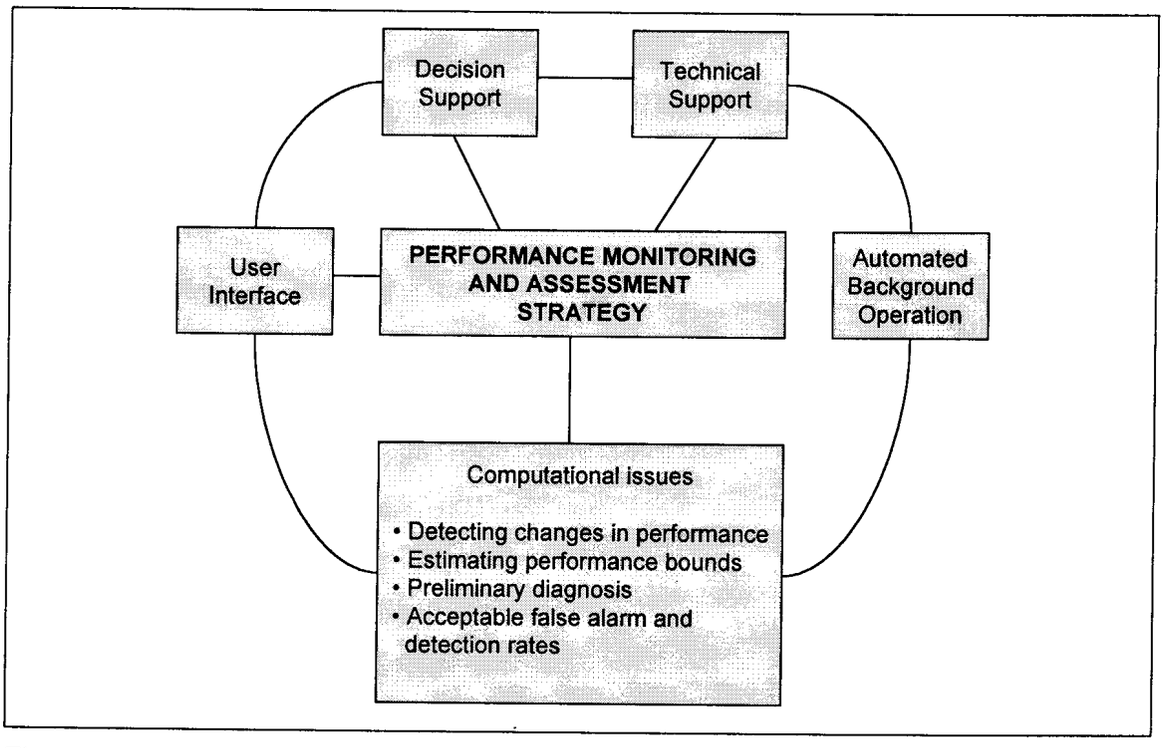

Figure 10.1. Important components of an industrial performance monitoring assessment, and diagnosis strategy.

The main aim here is to concentrate on the computational issues part of the industrial performance monitoring assessment and diagnosis strategy. Complete separate studies can be conducted on the other issues on their own and are therefore outside the scope of this study.

10.2. Interpretation

During the LPCA and NLPCA processes principal scores and loadings are calculated. A subset of the first few scores, $A < N$, provides information in a lower dimensional space, the *score space*, of the behaviour of the process during the period in which the measurements were made. This set of scores and the PCA loadings can be used to determine if the present process operation has changed its behavior relative to the data that were used to define the scores and loadings (Piovoso et al., 1992a).

There are several ways of interpreting the PCA results. Typical monitoring control charts include:

- the Q-statistic, a measure of the model mismatch;
- the Hotelling T^2-statistic, a measure of the fit of new observations to the model space;
- variance plots, a measure of the samples' variability;
- univariate and bivariate principal component score plots, a qualitative representation of the process performance, relative to the calibration model in the model space defined by the calibration model;
- and a time-series plot of the squared prediction error (SPE).

These have been widely used to obtain early warning of the occurrence of nonconforming operation. Once a NLPCA representation has been built for process performance monitoring and fault detection, action and warning limits require to be calculated.

Bivariate score plots and the squared prediction error (SPE) were used in the approach adopted by Dong and McAvoy (1996) for defining the action and warning limits for their non-linear performance monitoring scheme. However, adopting this approach requires that the non-linear scores and residuals follow a multivariate normal distribution.

10.3. Action and warning limits

For multivariate statistical process monitoring by NLMSPCA, the region of normal operation is determined at each scale from data representing normal operation. For new data, an abnormal situation is indicated when the current coefficient violates the detection limits. The actual state of the process is confirmed by checking whether the signal reconstructed from the selected coefficients violates the detection limits of the PCA model for the significant scales. This approach is equivalent to adaptively filtering each value of the scores and residuals by a filter of dyadic length that is best suited for separating the deterministic change from the normal process variation. The detection limits for the scores and residuals also adapt to the nature of the signal.

These detection limits consist of action and warning limits. Warning limits are usually 95% confidence bounds and serve as a warning that the process is approaching an abnormal condition. Action limits are usually 99% confidence bounds indicating that an abnormal operation is occurring and that action needs to be taken.

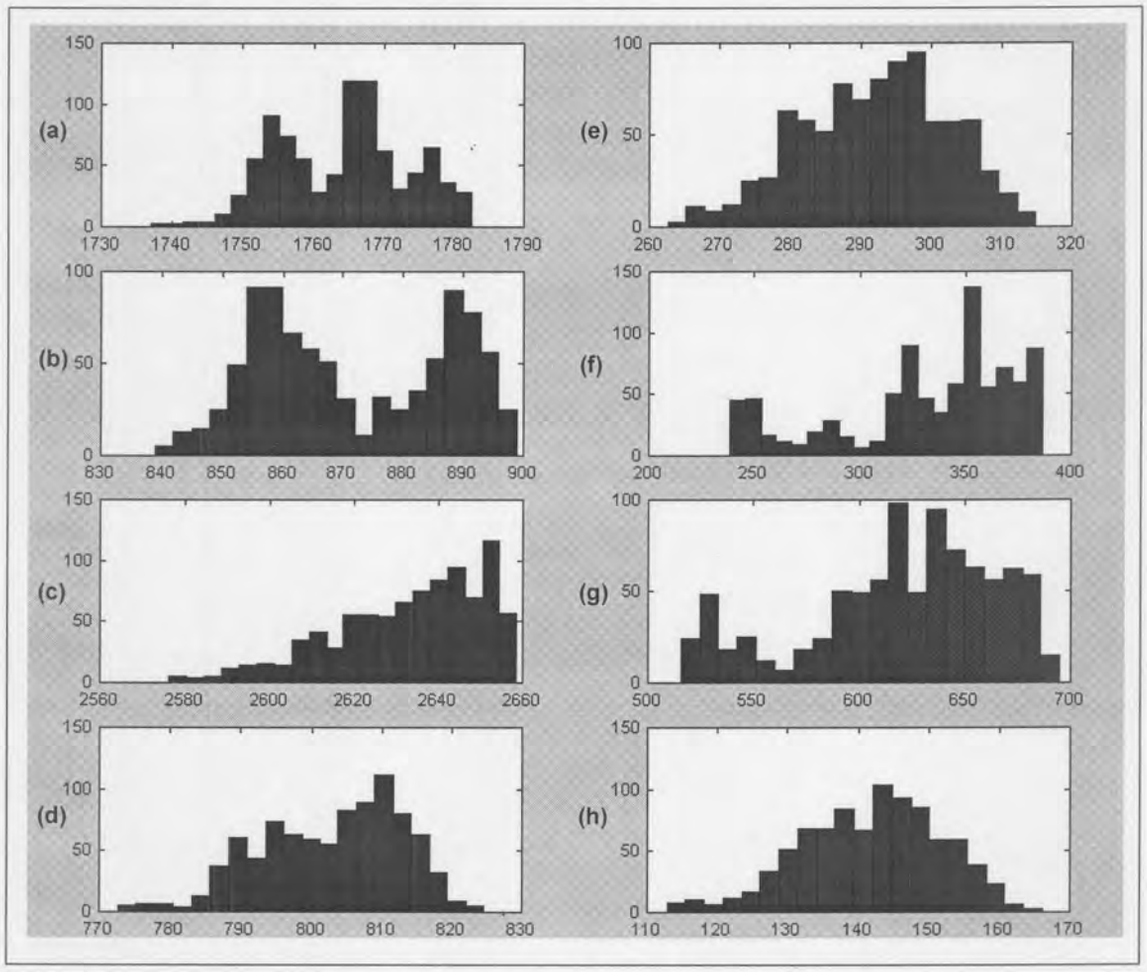

Figure 10.2. Histogram plots of the eight variables ((a)-(f)) used for training.

However, for non-linear PCA the assumption that the non-linear scores and residuals follow a multivariate normal distribution cannot be guaranteed. Although it is possible to define the region of normal operation without any underlying assumption concerning the probability distribution of the measurements, an assumption is still required to apply hypothesis-based statistical tests to identify when the process is moving outside the action or warning limits. The assumption of a normal distribution is incorrect as illustrated in Figure 10.2, which gives the histogram plots of the eight variables representing normal operation used in the application. Except for the last variable they do not closely resemble a normal distribution. Therefore, using this assumption will introduce a significant error. An alternative approach that effectively deals will this problem is introduced in Section 10.4.

10.4. Non-parametric bounds

An alternative approach to defining the action and warning limits is based upon non-parametric density estimation. Non-parametric bounds for process performance monitoring have previously been developed (Martin and Morris, 1996), using kernel estimation. Density estimation is the construction of an estimate of the density function from the observed data. A multivariate product kernel estimator can be constructed based upon the m-dimensional random samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ from a density f (Scott, 1992):

$$f(x) = \frac{1}{nh^m} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{10.1}$$

where h is the window width, also called the smoothing parameter or bandwidth, and K is the kernel function which satisfies the conditions

$$K(\mathbf{x}) \geq 0, \quad and \quad \int_{R^m} K(\mathbf{x}) d\mathbf{x} = 1 \tag{10.2}$$

The shape of the density estimate is determined by the choice of the smoothing parameter *h*, and to a lesser extent by the choice of the kernel (Scott, 1992; Bowman, 1984). An automatic procedure for determining the optimal window width was used, the minimisation of the mean integrated squared error cross validation (Bowman, 1984). When using the density estimation-based approach to define the action and warning limits for the monitoring charts, the density function of the non-linear principal component scores and squared prediction error are calculated for the nominal (reference) data. Depending upon the confidence level required, 95% for the warning limits and 99% for the action limits, the contour or value can be calculated to define the control limits for the non-linear principal component scores plot and the SPE. Action and warning limits based upon kernel density estimation are theoretically more appropriate in the development of a non-linear PCA

monitoring scheme. However, if the underlying distribution is normal, similar results to those obtained from the conventional approaches are obtained.

10.5. Detection limit adjustment for on-line monitoring

For on-line monitoring, the MSPCA algorithm is applied to measurements in a moving window of dyadic length. The use of a moving window makes the on-line wavelet decomposition algorithm equivalent to wavelet decomposition without downsampling, causing a signal of length n to result in a total of $n(L+1)$ coefficients, where L is the depth of the wavelet decomposition. This increase in the number of coefficients requires on-line monitoring by MSPCA to increase the detection limits at each scale to maintain the desired confidence limit for the reconstructed signal. For example, for normally distributed uncorrelated measurements in a window length of 128, approximately one sample will lie outside the 99% confidence limits. The off-line wavelets transform will also result in 128 uncorrelated coefficients, and approximately one coefficient will violate the 99% limits. In contrast, the on-line wavelet transform of these data will result in 128 coefficients at each scale, and approximately one coefficient will violate the 99% detection limits at each scale. Thus, if the signal is decomposed to four detail signals and one scaled signal, that is, for $L=4$ as in this case, application of the 99% confidence limit at each scale will result in an effective confidence of only 95% for the reconstructed signal, since the coefficients violating the detection limits at each scale need not be at the same location. Consequently, the detection limits at each scale for on-line monitoring by MSPCA need to be adjusted to account for the overcompleteness of the on-line wavelet decomposition by the following equation:

$$C_L = 100 - \frac{1}{L+1}(100 - C) \tag{10.3}$$

where C is the desired overall confidence limit, C_L is the adjusted confidence limit at each scale at present, and L is the number of scales to which the signal is decomposed, resulting in L detail signals and one scaled signal.

Another effect of the on-line wavelet decomposition approach is that the wavelet coefficients retain more of the autocorrelation in the signal due to the use of overlapping windows for the decomposition. Fortunately, the performance of monitoring by NLMSPCA is not adversely affected by the autocorrelated coefficients in adjacent windows, since the confidence limits at each scale are increased by equation 10.3, and even relatively small deterministic features are captured by large wavelet coefficients.

10.6. Residual analysis

Another interesting property of PCA is the fact that the equation

$$\mathbf{z} = \mathbf{U}'[\mathbf{x} - \bar{\mathbf{x}}] \tag{10.4a}$$

may be inverted so that the original variables may be stated as a function of the principal components, viz.,

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{U}\mathbf{z} \tag{10.4b}$$

because $\mathbf{U}$ is orthonormal and hence $\mathbf{U}^{-1} = \mathbf{U}'$. This means that, given the z-scores, the values of the original variables may be uniquely determined. However, $\mathbf{x}$ will be determined exactly only if all the pc's are used. If $k < p$ pc's are used, only an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ will be produced, viz.,

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{U}\mathbf{z} \tag{10.5}$$

where $\mathbf{U}$ is now $p \times k$ and z is $k \times 1$. Equation 10.4b can be rewritten as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{U}\mathbf{z} + (\mathbf{x} - \hat{\mathbf{x}}) \tag{10.6}$$

a type of expression similar to those often found in other linear models. In this case, the first term on the right-hand side of the equation represents the contribution of the multivariate mean, the second term represents the contribution due to the pc's, and the final term represents the amount that is unexplained by the pc model – the residual.

Gnanadesikan and Kettenring (1972) divided multivariate analysis into

1. The analysis of internal structure.

2. The analysis of superimposed or extraneous structure.

There are outliers associated with each of these and it is important to keep their identities distinct. (Hawkins refers to them as Type A and Type B outliers.)

The Type A outlier refers to a general outlier from the distribution form one wishes to assume. Usually this assumption will be multivariate normal and these outliers will be detected by large values of T^2 and/or large absolute values of the z-scores. The important thing about this type of outlier is that it would be an outlier whether or not PCA has been employed and hence could be picked up by conventional multivariate techniques without using PCA. However, the use of PCA might well enhance the chance of detecting it as well as diagnosing what the problem might be.

Here we will be concerned with the Type B outlier, the third term in equation 10.6, which is an indication that a particular observation vector cannot be adequately characterized by the subset of pc's one chose to use. This result can occur either because too few pc's were retained to produce a good model or because the observation is, truly, an outlier from the model. It is also possible in repetitive operations, such as quality control, that the underlying covariance structure and its associated vector space may change with time. This would lead to general lack-of-fit by the originally defined pc's.

10.7. Biplots

When most of the variance of the variables is summarized by only two principal components, then we can express the results as a biplot. Although biplots are originally meant two-dimensional plots, they may be used for any number of dimensions. For two-dimensional plots it means that for the singular value decomposition we are summarizing a lot of the information in only two dimensions. Two-dimensional plots are very popular because they are easy to work with but should always include some statement with regard to the proportion of the total variability explained by the first two characteristic roots. Unless this quantity is sufficiently large, the interpretation of the plot is suspect.

Although biplots will be used, they will only be used indirectly. Another technique will be introduced for viewing the information contained in a biplot.

Since the data contained in samples when the process was not operating normally, applying PCA as a simple outlier detector revealed which data could be classified as such. Score biplots of the first few principal components can be used to visually detect these outliers (Piovoso et al., 1992).

10.8. Hotelling's T^2 statistic: An overall measure of variability

Hotelling's T^2-statistic measures unusual variability within the calibration model space. That is, if the calibration model data represent process operation at one operating condition, and the process has shifted to a different one, then the T^2-statistic will show that data at this operating condition cannot be classified with the calibration data. The T^2-statistic is proportional to the sum of the squares of the scores on each of the principal components (Piovoso et al., 1992).

The T^2 statistic can be applied (Johnson and Wichen, 1992) to the principal component scores to calculate the control limits. It is based upon the assumption that the limits of the

control charts are calculated assuming that the original data $\mathbf{X}$ follows a multivariate normal distribution. Under these assumptions, the principal component scores and residuals obtained from linear PCA will also exhibit normality since PCA is a linear transformation and a linear combination of a normal distribution is itself normally distributed.

The T^2-quantity can be calculated as follows:

$$T^2 = \mathbf{y}'\mathbf{y} \tag{10.7}$$

which is a quantity indicating the overall conformance of an individual observation vector to its mean or an established standard. This quantity, due to Hotelling (1931), is a multivariate generalization of the Student t-test and does give a single answer to the question: "Is the process in control?"

The original form of T^2 is

$$T^2 = [\mathbf{x} - \bar{\mathbf{x}}]'\mathbf{S}^{-1}[\mathbf{x} - \bar{\mathbf{x}}] \tag{10.8}$$

which does not use PCA and is a statistic often used in multivariate quality control. Substituting $\mathbf{S}^{-1} = \mathbf{W}\mathbf{W}'$ and $y_i = \mathbf{w}_i'[\mathbf{x} - \bar{\mathbf{x}}]$ in equation10.8 results in

$$\begin{aligned} T^2 &= [\mathbf{x} - \bar{\mathbf{x}}]'\mathbf{S}^{-1}[\mathbf{x} - \bar{\mathbf{x}}] \\ &= [\mathbf{x} - \bar{\mathbf{x}}]'\mathbf{W}\mathbf{W}'[\mathbf{x} - \bar{\mathbf{x}}] = \mathbf{y}'\mathbf{y} \end{aligned} \tag{10.9}$$

so equations 10.7 and 10.8 are equivalent. The important thing about T^2 is that it not only fulfills Condition 1 for a proper multivariate quality control procedure as listed in Chapter 7, Section 7.9.2, but Conditions 2 and 3 as well. The only advantage of equations 10.7 over 10.8 is that if $\mathbf{W}$ has to be obtained, the computations are considerably easier as there is no matrix to invert. In fact, $\mathbf{y}'\mathbf{y}$ is merely the sum of squares of the principal components scaled in this manner ($T^2 = y_1^2 + y_2^2$ for the two-variable case) and demonstrates another advantage in using $\mathbf{W}$-vectors. If one uses $\mathbf{U}$-vectors, the computations become, essentially, a weighted sum of squares:

$$T^2 = \mathbf{z}'\mathbf{L}^{-1}\mathbf{z} \tag{10.10}$$

and the use of $\mathbf{V}$-vectors would produce a similar expression.

Few books include tables for the distribution of T^2 because it is directly related to the F-distribution by the relationship

$$T^2_{p,n,\alpha} = \frac{p(n-1)}{n-p} F_{p,n-p,\alpha} \tag{10.11}$$

In this example, $p = 8$, $n = 900$, $F_{8,892,0.05} = 3{,}8056$, so

$$T^2_{8,900,0.05} = 8{,}187$$

An observation vector that produces a value of T^2 greater than 8,187 will be out of control on the chart.

However, the traditional approach to calculating action and warning limits for multivariate process performance monitoring based on Hotelling's T^2, is inappropriate in the non-linear case since a non-linear mapping does not necessarily guarantee that the generated data will follow a normal distribution as discussed earlier. This problem was addressed by calculating the control limits using the non-parametric technique of kernel density estimation in Section 10.4. This approach has the advantage that no a priori assumption of normality is required.

An alternative method of plotting T^2 is to represent it in histogram form, each value of T^2 being subdivided into squares of the y-scores. This is sometimes referred to as a stacked bar-graph, and indicates the nature of the cause of any out-of-control situations. However, the ordinate scale would have to be arithmetic rather than logarithmic.

Process monitoring can also be referred to as a form of multivariate quality control. The procedure or guidelines for monitoring a multivariate process using PCA is as follows:

1. For each observation vector, obtain the y-scores of the principal components and from these, compute T^2. If this is in control, continue processing.

2. If T^2 is out of control, examine the y-scores. As the pc's are uncorrelated, it would be hoped that they would provide some insight into the nature of the out-of-control condition and may lead to the examination of particular original observations.

The important thing is that T^2 is examined first and the other information is examined only if T^2 is out of control. This will take care of the first three conditions listed in Section 7.9.2 and, hopefully, the second step will handle the fourth condition as well. Even if T^2 remains in control, the pc data may still be useful in detecting trends that will ultimately lead to an out-of-control condition.

10.9. The Q-statistic

The residual term of Equation 10.6 can be tested by means of the sum of squares of the residuals:

$$Q = (\mathbf{x} - \hat{\mathbf{x}})'(\mathbf{x} - \hat{\mathbf{x}}) \tag{10.12}$$

This represents the sum of squares of the distance of $\mathbf{x} - \hat{\mathbf{x}}$ from the k-dimensional space that the PCA model defines.

To obtain the upper limit for Q, let:

$$\theta_1 = \sum_{i=k+1}^{p} l_i$$

$$\theta_2 = \sum_{i=k+1}^{p} l_i^2$$

$$\theta_3 = \sum_{i=k+1}^{p} l_i^3$$

and

$$h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2}$$

Then the quantity

$$c = \theta_1 \frac{\left[\left(\frac{Q}{\theta_1}\right)^h - \frac{\theta_2 h_0(h_0 - 1)}{\theta_1^2} - 1\right]}{\sqrt{2\theta_2 h_0^2}} \tag{10.13}$$

is approximately normally distributed with zero mean and unit variance (Jackson and Mudholkar, 1979). Conversely, the critical value for Q is

$$Q_\alpha = \theta_1 \left[\frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + \frac{\theta_2 h_0(h_0 - 1)}{\theta_1^2} + 1\right]^{1/h_0} \tag{10.14}$$

where c_α is the normal deviate cutting of an area of α under the upper tail of the distribution if h_0 is positive and under the lower tail if h_0 is negative. This distribution

holds whether or not all of the significant components are used or even if some nonsignificant ones are employed.

In Section 2.6, it was suggested that the last two characteristic roots in the example were not significantly different from each other and hence the last two pc's were deleted. If only the first two pc's were retained, what would be the limit for Q? The last two roots were 29.33 and 16.41. From these, $\theta_1 = 45.74$, $\theta_2 = 1129.54$, $\theta_3 = 29650.12$, and from these $h_0 = 0.291$. Letting $\alpha = 0.05$, the limit for Q, using equation 10.14 is

$$Q_{0.05} = 45.74\left[\frac{(1.645)\sqrt{(2)(1129.54)(0.291)^2}}{45.74} + \frac{(1129.54)(0.291)(-0.709)}{(45.74)^2} + 1\right]^{1/0.291}$$
$$= 140.45$$

Values of Q higher than this are an indication that a data vector cannot be adequately represented by a two-component model.

10.10. Contribution plots

When a new observation moves outside the control limits, it is assumed that an unusual process event or equipment malfunction has occurred and operator personnel need a tool to identify which variables, or combination of variables, are responsible for, or indicative of, changes in the process. One approach is through the implementation of a process variable contribution plot (Miller et al., 1993). Consequently, for the identification of variables indicative of non-conforming operation, differential contribution plots based upon model residuals and non-linear principal component scores are used.

By comparing the contribution plot of a sample taken from the calibration set with one that is outside the confidence limits, differences in the expected variables' magnitude may provide an indication of which variables have exceeded their expected limits, and a possible compensation to correct the problem.

Contribution plots describe the change in the magnitude of the variables for the new observation relative to the average value calculated from the nominal linear PCA model. It decomposes the scores into their summation operands and graphs them versus the contributing variable. The summation operands are the products of the loadings of variable j and the corresponding value of variable j. A large product associated with a particular variable implies a correspondingly large contribution (Piovoso et al., 1992)

Using a similar argument, a contribution plot can be derived and applied in a non-linear situation. The contribution of the process variables to the SPE can be calculated in a similar way to that for linear PCA. However, since the mapping function between the process variables and their non-linear principal scores is non-linear, the relationship between the variables and the non-linear principal scores is not as straightforward as in the linear case where the scores can be decomposed as a weighted sum of the process measurements. An alternative approach is based upon the assumption that the partial derivative of a function with respect to a specific dimension can indicate the relative influence of the corresponding variable on that function. If the first-order partial derivatives of a multivariate function are known for a specific variable space coordinate, then these derivatives can be used to compare the relative influence of the individual variables on the function at a particular location in variable space. Thus a differential contribution plot which describes the difference between the contribution of the process variables to its non-linear scores can be defined by comparing the influence of the first-order partial derivatives of the non-linear scores to each process variable for a specific sample or time point. The differential contribution plot that indicates the contribution of the process variables at a specific time point $(\mathbf{x}_0)$ to a non-linear score t_i, can then be examined by calculating the individual components of the vector product

$$\mathbf{x}|_{\mathbf{x}=\mathbf{x}_0} \cdot \left.\frac{\partial \mathbf{t}}{\partial \mathbf{x}}\right|_{\mathbf{t}=\mathbf{t}_i} \tag{10.15}$$

where $\partial \mathbf{t} / \partial \mathbf{x}$ is the first-order partial derivative function between $\mathbf{t}$ and $\mathbf{x}$. The relationship between the non-linear principal scores $\mathbf{t}$ to the process variables $\mathbf{x}$ is given in Equation 9.3. This approach is also suitable for linear PCA since linearity can be viewed as a special case of non-linearity. In the linear case, the first-order partial derivatives of $\mathbf{t}$ relative to $\mathbf{x}$ become constant which in practice are the principal component loadings, $\mathbf{P}$. Thus Equation 10.15 can be simplified to $\mathbf{x}|_{\mathbf{x}=\mathbf{x}_0} \cdot \mathbf{p}_i$ which is the same expression as that for the contribution plot to the scores proposed by Miller et al. (1993).

10.11. Bivariate summary plots

Figure 10.3 illustrates the traditional biplots with detection limits for the normal linear case (a) and based on the kernel density estimation (b). The contours represent normal operation detection limits. If an observation moves outside these detection limits it will indicate that an abnormal condition has occurred. With a new observation we are actually just interested in how far from abnormal the condition is. This can be calculated using the methodology illustrated in Figure 10.4.

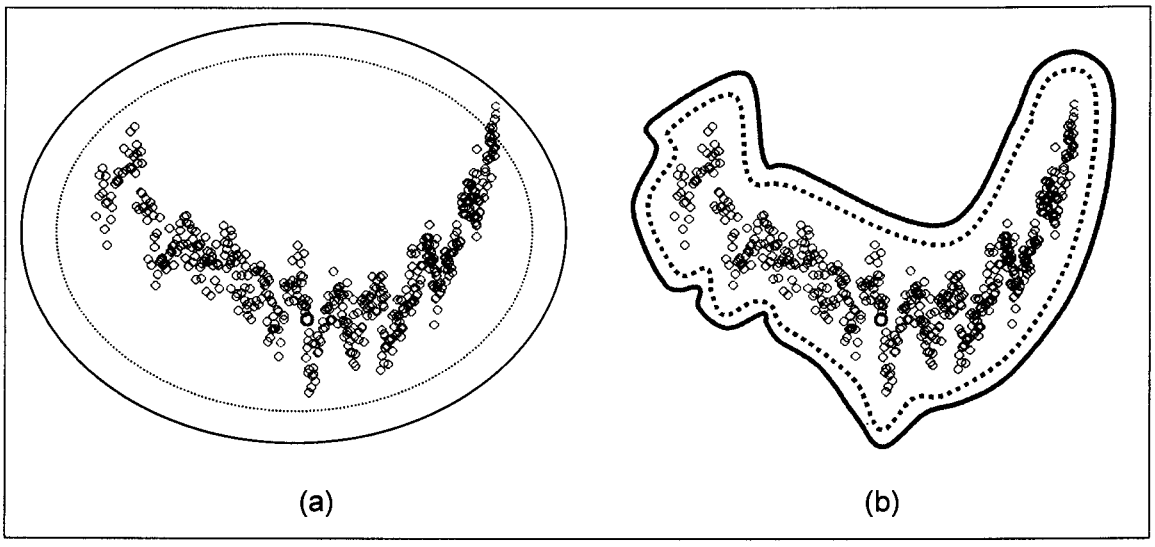

Figure 10.3. Traditional bivariate plot

We are interested in the in the shortest tangent line to the warning limit which can be extended to find the shortest tangent line to the action limit. In Figure 10.4(a) there exist two tangent lines with one (d_2) being the shortest line to any position on the warning limit contour. Figure 10.4(b) contains more than two such tangent lines. Thus, for a new observation we just need to calculate all the possible tangent lines from the observation point to the warning limit contour and select the shortest as an indication of how far the process is from a nonconforming condition. This can be calculated for different biplot combinations and summarized as illustrated in Figure 10.5 where the first bar is based on Figure 10.4(b). The warning limit line forms the baseline. d_m is a non-tangent line and thus will not be considered.

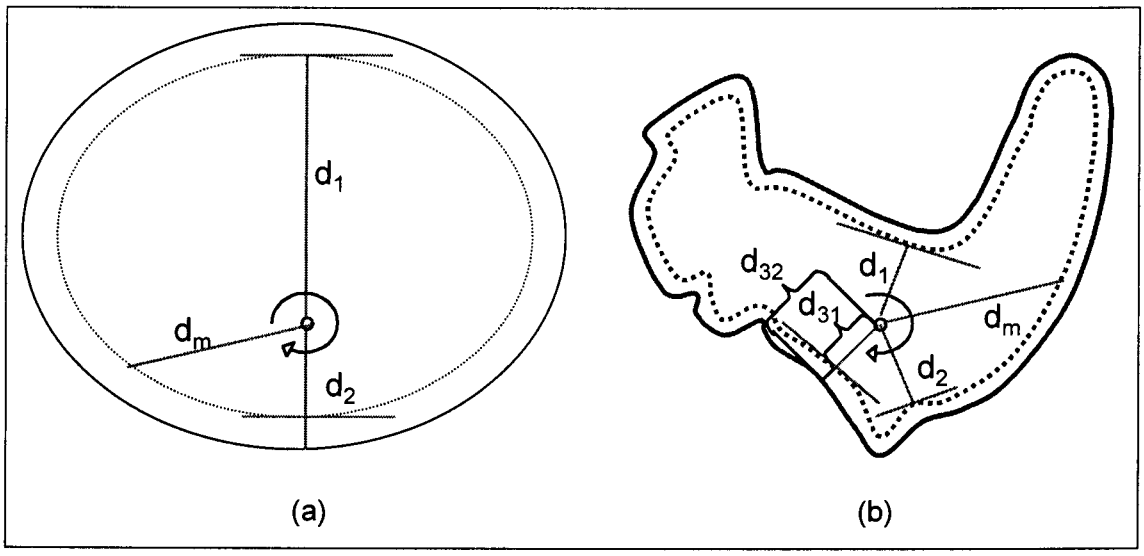

Figure 10.4. Bivariate summary plot calculation

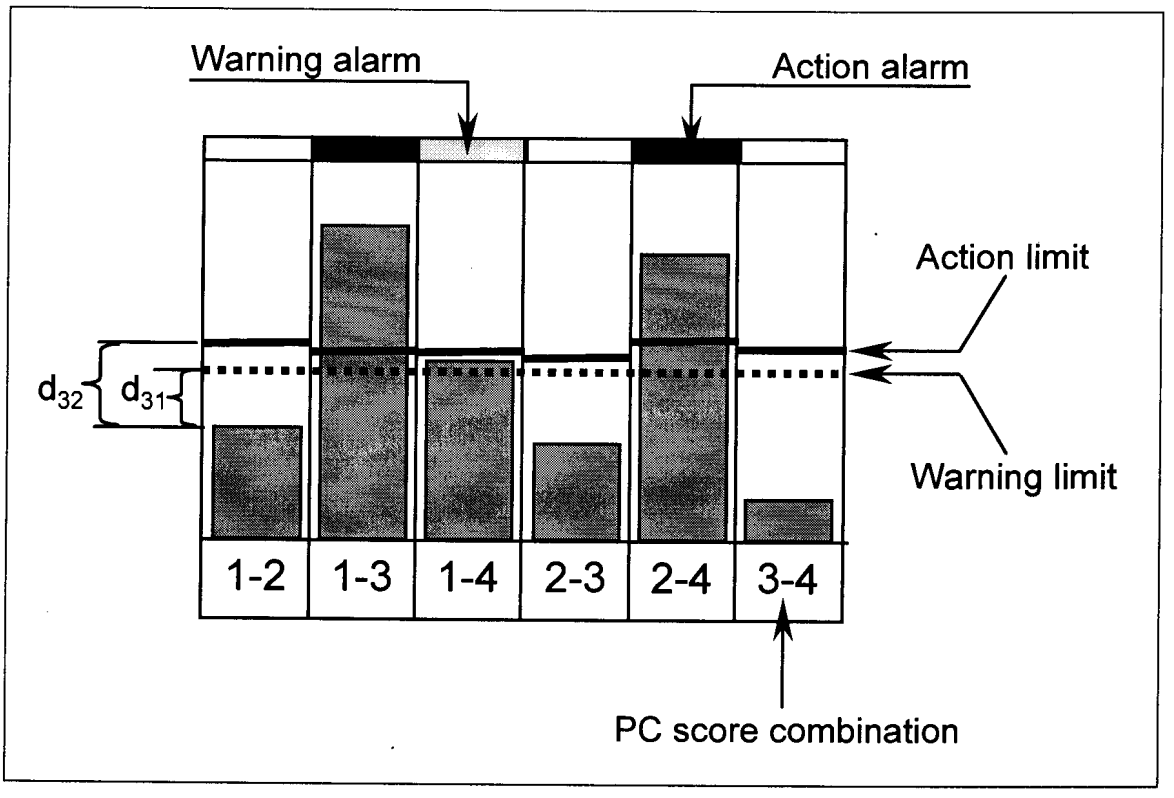

Figure 10.5. Bivariate summary plot for 6 biplots at one time instance

10.12. Application

10.12.1. SOFTWARE SETUP

Figure 10.6 is used to set up the bivariate contour plots. Its sole purpose is to choose the confidence limits for the action and warning limits. It shows the linear and nonlinear limits for comparison. The different combinations of scores can be viewed. The summary bivariate plots are automatically generated from the bivariate plots and thus do not need to be set up separately.

Figure 10.6 Tags:

1. Dataset number slider
2. Dataset number display. In this case one will first set up the bivariate plots for dataset one and then for dataset two.
3. Y-axes principal component number slider.
4. Y-axes principal component number display. For this application one will have a choice between three principal components for dataset one and four for dataset two.

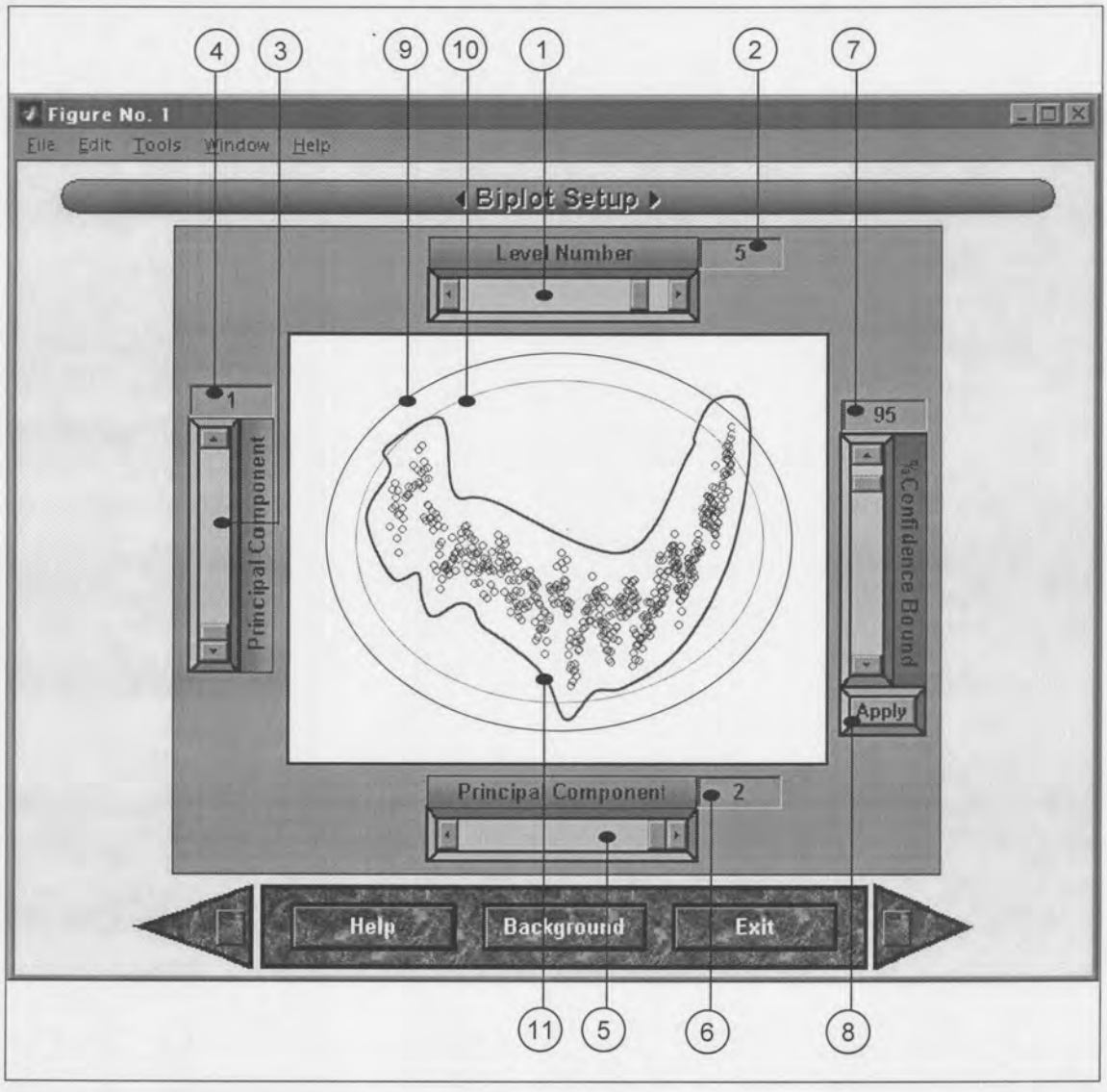

Figure 10.6. Bivariate plot setup interface

5. X-axes principal component number slider.

6. X-axes principal component number display. For this application one will have a choice between three principal components for dataset one and four for dataset two.

7. Confidence limit for the bivariate principal component score plot. As one changes the confidence limit the contour plot on the interface will change accordingly.

8. Application button to save the parameters to the database. The fist one accepted will be the warning limit and the second one will be the action limit. After the second limit has been applied one can advance to the next pair of principal component scores for a new bivariate plot.

9. Action limit for linear case where normality is assumed.

10. Warning limit for linear case where normality is assumed.

11. Action limit for nonlinear case when using density estimation.

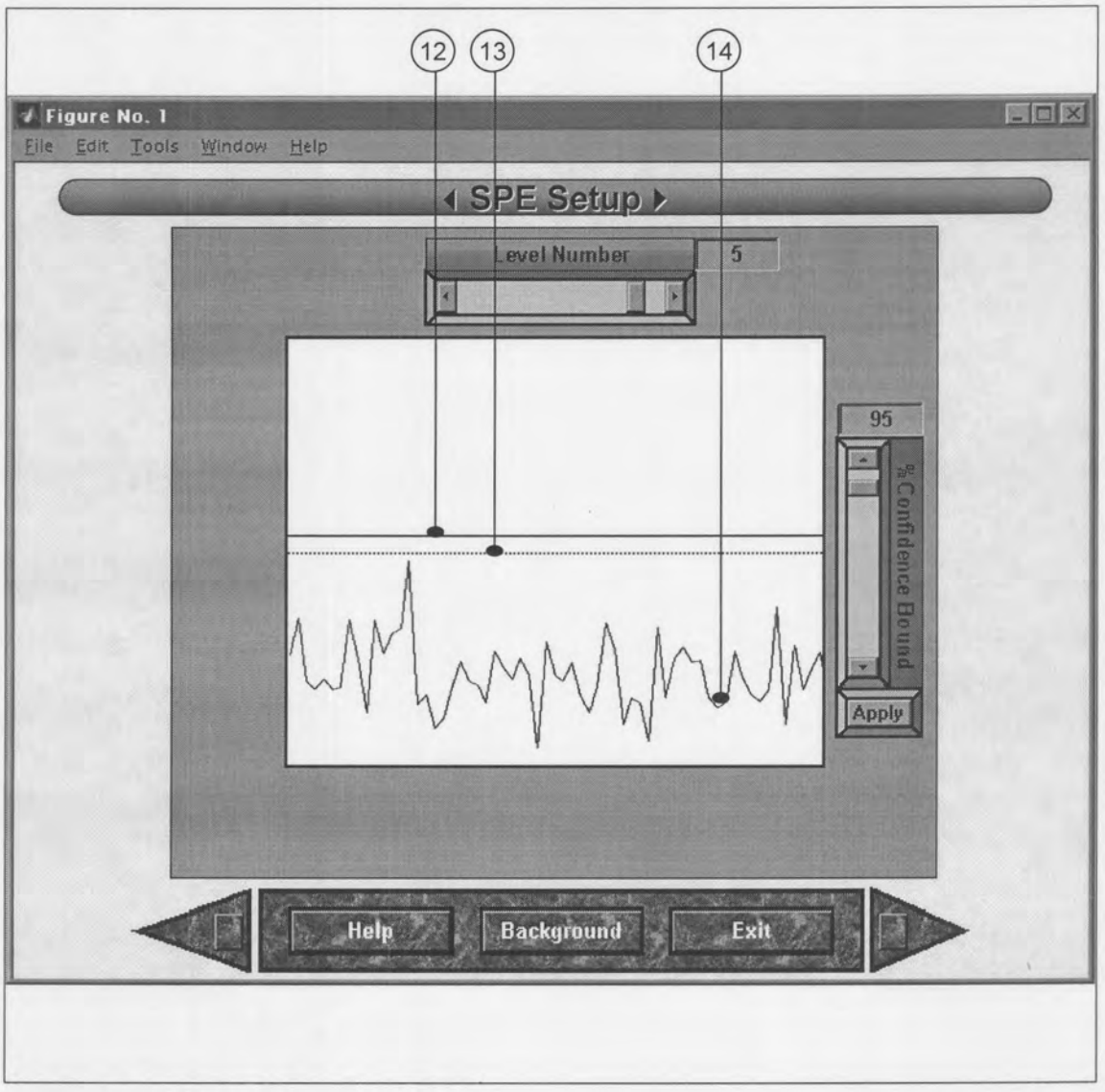

Figure 10.7. SPE Setup interface

Figure 10.7 is used to set up the action and warning limits for the SPE plot and works in a similar way to Figure 10.6.

Figure 10.7 Tags:

12. SPE action limit.

13. SPE warning limit.

14. SPE plot

Up to this point all previous software was just used to set up the NLMCPCA model and is not used again except when changes to the model need to done. All the relevant information is stored in the database. Figure 10.7 is the actual process monitoring interface which is used to monitor new process data. This interface can be accessed by using the *next* button in Figure 10.6 or can be accessed directly via the Main interface.

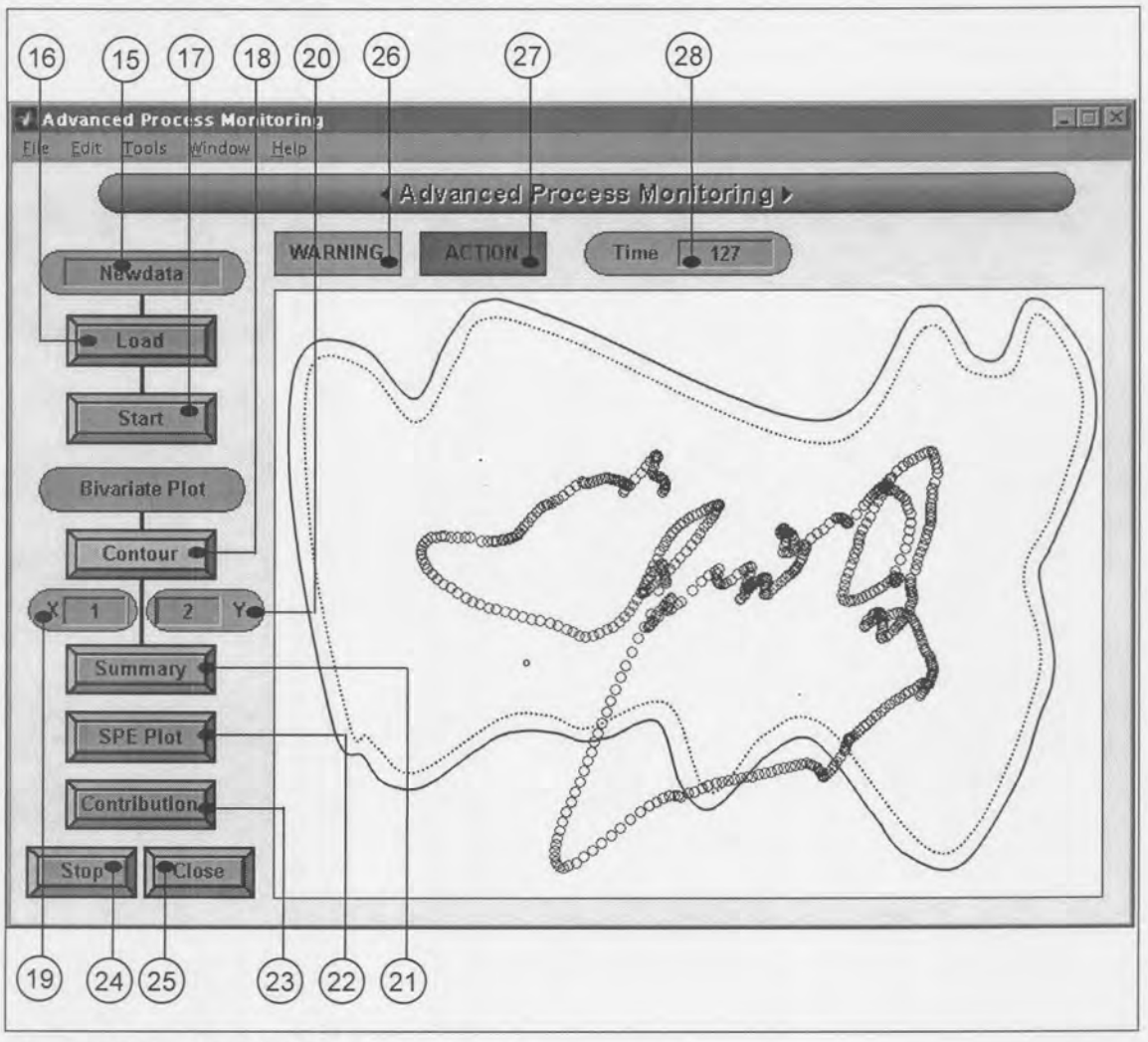

Figure 10.8. NLMSPCA Monitor interface

This interface shown in Figure 10.8 allows the user to choose between the following plots:

- Bivariate contour plot
- Bivariate summary plot
- SPE plot
- Contribution plot

Only one of these plots can be viewed at a time. One would normally use the bivariate summary plot and only view the SPE and contribution plots when an abnormal operation is detected. While viewing the bivariate summary plot, the SPE is also calculated. In an event of an abnormal operation first being detected by the SPE the SPE plot will automatically replace the bivariate summary or contour plot.

Figure 10.8 Tags:

15. Name of the variable containing the new data for investigation purposes. The NLMSPCA model will be applied to this data in order to detect the existence of abnormal behavior in the data. This variable must reside in the matlab workspace and contain the data of each variable in a separate column.

16. Load the data into the NLMSPCA model.

17. Start the NLMSPCA monitoring process using the parameters selected during the setup/training of the NLMSPCA model with normal data.

18. Bivariate contour plot. Only one bivariate contour plot can be plotted at a time. By default, principal component one is plotted versus principal component two. Other combinations can be selected using 5 and 6. If dataset 1 contains four principal components and dataset contains six, principal component one to four will refer to dataset 1 and five to ten to dataset 2.

19. Select principal component number for the x-axes.

20. Select principal component number for the y-axes.

21. Summary bivariate plot. All possible combinations for dataset 1 and dataset 2 are plotted.

22. View the SPE-plot.

23. View the contribution plot.

24. Stop the monitoring process.

25. Close the current window (exit the monitor interface).

26. Warning alarm. This alarm is shown as soon as the warning limits of the bivariate or SPE plots are violated. This alarm will remain for three time intervals before being cleared automatically.

27. Action alarm. This alarm is shown as soon as the action limits of the bivariate or SPE plots are violated. This alarm will remain for three time intervals before being cleared automatically.

28. The current time-interval.

10.12.2. EXPERIMENTAL DATA

First a calibration model was developed based upon 900 one-minute samples taken of the eight monitored variables according to the preceding chapters. To test the new methodology, data representing abnormal operation had to be collected. Figure 10.9 shows typical variable traces taken of 830 process data points at one minute intervals and gives a plot of the nonconforming or abnormal operation data used in the assessment and validation of the methodology.

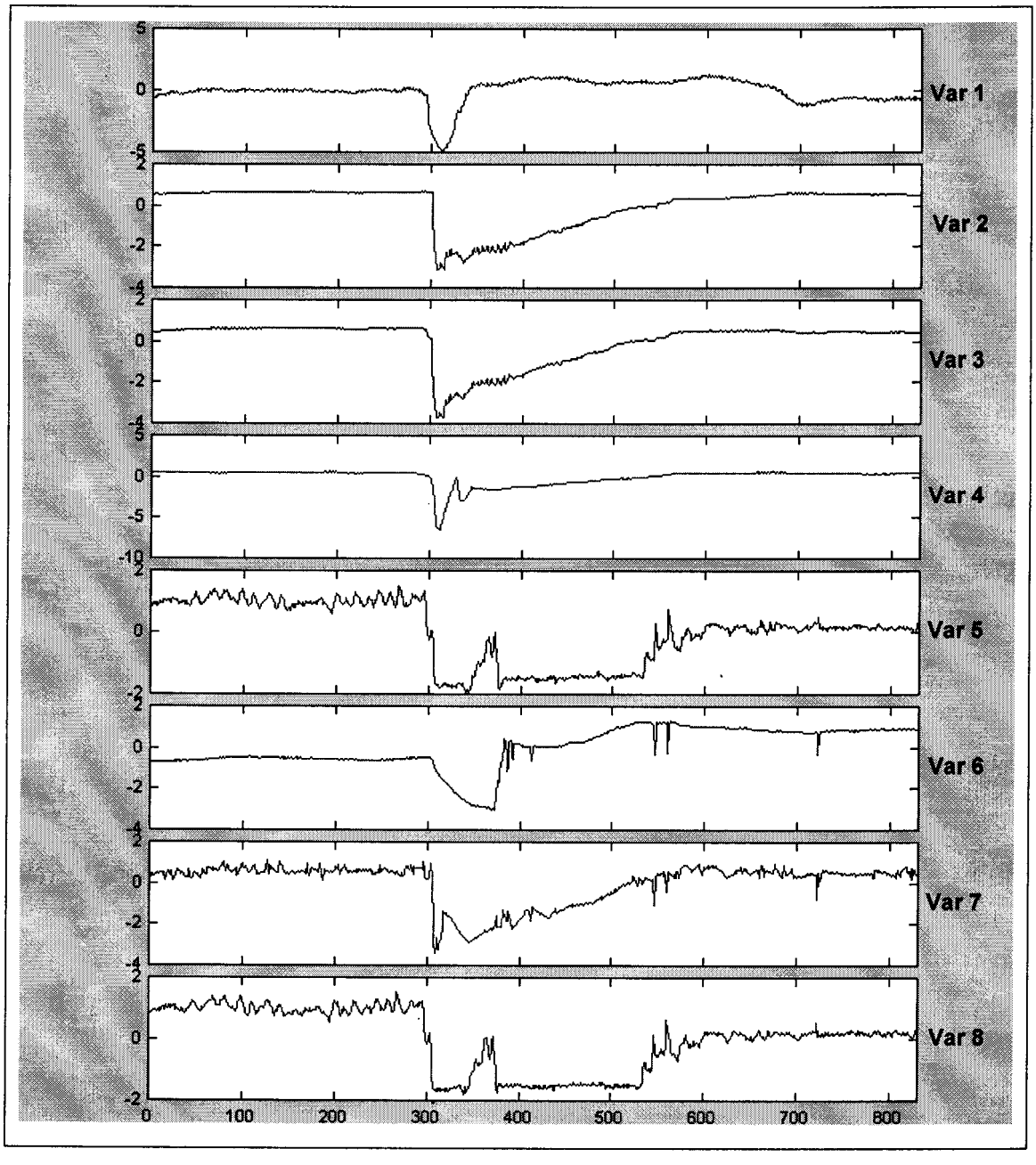

Figure 10.9. Data representing abnormal operation

10.12.3. EVALUATION OF COMPARED MODELS

The aim is not just to show that the algorithm works, but also to satisfy the 3rd stated objective in terms of the critical assessment and validation of the methodology so that the comparative advantages that each of the elements of the NLMSPCA approach offers, becomes clear. This is the main objective of this section. It sets a standard to which the NLMSPCA methodology can be compared. In the end this should enable one to answer the question: How much better is the NLMSPCA approach?

The two methodologies that follow were assessed using the set of unseen data in Figure 10.9 through both the SPE and principal component scores plot. In both cases the action and warning limits were calculated using kernel density estimation.

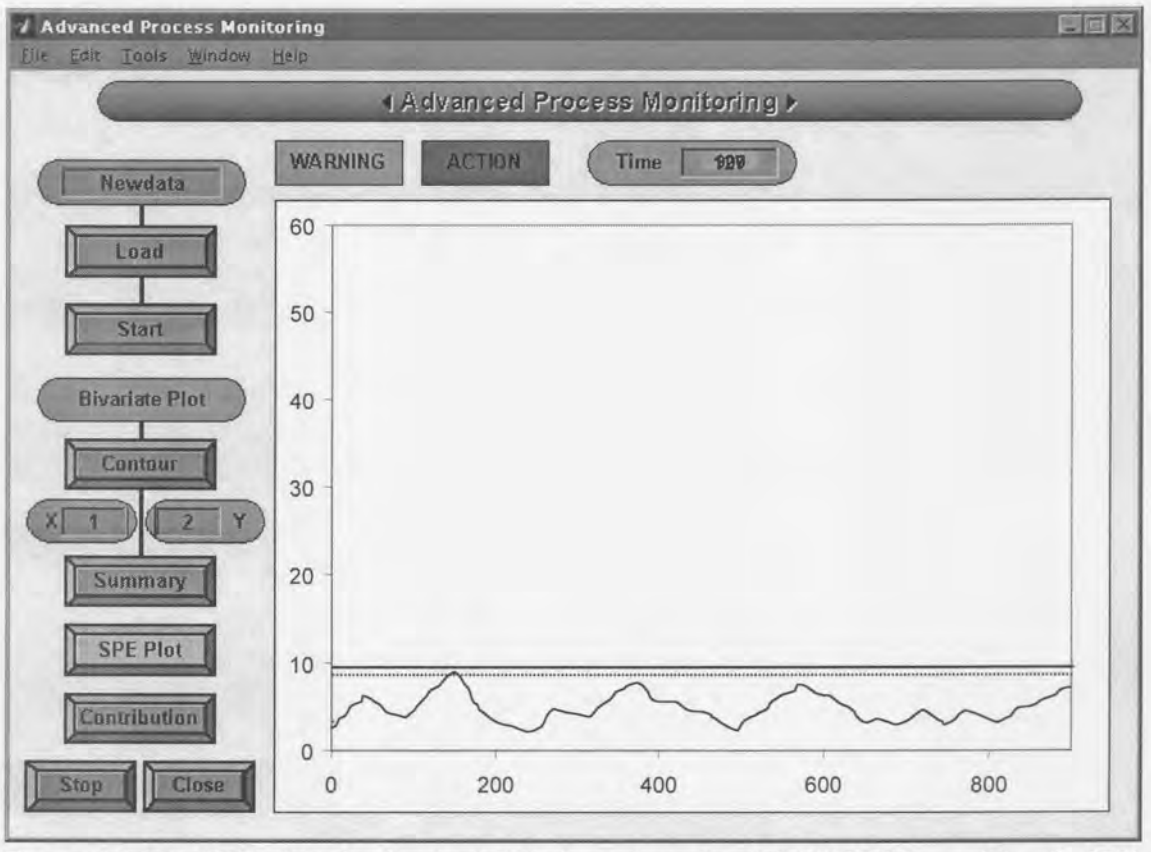

Figure 10.10. SPE plots for the test data based on the 4-3 LMSPCA model with 95% and 99% non-parametric limits

In Figure 10.10 and Figure 10.11 a similar methodology to NLMSPCA was applied except that, instead of using NLPCA, a classical linear PCA was used. The whole process of using neural networks was thus omitted. This will be referred to as the LMSPCA methodology. Using linear principal scores results in different action and warning limits as compared to nonlinear principal scores. Furthermore, using LPCA resulted in six and five principal components to be retained for dataset 1 and dataset 2 respectively to describe the same degree of variability, compared to four and three in the case of NLPCA. As can be seen from the results, this methodology was unable to

effectively detect the point of nonconforming operation. This is to be expected, since the data used was highly nonlinear.

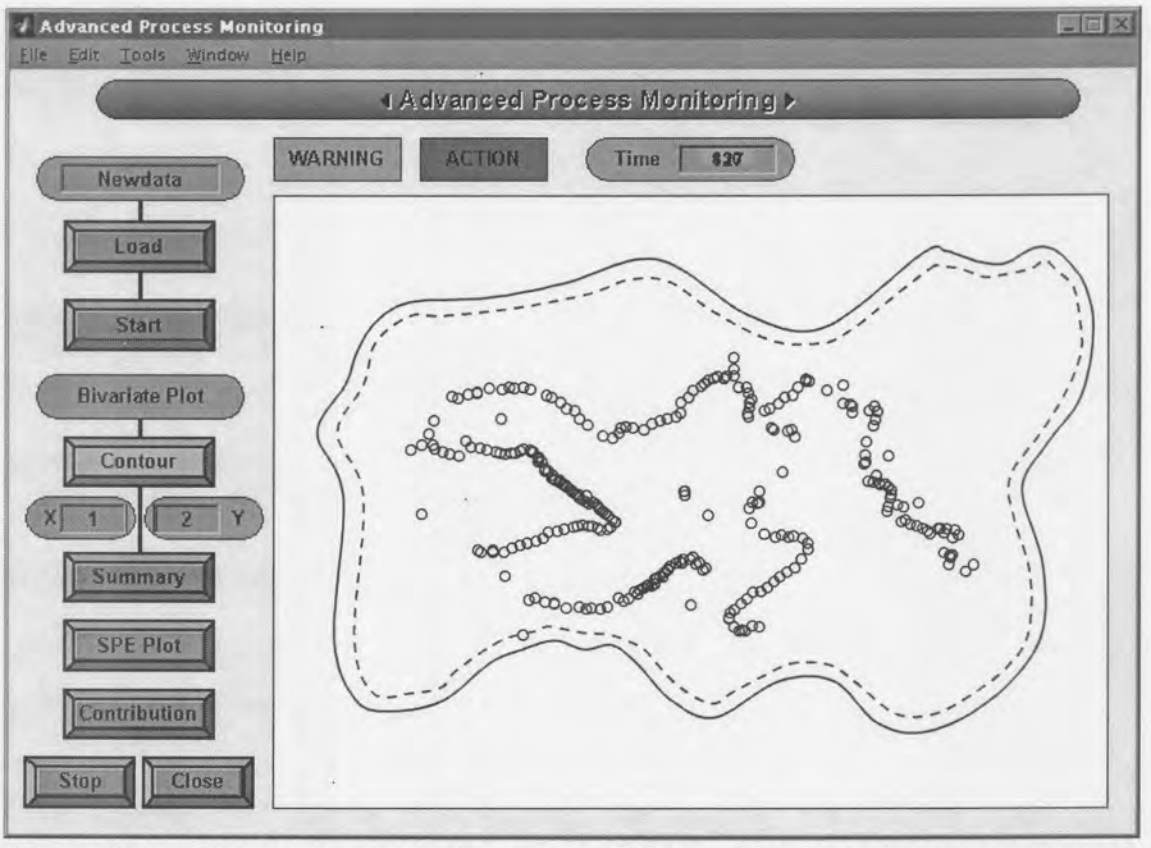

Figure 10.11. Scores plots for the non-conforming test data based on the 4-3 LMSPCA-model.

In the next exercise, again the same methodology was applied to the data in Figure 10.9 except that, instead of using a multiscale methodology, a singlescale methodology was used. This will be referred to as the NLPCA methodology. The processes of multiresolution analysis and wavelet thresholding were thus omitted. This also resulted in only one dataset to be used instead of two. Figure 10.12 and Figure 10.13 gives the results after applying this methodology. As can be seen, it was able to detect the point of nonconforming operation three time intervals earlier than the current alarm system. However, it continued giving false alarms after the process returned to normal operation.

Figure 10.12. SPE plots for the test data based on the 4-3 NLPCA model with 95% and 99% non-parametric limits

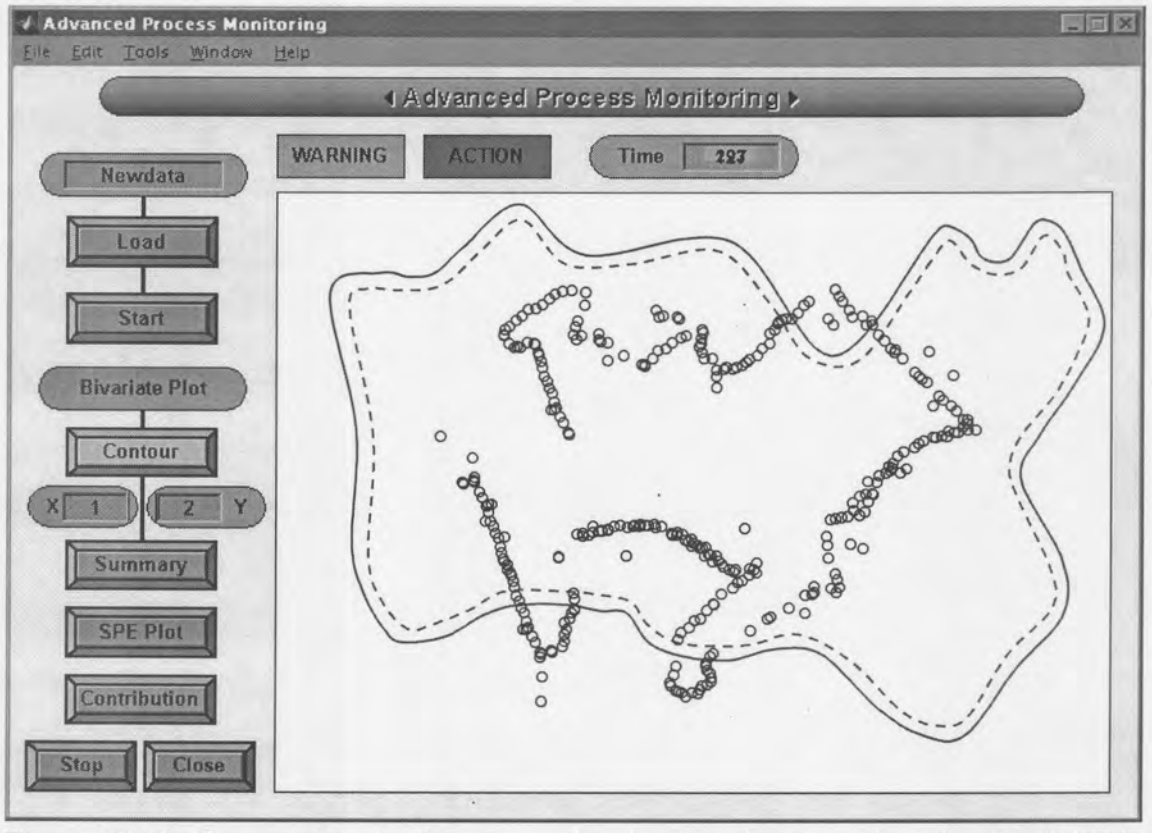

Figure 10.13. Scores plots for the non-conforming test data based on the 4-3 NLPCA-model.

10.12.4. EXPERIMENTAL

In the industrial application indications of failure are difficult to identify, due to the large number of monitored variables, large interactions and nonlinearities as illustrated through Figure 10.10 and Figure 10.11. Current alarm limits as well as the number and type of alarms also have a tendency to conceal the development of abnormal situations. In some cases, for the process under investigation, what was thought to be a failure mode turned out to be a false alarm with no evidence of failure as was partly illustrated in Figure 10.12 and Figure 10.13. It is under these circumstances that one can appreciate a process monitoring scheme, like the one developed here, that is able to overcome these problems and limitations.

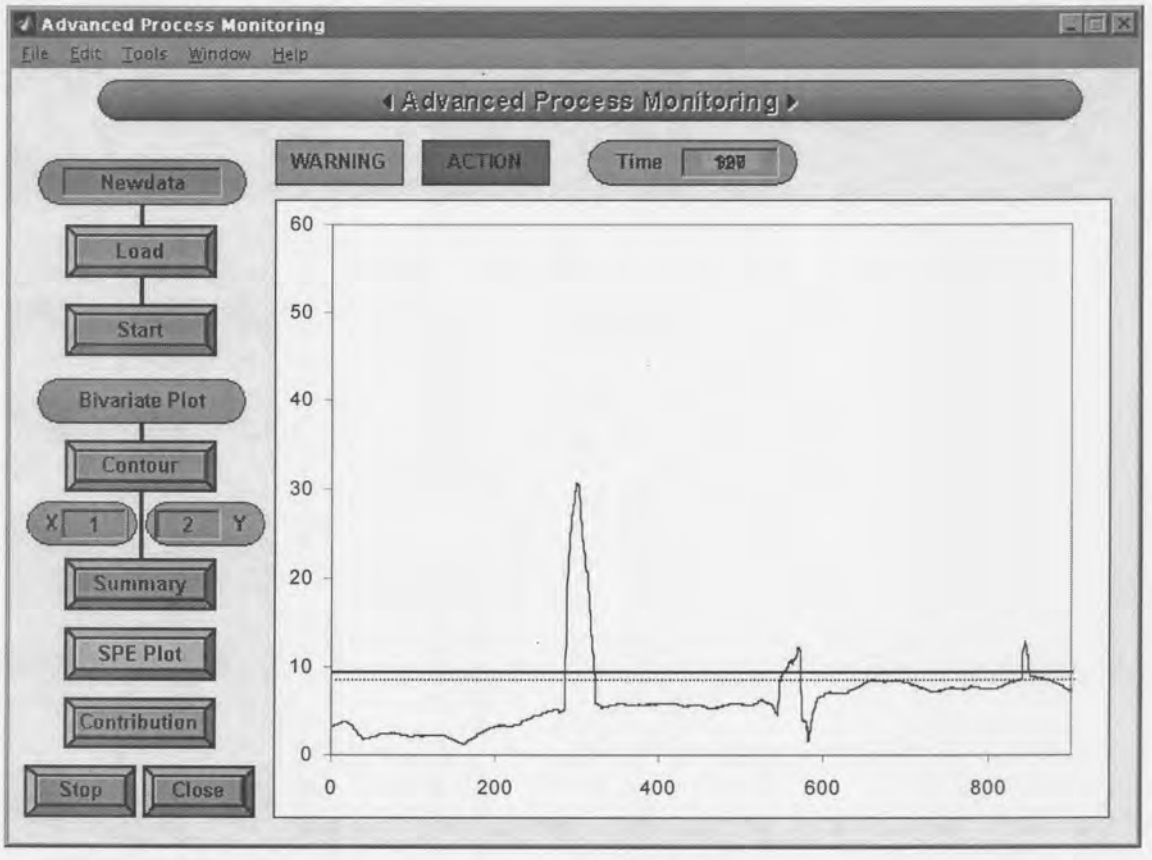

Figure 10.14. SPE plots for the test data based on the 4-3 NLMSPCA model with 95% and 99% non-parametric limits

The nonlinear multiscale PCA scheme introduced in Chapter 8 and 9 was assessed using the set of unseen data in Figure 10.9 through both the SPE and nonlinear principal component scores plot. Figure 10.14 illustrates the results for the SPE for the nonconforming test data set. Also shown are the action and warning limits calculated using kernel density estimation. In this application, the non-parametric control limits are wider than the corresponding limits calculated based upon the assumption of normality. Figure 10.15 shows the nonlinear bivariate scores plot of principal component one

versus two of dataset 1 of the test data with non-parametric control limits, indicating 39 points violating the action limits with the first indication of nonconformance at sample 289.

Figure 10.12 shows the summary plot of the bivariate scores plots for dataset 1 and dataset 2 of the testdata. It can be seen that the process disturbance could be identified from the SPE and bivariate scores plots. The advanced monitoring system was able to detect the process disturbance seven time intervals earlier than the current alarm system. The number of false alarms are also reduced. This illustrates its superiority over the methodologies in Section 10.12.3 and can thus be assumed to be a better methodology.

After a process deviation is identified, the next step is to investigate the cause. In Figure 10.17, a differential contribution plot for non-linear principal component two of dataset 1 and a residual contribution plot were calculated for sample 289, respectively. From both figures, process variable two has the largest contribution, therefore reflecting the possible cause of the process deviation, which gave a positive indication in this case.

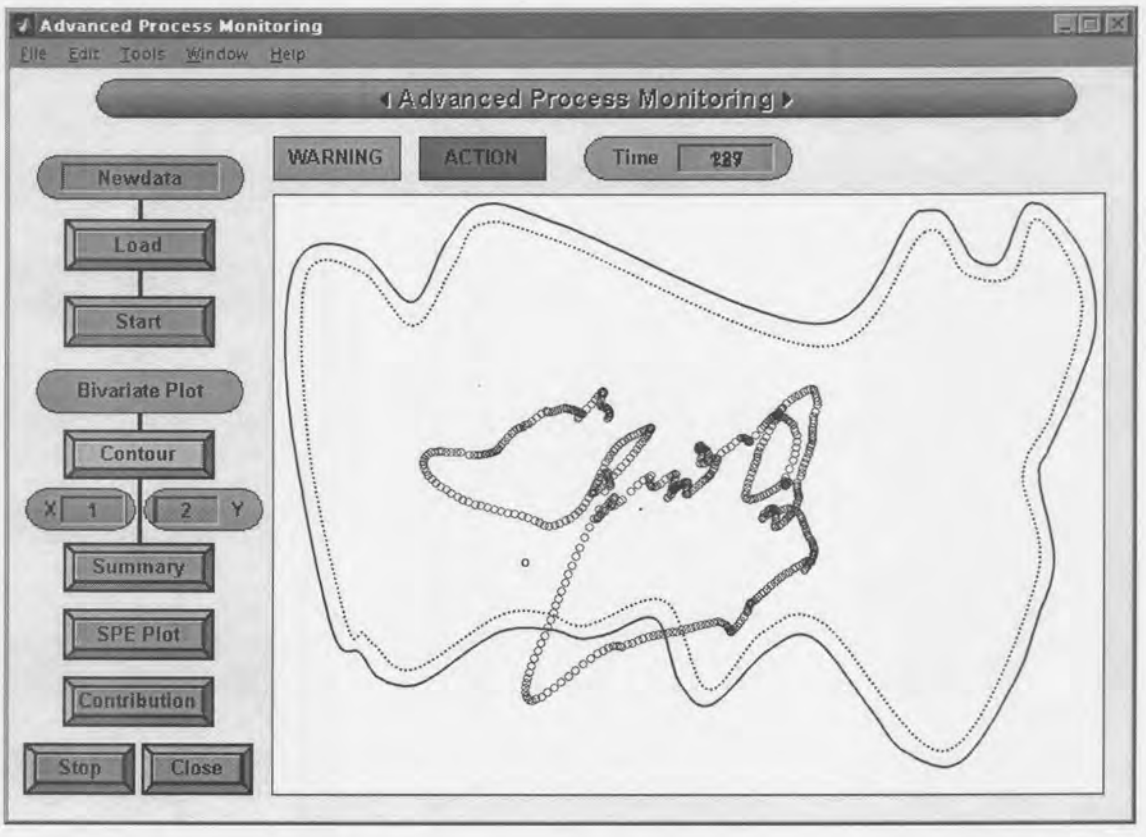

Figure 10.15. Scores plots for the non-conforming test data based on the 4-3 NLMSPCA-model.

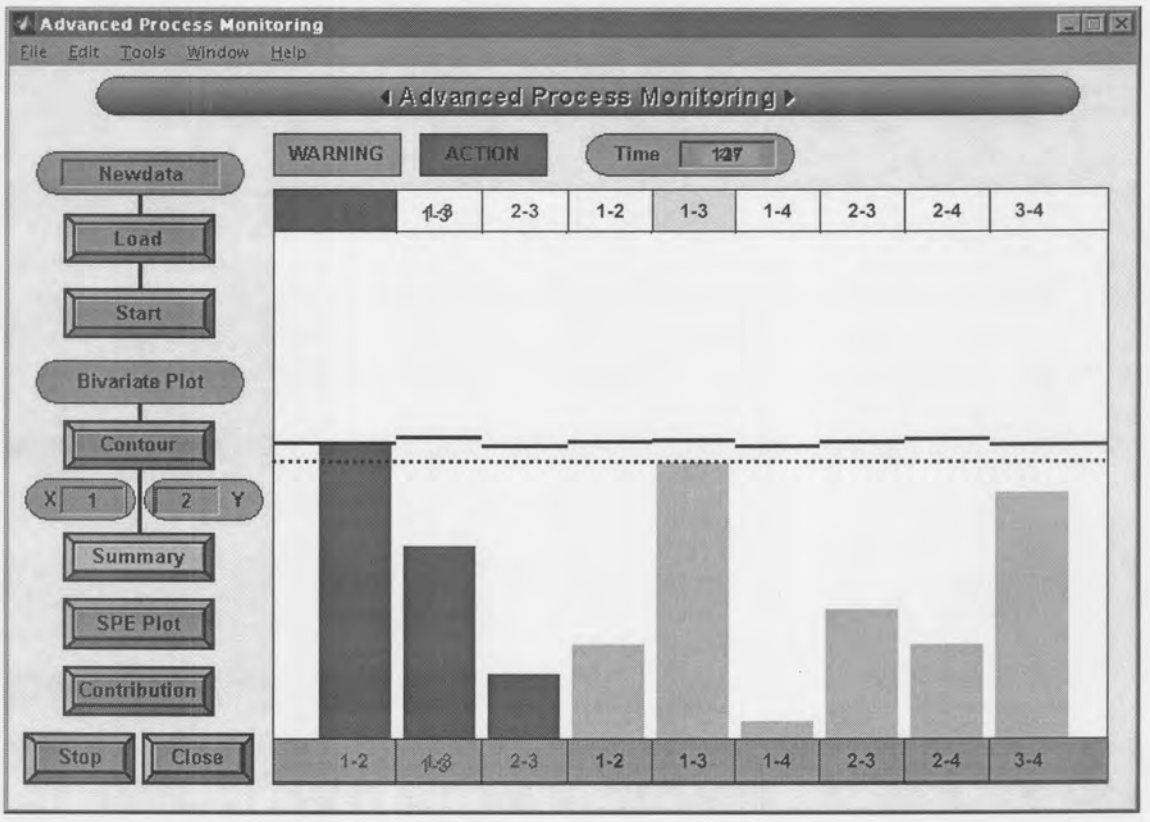

Figure 10.16. Summary plot of the bivariate sores plots based on the 4-3 NLMSPCA-model

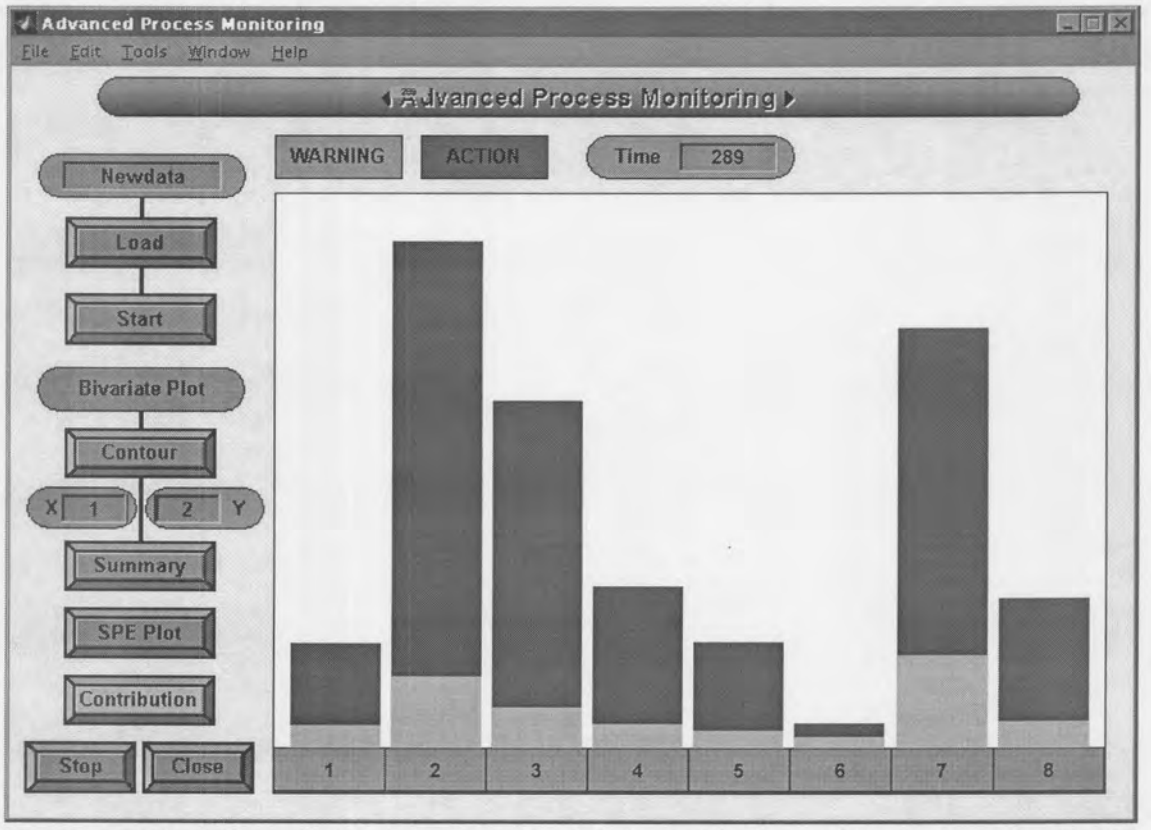

Figure 10.17. Differential ( ) and residual ( ) contribution plots to investigate the cause of process deviation in the non-conforming data

From the differential contribution plot, it is interesting to observe that not only is variable two flagged up, but a large number of other variables appear to contribute to the out-of-control signal. This scenario has been discussed by a number of researchers including Dunia et al. (1996) and Tong and Crowe (1995) and has been identified to be due to the contribution analysis to the SPE being based upon reconstruction. As a consequence, the effect of the changes in the original set of non-conforming variables can propagate to other variable estimates, increasing the chance of erroneous identification. In this respect, interrogation of the contribution plot for the non-linear principal component score is more reliable for the diagnosis step than the linear case. From this industrial application, it can be seen that the nonlinear multiscale PCA model achieves good fault detection results which is also better than using classical LPCA or singlescale analysis.

Chapter 11 CONCLUSION

11.1. Summary

In this study an online multiscale nonlinear PCA approach was derived for process monitoring and fault detection and its performance validated on unseen test data from a nonlinear industrial process. The advantage of this method is that both linear and nonlinear correlations can be extracted from the process data to obtain a more parsimonious description of the original data. The data was first decomposed into different levels of detail and approximations through multilevel wavelet decomposition. Heavy high-frequency noise and sharp data spikes in the industrial data sets were then eliminated through wavelet thresholding. The thresholded level coefficient vectors that contained important information were reconstructed to form reconstructed details and approximations containing the most important information of the data at different levels. Thus, in applying the discrete wavelet transform the underlying process trend was preserved in the approximation and detail coefficients. This was then used to develop a linear and nonlinear principal component model. All the scales were also combined for deriving a combined principal component model. Using wavelet coefficients in the derivation of the nonlinear PCA model significantly reduces the computational burden without impacting upon the predictive ability of the process representation. Moreover, the possibility of the input-training network to overfit the data is greatly reduced and the generalisation properties of the network enhanced. Fortunately, the last MSPCA steps of selecting the scales that indicate significant events, reconstructing the signal to the multilevel time domain, and computing the scores and residuals for both the thresholded and non-thresholded reconstructed signals, improve the speed of detecting abnormal operation and eliminate false alarms after a process returns to normal operation. Using the multilevel methodology also greatly enhances the ability of the monitoring system to detect different types of abnormal conditions. Data-driven, nonlinear control limits and modified contribution plots were derived to facilitate the comprehensive and robust monitoring and fault detection.

The results of the application of the conjunction of the multilevel wavelet decomposition, wavelet thresholding technique and nonlinear PCA algorithm to an industrial process demonstrates the advanced performance for fault detection and isolation. According to the results it should be possible to determine the development of an abnormal situation in the steam distribution system early enough in order to reduce the consequences of the abnormal event. Here the methodology was only applied to one specific case. The accuracy and reliability of the methodology needs to be validated on more scenarios.

Keeping the process in mind it should be clear at this stage that it is not yet possible to apply this methodology in real time since the factory currently lacks the infrastructure. At this stage it is not possible to access or monitor all the variables throughout the factory from a single point. It is only possible to access a specific unit's variables from that unit's control room. However, this infrastructure will be implemented over the next two years. The only way to currently gain access to all the variables from a single point is through the

History Module which is a central database. This database only saves data at a minimum sampling rate of one minute and access to the database is not very reliable.

Apart from this industrial application, it can be seen that the NLMSPCA model achieves good fault detection results. Moreover, the non-parametric control limits are statistically more valid for non-linear, on-line, process performance monitoring.

The use of the summarised plots allows enhanced global visualisation capabilities and interpretation and reduces the space taken up by conventional multivariate statistical plots. It can be concluded that the advanced monitoring system architecture is qualified for further development.

11.2. Further development

The next step would involve the development of an application to create plans to recover from malfunctions and threatened goals. It should be capable of replanning in real time, and use knowledge of the process represented in blackboards to carry out planning without the need for human planners to exhaustively explore every possible scenario. This function should assess the success of the plan and be capable of closed loop control of the process if so authorized.

11.3. Practical implications

In process monitoring and fault detection the major issue becomes that of practical implications.

Common to all approaches described as intelligent fault detection, is that they derive or synthesize higher order statements about the plant from lower order information, e.g. process measurements, event information, alarms. They must all be seen as add-ons which complement an existing good quality basic process alarm system. They will produce results if the basic information system is sound. None of the approaches will cure fundamental faults in the basic alarm system, and should not be considered as doing so.

The major problem with all the computerized fault detection techniques is that, even with sufficient implementation tools, they require considerable engineering analysis of plant behavior. Some of this can be done 'on paper' from the plant design information, but generally considerable post-commissioning tuning is also required. Applying these techniques also demands some 'failure mode analysis' to be performed to ensure missing or incorrect input data not causing false conclusions. Questions also remain with artificial intelligence and expert systems techniques

about demonstrating that a procedure that is developed on a limited range of plant transients will be effective in unexpected situation.

The development and application of this technique would require specialist knowledge. Unfortunately the reports of large scale practical applications on working plants are few and far between. Priority should be given to applying other more basic methods to eliminate the simple problems, and only then to invest in the more advanced methods. So why develop more advanced technology? By the time the more basic problems have been addressed, advanced methods like the one developed here should be ready for application since the experimental stage, and especially the period up to general acceptance and reliability, for new and advanced technology is much longer.