STATISTICAL MONITORING AND FAULT DIAGNOSING OF MULTIVARIATE PROCESSES BASED ON NONLINEAR PRINCIPAL COMPONENT ANALYSIS

by

Xueman Li

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTERS OF APPLIED SCIENCE
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY

March 2004

Approval

Name:	Xueman Li
Degree:	Master of Applied Science
Title of Thesis:	Statistical Monitoring and Fault Diagnosing of Multivariate Processes Based on Nonlinear Principal Component

Examining Committee:

Chair: Dr. John C. Dill
Professor, School of engineering Science
Simon Fraser University

Dr. Mehrdad Saif
Senior Supervisor
Professor, School of Engineering Science
Simon Fraser University

Dr. Andrew H. Rawicz
Supervisor
Professor, School of Engineering Science
Simon Fraser University

Dr. John D. Jones
Examiner
Associate Professor, School of Engineering Science
Simon Fraser University

Date Approved: March 19 2004

Abstract

Model-based fault detection and isolation (FDI) in plants of complex control systems has been a subject of tremendous research over the last three decades. Most common FDI approaches are based on analytical models of the systems which are often not available in practice for complex multivariate processes.

On the other hand, statistical linear correlation models developed using principal component analysis (PCA) can be built from historical operating databases and require no prior knowledge of the plant. Statistical process monitoring (SPM) approaches using these models can easily handle a large number of variables and are very powerful for fault detection. Their main limitations lie in the linear assumption of the process variables and the ability to isolate or diagnose faults.

This thesis presents a multivariate statistical process monitoring (MSPM) and fault diagnosis approach based on nonlinear principal component analysis (NLPCA). A technique called NLPCA neural network is applied to model the process of interest. It addresses the linearity limitation of the PCA by assuming that the hidden principal components are nonlinear functions of the observed process variables; therefore, it is more effective in extracting the information from nonlinearly correlated variables than linear methods. A new statistic fault diagnosing scheme is also developed based on analyzing the distribution patterns of the process data in the nonlinear principal component feature space through the use of self-organizing feature mapping (SOFM) and vector quantization algorithms.

The proposed procedure is compared with observer-based methods and current statistical methods in performing process monitoring and fault diagnosis of linear and nonlinear processes. Its applications are illustrated on the diesel engine actuator benchmark system as well as the three-tank benchmark system.

Dedication

I dedicate this thesis to my husband Hui , my lovely son Liam and my parents.

Acknowledgements

I am extremely grateful to my Senior Supervisor, Professor Mehrdad Saif, for introducing and motivating me to the area of statistical fault diagnosing, and for providing me with timely guidance and valuable suggestions during the entire M.A.Sc program. I also thank him for his kind understanding and continuous support, which made it easier for me to go through those stressful times in my life.

I give special thanks to Professor Andrew H. Rawicz for serving as my Supervisory Committee member. Thanks are also due to Professor John D. Jones for accepting to be my Internal Examiner. I am very grateful to them for taking their valuable time to review my thesis.

I have benefited greatly from enthusiastic and enlightening discussions with my friends and colleagues: Dr. Wen Chen, Dr. Weitian Chen, Ms. Fang Liu, Mr. Guoyu Wang, Mr. Qinguo Li and Mr. Guangqin Jia. I wish to thank them for their help and for the delightful time we have spent together.

My particular thanks to my husband Hui Feng, for supporting me during these years. Our son Liam joined us last year and has been a source of great joy. I would also like to thank my parents, my brother and sister for their encouragement and love.

Table of Contents

List of Figures

List of Tables

Table of Acronyms

CVA:	Canonical Variate Analysis
CVSS:	Canonical Variate State Space
CUSUM:	Cumulative Sum
EWMA:	Exponentially Weighted Moving Average
FDI:	Fault Detection and Isolation
FNN:	Feedforward Neural Networks
LVQ:	Learning Vector Quantization
MLP:	Multilayer Perception
MIDAS:	Model Integrated Diagnostic Analysis System
MSPM:	Multivariate Statistical Process Monitoring
MSPC:	Multivariate Statistical Process Control
NOC:	Normal Operation Condition
NLPCA:	Nonlinear Principal Component Analysis
OBD II:	On Board Diagnostics Generation II
PCA:	Principal Component Analysis
PC:	Principal Component
PLS:	Partial Least Squares
RBFN:	Radio Basis Function Network
SDG:	Signed Directed Graph
SIM:	Subspace Identification Method
SOFM:	Self-Organizing Feature Mapping
SPM:	Statistical Process Monitoring
TDNN:	Time-Delay Neural Network

Chapter 1

Introduction

Over the past three decades, efforts to manufacture high quality products and to improve the efficiency, reliability and safety of modern complex systems have led to the increased use of process monitoring and quality control techniques in a variety of industries. For example, the new California Air Resource Board and the Environmental Protection Agency regulations have required that all 1996 and newer light duty vehicles should be equipped with the On Board Diagnostics Generation II (OBD II)system. The OBD II system monitors virtually every component that can affect the emission performance of the vehicle to ensure that the vehicle remains as clean as possible over its entire life, and assists repair technicians in diagnosing and fixing problems. In the chemical industry, statistical process monitoring techniques are widely adopted to detect leaks, clogs, and other faults that mostly occur in the transport of fluids. The field of fault diagnosis has become a widespread and interesting topic of research. The purpose of fault diagnosis is to detect unexpected changes in functional units that tend to degrade overall system performance, and to determine the causes of faults early enough so that a complete system failure can be avoided.

1.1 A Review of Model-Based Fault Diagnosis

Most fault detection and isolation (FDI) methods developed today fall under the umbrella of model-based approaches, and can be classified into three categories [1]: analytical model-based methods, qualitative model-based methods, and statistical model-based methods. These three classes of methods are described next.

1.1.1 Analytical Model-Based Methods

In this approach, analytical system models are obtained from theory, or identified empirically from experiments. Numerous FDI methods developed based on analytical system models can be subdivided into three groups [2]:

- Parity Space Approach

  The parity space approach tests the consistency of parity equations using the measured signals of the actual process. From the inconsistency of the parity equations one can detect faults. Chow and Willsky [3] derived the parity equations from the state space model of the system. Further contributions are due to Gertler and his co-workers [4].

- Observer-Based Approach

  The analytical observer-based approach reconstructs the outputs of the system with the aid of observers or Kalman filters and uses the estimation errors as residuals. Fault isolation is implemented by constructing structured residuals such that individual faults are mapped into different directions. The residuals are independent of each other, and for the sake of robustness, are independent of unknown inputs. Fault isolation can be achieved either in the time domain [5] or in the frequency domain [6, 7].

- Parameter Estimation Approach

  The basic idea of the parameter estimation detection approach is that the parameters of the actual process are repeatedly estimated on-line using well known parameter estimation methods and the results are compared with the parameters of the reference model obtained initially under fault-free conditions. Any substantial discrepancy is declared as a fault [8].

Due to the complexity of practical processes, obtaining complete mathematical models is usually difficult. The analytical model-based approach is generally limited to processes with a small number of variables.

1.1.2 Qualitative Model-Based Methods

As was noted, the analytical approach suffers from the fact that under real conditions, a precise model of the system of interest is generally unavailable. In the case of noticeable modelling uncertainty, a more suitable strategy consists of using qualitative model-based techniques. In this approach, the knowledge is derived in terms of facts and rules from the description of the system structure. Model integrated diagnostic analysis system (MIDAS) [9], signed directed graph (SDG) and fault diagnosis tree are the most common qualitative models for fault detection and isolation.

1.1.3 Statistical Model-Based Methods

In recent years, the huge increase in computing power has prompted significant technological advances in instrumentation and data logging in the field of process control. Many large plants are now equipped with sophisticated distributed control systems capable of logging literally hundreds of variables of the process in every passing second. The tremendous number of data available from routine operating processes, coupled with the realization that obtaining complete analytical models of processes is difficult due to the process complexity, has led to serious considerations of process monitoring using statistical techniques. The most important approaches are the so called correlation model-based methods and neural network-based methods.

Correlation Model-Based Methods

Statistical process monitoring and fault diagnosing involve three activities [45]: 1) detecting out-of-control status, 2) identifying the variable(s) responsible for the process going out of control, and 3) diagnosing the source cause for the abnormal behavior. Monitoring focuses on detecting activities, while diagnosing provides information for the intervention or control stage. Classical univariate statistical monitoring techniques rely on the use of the Shewhart chart [10], the cumulative sum (CUSUM) plots [11], and the exponentially weighted moving average (EWMA) chart [12]. All of these are typically used for monitoring a small number of quality variables. Little information is derived from the interactions between variables, which are very important for fault diagnosis in complex dynamic processes. Multivariate statistical process monitoring (MSPM) methods address the limitations of univariate monitoring techniques by processing all the observations simultaneously and extracting features on the 'directionality' of the process variations. FDI approaches using these models fall under the category of multivariate statistical process control (MSPC).

The classic techniques of MSPM are the projection methods of principal component analysis (PCA) [13, 14] and partial least squares (PLS) [15]. The philosophy behind these approaches is to reduce the dimensionality of the problem by eliminating the linear correlations between the random variables. By forming a new set of uncorrelated variables or features in the lower dimensional space, an enhanced understanding of the process behavior may be obtained.

Another approach for MSPM is based on the well-known subspace identification methods (SIMs). The most influential SIMs are canonical variate analysis (CVA) method [16, 17], multivariate output error state space (MOESP) method [18] and numerical subspace state-space system identification method (N4SID) [19]. SIMs try to estimate the state variables directly from the system input and output data. Process monitoring is based on the Hotelling T^2 statistic of the state variables.

Statistical correlation models developed using PCA or CVA can easily handle a large number of variables and are very powerful for fault detection. However, their main limitations lie in their ability to isolate or diagnose faults.

Neural Network-Based Methods

It has been recognized that neural networks offer a number of advantages for modelling the dynamics of systems. One of the main advantages of neural networks is the ability to model non-linear systems; in practice most systems are non-linear and their mathematical models are difficult to build. Another attractive property of neural networks is the capability of generalizing underlying rules of system behaviors through learning examples. The learning ability of neural networks is rooted in statistical mechanics. Neural networks can also handle complex systems with a large number of process variables. These properties of neural networks offer the possibility of using them to solve fault diagnosis problems. Since the late 1980s artificial neural networks have been widely discussed for model-based fault detection and isolation in slowly varying complex systems where analytical models are not available. [20, 21, 22].

Currently, there are two kinds of neural network structures used for system modelling. They are called multilayer feedforward networks and recurrent networks. The feedforward network structure is one of the most popular and arguably the most important ANN structures. It builds the system model based on studying the spatial correlations of the process; it is therefore considered as a static network structure. For dynamic system modelling, feedback has to be introduced into the neural network topology to derive dynamics. Such a neural network is called the recurrent neural network.

1.2 Thesis Outline

Standard PCA and SIM based MSPM methods are limited by the assumption that the hidden features or state variables are linearly correlated with the process observations. These methods are not efficient when the variables are from highly nonlinearly correlated processes. Since in practice, most manufacturing processes are inherently nonlinear, this thesis work has focused on the study of statistical representations of nonlinear systems in an attempt to design a nonlinear model-based process monitoring and fault diagnosis system.

One technique applied here is called nonlinear principal component analysis (NLPCA) neural network which is used to model nonlinear processes. NLPCA is a generalization of PCA. It addresses the limitation of PCA by assuming that the hidden principal components are nonlinear functions of the observed process variables. The NLPCA technique has been shown to be able to describe the underlying structure of data and is more effective in extracting information from nonlinearly correlated variables than conventional linear methods [23, 24]. NLPCA has been applied to various fields such as pattern recognition [25, 26], data compressing [27], medical signal processing [28], and nonlinear dynamical problems that appear in chemical engineering [29]. NLPCA based methods for nonlinear process modelling have been previously introduced by a few researchers [20, 24, 30], with most techniques being based upon artificial neural networks. Of particular interest is a special multilayer feedforward network called the autoassociative neural network. In this thesis, the NLPCA autoassociative network is applied to monitor slow-varying nonlinear processes with a large number of process variables. An extension of the static NLPCA network is also achieved by combining the NLPCA network with the time-delay neural network (TDNN). The TDNN is an efficient tool to discover spatiotemporal relationships both directly in the input layer and in the more abstract representation of the hidden layer. The NLPCA neural network with TDNN is used to monitor dynamic systems with multiple process variables. In addition, by selecting different groups of process variables based on causal knowledge of the process, the NLPCA neural network with TDNNs can be designed to be sensitive to a specific kind of fault and to be insensitive to other faults, which makes fault isolation possible.

A new statistical fault identification approach based on studying the distribution patterns of the process principal components is also introduced in this thesis. It is believed that most industrial processes are usually operated in a few normal operation regions. A process is subject to input changes as well as system parameter changes. These changes affect the process operation and consequently result in the process drifting away from its normal operation region. By studying the directions of the process movement, different faults can thus be identified. Score plots of the principal components in two or three dimensions have previously been used to identify the onsets of faults. The idea is based on the assumption that different faults may cause the process to move in different directions, and by checking the direction of the process movement, the cause-effect relationship may be established. However, due to the limitation of dimensionality, score plots are usually suitable for processes with small number of variables. For nonlinear complex processes or processes with large numbers of variables, score plots are not reliable for fault identification since the movements of the process on different fault conditions are not always distinctive in the score plots. Instead, they often overlap with each other or are buried within the nominal region which makes them difficult to identify. In addition, process movements are subject to change when fault levels change, which makes score plots based on fault identification even more difficult. In this work, fault identification is automated by using a new type of network called the self-organizing feature mapping (SOFM) network. The SOFM neural network has received wide interest in recent years because of its significant success in analyzing and visualizing high-dimensional data. It carries out a nonlinear mapping of input data onto a low-dimensional grid. The mapping preserves the most important topological and metric relationships of the data. In this research work, the SOFM network turned out to be an efficient tool to describe the distribution patterns of process data in the principal component feature space, which makes computer-aided statistical fault identification possible.

A new fault diagnosis scheme that combines the NLPCA neural network with the SOFM is also presented in this thesis to solve the problem of process monitoring and fault identification of slowly varying processes with large number of variables. A Kramer's NLPCA autoassociative neural network is first trained to model the normal operation of the processes of interest. This model is then used to produce residuals when the process is running under unknown conditions. For fault identification purpose, historical process data from various known fault conditions are projected to the same model and nonlinear principal components are extracted to form the features of the corresponding faults. SOFM networks are subsequently employed to learn the distributions of different faults in the feature space. Statistical distribution models are then built from the feature space. Given an unknown faulty condition, fault identification is achieved by comparing its distribution in the feature space to known fault models. Bayes decision theory is used to declare a fault condition if the distribution is close to a known fault model.

The remainder of this thesis is organized as follows: In Chapter 2, the NLPCA concept and the NLPCA neural network model are first presented. In Chapter 3, multivariate statistical process monitoring methods are introduced, such as the standard PCA method, the state space analysis method, and the NLPCA method. The applications of these methods to linear and nonlinear processes are also discussed. In Chapter 4, the time-delay neural network (TDNN) is introduced as an extension of the static NLPCA neural network to model dynamic systems. Its implementation is illustrated on the industrial actuator benchmark system. In Chapter 5, the fault identification method using SOFM is discussed. In Chapter 6, the implementation of the NLPCA-SOFM-based fault diagnosis approach is illustrated on the three-tank benchmark system. Lastly, Chapter 7 contains concluding remarks.

Chapter 2

Principal Component Neural Networks

2.1 Introduction

A central problem in statistical data processing is finding a suitable representation of data by means of a transformation. The transformation facilitates the subsequent analysis of data, whether it is for the purpose of data visualization, de-noising, pattern recognition, or anything else. This chapter first introduces a classical linear transformation method called principal component analysis (PCA). Next, an extension to PCA is described called nonlinear principal component analysis (NLPCA) which generalizes PCA to nonlinearly correlated data. The neural network technique for implementing NLPCA is covered in the last part of the chapter.

2.2 Principal Component Analysis

Principal component analysis, or PCA, is widely used in signal processing, statistics and neural computing. In some application areas, it is also called the Karhunen-Loeve transform.

Given an n-dimensional random variable $\mathbf{x}$, the basic idea of PCA is to find a linear transformation $W \in R^{p\times n}$ so that the linearly transformed components defined by:

$$\mathrm{s} = W\mathbf{x}, \qquad \mathrm{s} \in R^p \tag{2.1}$$

explain the maximum amount of variance of random variable $\mathbf{x}$. Here $\mathrm{s} = [s_1, s_2, ..., s_p]^T$ is a p-dimensional vector in the transformed feature space. The p elements $s_1, s_2, ..., s_p$ are called principal components, which are calculated by:

$$s_i = \mathbf{w}_\mathrm{i}\mathbf{x} \qquad i = 1, 2, ..., p. \tag{2.2}$$

Here $\mathbf{w}_\mathrm{i}$ is the eigenvector that corresponds to the *ith* largest eigenvalue of the covariance matrix

$$C_x = E\{\mathbf{x}\mathbf{x}^\mathbf{T}\}. \tag{2.3}$$

In addition, from the p principal components $s_1, s_2, ..., s_p$, it is also possible to reconstruct the random variable $\mathbf{x}$ through

$$\hat{\mathbf{x}} = \sum_{i=1}^{p} s_i \mathbf{w}_i^T, \qquad p \leq n. \tag{2.4}$$

The reconstruction error is defined by

$$J = E\{||\mathbf{x} - \hat{\mathbf{x}}||^2\}. \tag{2.5}$$

The fundamental purpose of PCA is to reduce the dimension of data ($p \ll n$). In fact, it has been proved that the representation given by PCA is an optimal linear dimension reduction technique in the sense that the mean square error defined by (2.5) is minimized. The dimension reduction has some important benefits. First, it can reduce the computational load in the subsequent processing stages. Second, noise may be deleted, since the data corresponding to the discarded $n - p$ smallest components are most likely due to noise. Finally, a projection into a subspace of low dimension enables data visualization.

2.3 Nonlinear Principal Component Analysis

The classic linear PCA method assumes that the transformed features of the process are linear functions of the observed variables. In many industrial situations, however, this assumption may not be true when the observations are from highly nonlinear processes. In such cases, it may be more appropriate to assume that the feature subspace is defined by nonlinear functions of the process variables.

In general, it is assumed that from an n-dimensional observation vector $\mathbf{x} = [x_1, \ldots, x_n]^T$, we can extract the underlying feature vector $\mathbf{s} = [s_1, \ldots, s_p]^T$, $(p \leq n)$ via function $h : R_n \Rightarrow R_p$, and can also reconstruct the observation from the feature vector via function $g : R_p \Rightarrow R_n$. The mapping from the data space to the feature space is referred to as coding and the reverse mapping as decoding. The coding function h and decoding function g are members of nonlinear continuous function sets F_h and F_g respectively.

The target is to minimize the nonlinear reconstruction mean square error (MSE) by choosing the optimal functions $g \in F_g$ and $h \in F_h$. Naturally, the problem

$$\min_{g,h} J = E\{||\mathbf{x} - g(h(\mathbf{x}))||^2\} \tag{2.6}$$

is called the nonlinear principal component analysis (NLPCA) problem. Clearly, the solution to the NLPCA problem depends on both the choice of the function sets F_h and F_g:

The elements $s_1, \ldots, s_p$ of the feature vector $\mathbf{s}$ are called nonlinear principal components. They are the nonlinear features that underlie the distribution of the data $\mathbf{x}$. The optimal choice of h and g (i.e. h^* and g^*) is not unique. Indeed, if $[h^*(\cdot), g^*(\cdot)]$ are NLPCA solutions for $\mathbf{x}$, which achieve the minimum error

$$J_{min} = E\{||\mathbf{x} - g^*(h^*(\mathbf{x}))||^2\}, \tag{2.7}$$

then the same error is achieved by any pair of functions of the form $[h_q = q^{-1}(h^*(.)), g_q = g^*(q(.))]$ for any invertible function $q : R^p \Rightarrow R^p$ [32]. In general, the unique recovery of the hidden factors is impossible. However, we can obtain some unique properties of h and g starting from the following theorem [31]:

Theorem 2.1 *Let the variables* $\mathbf{x} \in R^n$ *and* $\mathbf{s} \in R^p$ *be jointly distributed. Then the best estimate of* $\mathbf{x}$ *by a function of* $\mathbf{s}$ *is the conditional expectation of* $\mathbf{x}$ *given* $\mathbf{s}$, *namely*

$$\min_{\hat{\mathbf{x}}} E\{||\mathbf{x} - \hat{\mathbf{x}}(\mathbf{s})||^2\} = E\{||\mathbf{x} - g(\mathbf{s})||^2\} \tag{2.8}$$

where

$$g(\mathbf{s}) = E\{\mathbf{x}|\mathbf{s}\}. \tag{2.9}$$

Thus, given the coding function h, the optimal solution to the objective function (2.6) is when

$$g(\mathbf{s}) = E\{\mathbf{x}|(\mathbf{s} = h(\mathbf{x}))\}. \tag{2.10}$$

Since function h maps a higher-dimensional data space to a lower-dimensional feature space, it is a many-to-one function. For a given feature $\mathbf{s}$, the set $h^{-1}(\mathbf{s}) = \{\mathbf{x} : h(\mathbf{x}) = \mathbf{s}\}$ is the isometric surface of h corresponding to value $\mathbf{s}$. Moreover, the function $g(\mathbf{s})$ is a p-parametric surface called the principal component surface of $\mathbf{x}$, and by (2.10), $g(\mathbf{s})$ is the mean of the set $h^{-1}(\mathbf{s})$.

As it has been mentioned in the above example, if a pair of functions $[h(\cdot), g(\cdot)]$ get the minimum reconstruction error defined by (2.6), then the same is also true for the functions $[h_q = q^{-1}(h(.)), g_q = g(q(.))]$, where $q : R^p \Rightarrow R^p$ is any invertible function . Let I be the set of isometric surfaces

$$I(h) = \{h^{-1}(\mathrm{s}), \forall \mathrm{s} \in R^p\}. \tag{2.11}$$

Since

$$\mathrm{s} = h_q(\mathbf{x}) = q^{-1}(h(\mathbf{x})) \tag{2.12}$$

and

$$h_q^{-1}(\mathrm{s}) = h^{-1}(q(\mathrm{s})) \tag{2.13}$$

therefore,

$$I(h_q) = \{h^{-1}(q(\mathrm{s})), \forall \mathrm{s} \in R^p\} = \{h^{-1}(\mathrm{s}'), \forall \mathrm{s}' \in R^p\} = I(h) \equiv I \tag{2.14}$$

where

$$\mathrm{s}' = q(\mathrm{s}). \tag{2.15}$$

Therefore, for a given distribution of $\mathbf{x}$, we can conclude that the isometric surface set I is invariant with respect to the coding function h; in other words, it is unique [32]. Furthermore, since the principal component surface $g(\mathrm{s})$ is the mean of the set $h^{-1}(\mathrm{s})$, it is invariant with respect to the decoding function g. Therefore, it is also unique. These unique properties are the foundation for implementing the NLPCA technique to model a random process. The functions h and g can be accomplished by different kinds of neural structures.

The linear PCA is a special case of NLPCA when both h and g are linear mappings. In general, the performance of NLPCA cannot be worse than linear PCA if F_h and F_g contain all linear mappings of the corresponding dimensions.

The drawback of NLPCA lies in its computational complexity since the nonlinear optimization problem is much harder to solve and usually has no closed-form solution. On the other hands, neural learning algorithms are much more computationally attractive [33, 34, 35, 36].

2.4 NLPCA Neural Networks

Currently, two kinds of neural networks are often used for system modelling. They are the multilayer perceptrons (MLPs) and the radial basis function networks (RBFNs). In the MLPs, neurons are activated by all inputs; therefore, they are called global neural networks. In the RBFNs, neurons only respond to some particular regions of input space; hence they are called local neural networks. MLPs usually need smaller numbers of hidden neurons than RBFNs, and are widely used for function approximation problems. RBFNs, on the other hand, need shorter training time and are often used for classification problems. In this thesis, the MLP networks are used for process modelling.

2.4.1 Feedforward Neural Networks

Feedforward neural networks (FNNs), commonly referred to as multilayer perceptrons, are an important class of neural networks. Typically, a feedforward network is composed of an input layer, one or more hidden layers of computation neurons, and an output layer of computation neurons. All these layers are connected in a series structure. The input signal propagates through the network layers in a forward direction. In this section the architecture of FNNs is introduced starting from a single neuron model.

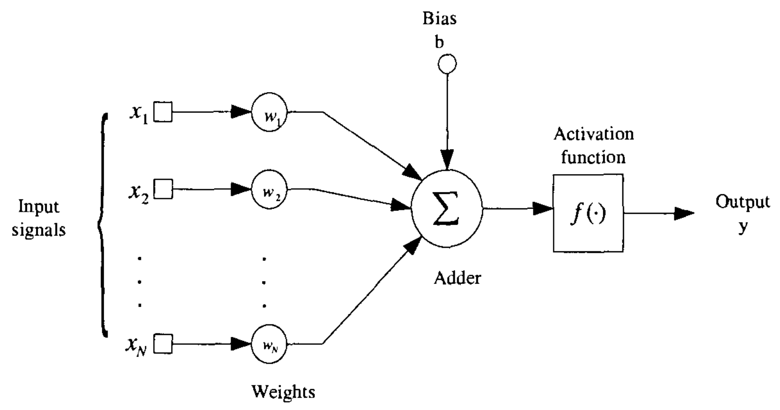

Figure 2.1: A single neuron

A Single Neuron Model

A neuron is the basic information-processing unit of a neural network. Figure 2.1 shows the model of a neuron, which is the elementary building block of a feedforward network. There are three basic elements in the neural model:

1. A set of Links or weights, each of which scales the input signal by a value of a weight and connects to the neuron. Specifically, a signal x_j at the input of link j connected to the neuron is multiplied by the weight w_j.

2. An adder, which sums the input signals weighted by the respective weights of the neuron; the operation described here constitutes a linear combiner.

3. An activation function, which limits the amplitude of the output of a neuron to some finite range. The three basic types of activation functions are the threshold function, the piecewise-linear function, and the sigmoid function.

Besides these three basic elements, another external parameter of the neuron model is the bias denoted by b. The bias increases or lowers the net input of the activation

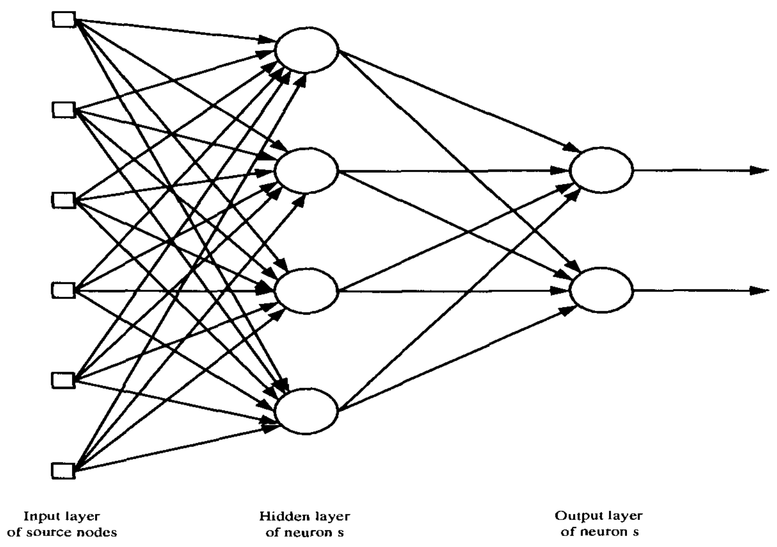

Figure 2.2: An MLP network with one hidden layer and one output layer

function depending on whether it is positive or negative, respectively. In mathematical terms, we may describe a neuron using the equation

$$y = f(\sum_{j=1}^{N} w_j x_j + b) \tag{2.16}$$

where $x_1, x_2, ..., x_N$ are the input signals, $w_1, w_2, ..., w_N$ are the weights of the neuron, b is the bias, $f(\cdot)$ is the activation function, and y is the output signal of the neuron.

Multilayer Perceptrons

The power of a single neuron can be greatly increased by using multiple neurons in a layered and interconnected network architecture. The simplest form of a layered network has an input layer of source nodes that projects onto an output layer of neurons. Such a network is called a *single-layer* network with the term *single-layer* referring to the output layer of computation neurons. The input source node layer is not counted as a layer because it involves no computation. A more complex layered network contains one or more hidden layers of neurons intervening between the external input and the network output. By adding one or more hidden layers, the network is able to extract higher-order statistics [37]. The architectural graph in Figure 2.2 illustrates the layout of a multilayer perceptron network with a single hidden layer. External input signals are connected to the source nodes of the input layer, and are supplied to the neurons in the hidden layer. The output signals of the hidden layer are used as inputs to the output layer. The set of output signals of the output layer constitutes the overall response of the network to the activation signal supplied by the source nodes in the input layer. For brevity, the network in figure 2.2 is referred to as a 6-4-2 network, which means the network has 6 source nodes, 4 hidden neurons, and 2 output neurons.

2.4.2 Kramer's NLPCA Neural Network

A special FNN called the NLPCA neural network, originally introduced by Kramer [38], offers a powerful tool to learn and describe nonlinear subspaces [39] of high-dimensional variables. It has been successfully applied to many complex nonlinear process applications. Specifically, many dynamical problems that appear in chemical engineering are described by nonlinear equations which need to be modelled for study. The human body is also a complicated nonlinear system. The NLPCA network has been applied to analyze medical signals [28], such as Electrocardiograph (ECG) and Electroencephalograph (EEG) signals. Other applications include data compression [22] and pattern recognition [25, 26].

Kramer's NLPCA network, shown in figure 2.3, is a four-layer feedforward MLP network with a bottleneck layer to reduce the dimensionality of the input variables [32]. It is comprised of two symmetric subnetworks called the coding subnetwork and

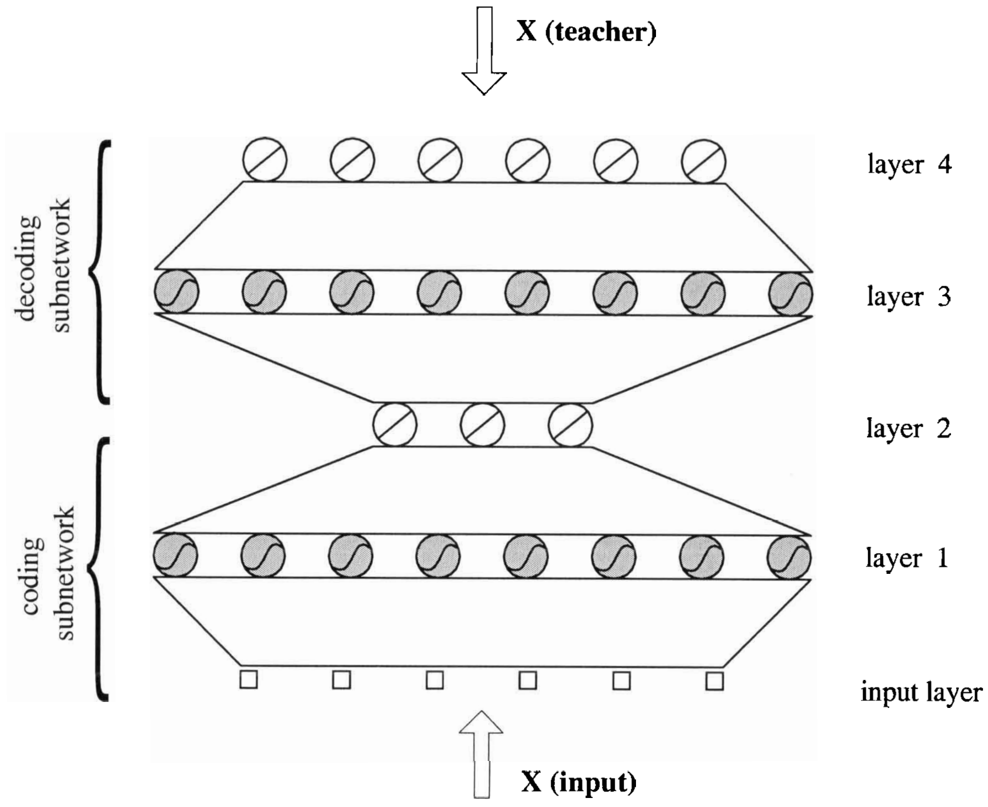

Figure 2.3: An autoassociative nonlinear network performing NLPCA

the decoding subnetwork. The first and second layer constitute the coding subnetwork which maps the input data to the principal component feature space (we follow the convention that the input layer counts as the zeroth layer). The second layer is called the principal component feature layer. It usually has much fewer nodes than the input layer and has the function to extract lower dimensional features of the input data. The third and fourth layer constitute the decoding subnetwork, which reconstructs the output data from the principal component features.

The first layer and the third layer are nonlinear layers. The neurons in these two layers use the nonlinear sigmoid activation function $f(x) = \frac{1}{1+e^{-x}}$. The second layer and the fourth layer are linear layers. It is known that a two-layer neural network as described in figure 2.2 using linear actuation functions in the output layer, and nonlinear sigmoid functions in the hidden layer, can approximate any piecewise continuous function from a closed bounded subset of $R_n \Rightarrow R_p$ with arbitrary accuracy provided that sufficient neurons are used in the hidden layer [40, 41]. That is, the functions of the form

$$\sigma(\mathbf{x}) = \sum_{i=1}^{N} \overline{\mathbf{w}}_i f(\underline{\mathbf{w}}_i^T x + \underline{b}_i) + \overline{\mathbf{b}} \tag{2.17}$$

are universal approximators [32]. Here $\underline{\mathbf{w}}_i$ is the weight vector that links the inputs to neuron i in the hidden layer, $\underline{b}_i$ is the bias of neuron i, $\overline{\mathbf{w}}_i$ is the weight vector that links neuron i to the output layer, and $\overline{\mathbf{b}}$ is the bias vector of the output layer.

Therefore, the coding and decoding subnetwork in figure 2.3 can approximate any continuous bounded function with any degree of accuracy. This is the reason why we need not consider the case where layers 2 and 4 are nonlinear. For simplicity we will assume that both of them are linear.

Suppose the input (layer 0) and output (layer 4) layer have n units corresponding to the n dimensional input observations, and the feature layer (layer 2) has p units

$(p < n)$. The coding subnetwork finds the nonlinear map from the n-dimensional observation space to the p-dimensional feature space. The decoding subnetwork reconstructs the output from the features by minimizing the square error between input and output.

Let $a_i(l)$ be the activation of unit i in layer l, $w_{ij}(l)$ be the synaptic strength of the connection between unit i in layer l and unit j in layer $l-1$, and $\theta_i(l)$ be the bias for unit i in layer l. With this notation we have:

$$a_i(l) = \begin{cases} f(u_i(l)) & : \ if \quad l = 1 \quad or \quad l = 3, \\ u_i(l) & : \ if \quad l = 2 \quad or \quad l = 4, \end{cases} \tag{2.18}$$

where

$$u_i(l) = \sum_{j=1}^{N_{l-1}} w_{ij} a_j(l-1) + b_i(l). \tag{2.19}$$

The number of units in layer l is denoted by N_l. The activation of the zeroth layer is the input $a_i(0) = x_i$, and activation of the fourth layer is the output $a_i(4) = \hat{x}_i$.

The training mode is autoassociative which means the network input is also the teacher of the network output. As a result, the number of neurons in the output layer is always the same as the number of input sources. The target is to minimize the quadratic error between the training samples and the corresponding outputs

$$J_{NET} = \frac{1}{2} \sum_{k=1}^{M} \sum_{i=1}^{N} (a_{i,k}(4) - x_{i,k})^2 \tag{2.20}$$

Here we assume that there are M input training data vectors $\mathbf{x_1}, \cdots \mathbf{x_m}$. The value $a_{i,k}(4)$ is the activation of the output unit i in layer 4 for input training data $\mathbf{x_k}$.

2.5 Illustrative Examples

We use the following example to illustrate the superiority of the NLPCA network to linear PCA in nonlinear random process modelling.

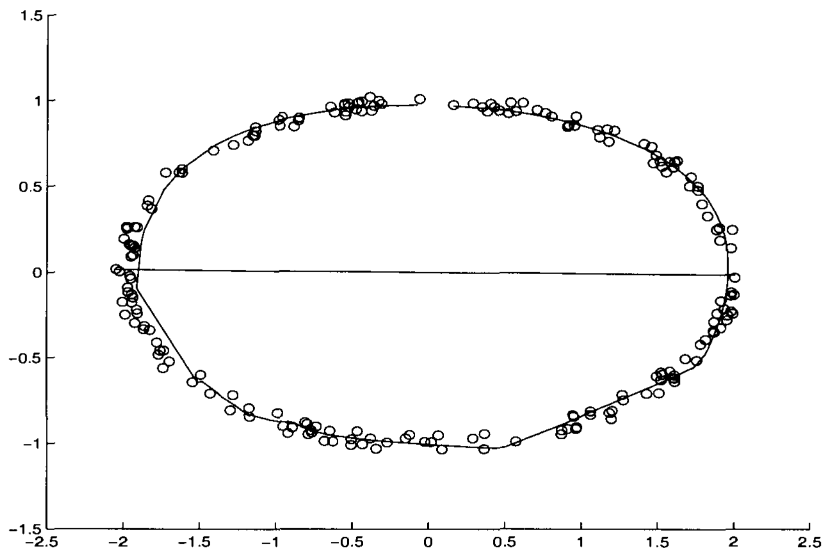

Figure 2.4: Data approximated by NLPCA networks and PCA

Example1.1 Consider a random process with two variables x_1 and x_2 generated by the equations

$$\begin{aligned} x_1 &= 2sin(\phi) + \varepsilon_1 \\ x_2 &= cos(\phi) + \varepsilon_2, \end{aligned} \tag{2.21}$$

where ϕ is the input random variable and is uniformly distributed between $[0, 2\pi]$, and $\varepsilon_1, \varepsilon_2$ are independent uniformly distributed additive noises with amplitudes between $[-0.05, 0.05]$. A NLPCA network is applied to model this process. The process observations x_1 and x_2 are the inputs to the network. Since the process measurements are determined by a single random variable ϕ, the principal component feature layer of the network has size of $N_2 = 1$. The choice of the size of layer 1,3 usually depends on the complexity of the process. Here we choose $N_1 = N_3 = 10$ to form a 2-10-1-10-2 network structure. The coding subnetwork is designed to extract the single hidden factor corresponding to variable ϕ from measurements, and the decoding subnetwork reconstructs the observations from the hidden factor.

Figure 2.4 shows the result of the experiment. The NLPCA network achieves a good approximation of the random process by a elliptic principal component curve. However, if we use the linear PCA model with a single principal component, the result is a straight line on the main axis of the ellipse as shown in figure 2.4. Obviously, this line is a very poor approximation of the process observation. Therefore, it can be concluded that the NLPCA networks are more effective than the standard PCA method in extracting features of nonlinear process data.

2.6 Conclusions

The NLPCA is an extension of the classic linear PCA. It assumes that the mapping from the input data space to the feature space and the mapping from the feature space to the output space could both be nonlinear. Neural networks offer a powerful tool for NLPCA implementation and have been proved to be very successful in extracting the underlying features of highly nonlinear processes. The classical PCA is a purely second-order statistics method, which means it only uses the information contained in the covariance matrix of the process data vector. The NLPCA technique, on the other hand, has the advantage over PCA by taking into account higher order statistics at least implicitly.

The limitations of NLPCA neural networks lie in the constraint that the mapping from the data space to the feature space must be a continuous function. This constraint has some undesirable consequences [42]:

1. the mapping points will be incorrect when the training data are close to any ambiguity point. An ambiguity point is a point that can be mapped to more than one place on the principal surface or curve according to (2.6).

2. Kramer's NLPCA cannot correctly model curves or surfaces that intersect themselves.
3. NLPCA cannot model parameterizations that have discontinuous jumps.

These events may or may not be present in practical problems. Therefore, special attention must be paid to avoid using incorrect results derived from the above situations.

Chapter 3

Multivariate Statistical Process Monitoring

3.1 Introduction

Multivariate statistical process monitoring (MSPM) approaches have evolved in two different directions [45]. One group of methods, such as cumulative sum (CUSUM), and subspace identification methods (SIMs), involve the development of efficient techniques for monitoring the out-of-control status. These methods use the Hotelling T^2 technique to detect the out-of-control status. Once a deviation is detected, other techniques such as univariate Shewhart charts are then used to identify the specific variables that cause the out-of-control status. The second group of methods, called residual-based methods, focuses on the integration of detection and identification. Residuals measure the difference between the observations and the nominal system model outputs, and are commonly used when system models are available. Statistical projection methods, such as principal component analysis (PCA), partial least squares (PLS) and nonlinear principal component analysis (NLPCA) belong to the class of residual-based methods. These methods first develop models describing the process variations under their normal operation conditions (NOCs). These models are then used to detect any deviation from the NOCs.

In the following sections, we'll introduce three multivariate process monitoring methods using SIMs, standard PCA and NLPCA neural network techniques, and compare their performances in linear and nonlinear process monitoring.

3.2 Canonical Variate State Space Analysis Method

Canonical variate analysis (CVA) is one of the most influential SIMs. CVA was introduced by Hotelling in 1936 [43] and applied to dynamic systems by Akaike in 1974 [44] and further developed by Larimore [16, 17]. Negiz and Çinar [46] developed a multivariate statistical process monitoring method based on a canonical variate state space (CVSS) model.

Consider a state space model:

$$\begin{aligned} \mathbf{x}_{k+1} &= A\mathbf{x}_k + B\epsilon_k \\ \mathbf{y}_k &= C\mathbf{x}_k + \epsilon_k, \end{aligned} \tag{3.1}$$

where $\mathbf{x}_k$ is the vector of n state variables at time k, $\mathbf{y}_k$ is the vector of p observations, ϵ_k is the innovation vector with zero mean and covariance $E(\epsilon_k \epsilon_{k+l}^T) = \Delta$ if $l = 0$, and 0 otherwise. System matrices A, B and C, and the covariance matrix Δ are to be determined using CVSS realization methods which try to describe the dynamics of the process with minimum number of state variables.

The CVSS realization procedure starts from a Hankel matrix that expresses the covariance between the vectors of stacked future outputs (Y_{kJ}^{+}) and past outputs (Y_{kQ}^{-}).

$$Y_{kJ}^{+} = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k+1} \\ \vdots \\ \mathbf{y}_{k+J-1} \end{bmatrix} \qquad Y_{kQ}^{-} = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k-1} \\ \vdots \\ \mathbf{y}_{k-Q+1} \end{bmatrix} \tag{3.2}$$

The number of future (J) and past(Q) observations are parameters defined by the user. Q is selected to cover all lagged values of process variables that have significant autocorrelation with current values. J usually set equal to or greater than Q.

The Hankel matrix is defined as:

$$\begin{aligned} H_{JK} &= E\{Y_{kJ}^{+} Y_{k-1Q}^{-T}\} \\ &= E\left\{ \begin{bmatrix} \mathbf{y}_k \\ \vdots \\ \mathbf{y}_{k+J-1} \end{bmatrix} [\mathbf{y}_{k-1}^{T} \cdots \mathbf{y}_{k-Q}^{T}] \right\} \end{aligned} \tag{3.3}$$

The dimension of the state vector is determined by the number of dominant singular values of the Hankel matrix. The scaled Hankel matrix is given by

$$\overline{H}_{JK} = (R_J^{+})^{-\frac{1}{2}} H_{JK} (R_K^{-})^{-\frac{1}{2}}, \tag{3.4}$$

where $(R_J^{+}) = E\{Y_{kJ}^{+} Y_{kJ}^{+T}\}$ and $(R_K^{-}) = E\{Y_{k-1K}^{-} Y_{k-1K}^{-T}\}$ are the scaling factors. The singular value decomposition of the scaled Hankel matrix is

$$\overline{H}_{JK} = U \Sigma V^{T}. \tag{3.5}$$

The state variables can be expressed as

$$\mathbf{x}_k = \Sigma^{1/2} V^{T} (R_k^{-})^{-1/2} Y_{k-1K}^{-}. \tag{3.6}$$

Once the state variables are computed, the system matrices A, B and C, and the covariance matrix Δ can be determined [46]. The CV state variables obtained from

(3.6) are linear combinations of the past process measurements that are best suited to predict the variability of the future measurements. Therefore, the state variables computed from the CVSS model can be used to describe the variation of the process based on the T^2 statistics:

$$T^2 = (\mathbf{x} - \mu_{\mathbf{x}})^T C_x^{-1} (\mathbf{x} - \mu_{\mathbf{x}}), \tag{3.7}$$

where μ_x is the mean value of $\mathbf{x}$, and C_x is the estimation of Σ. By monitoring the magnitude of the T^2 statistics, process deviations from in-control conditions can be detected.

3.3 PCA Method

PCA is a statistical modelling technique which seeks to find a few linear combinations of the process variables that can be used to summarize the process variations with a minimal loss of information.

Consider an $m \times n$ matrix $X = [\mathbf{x_1}, \ldots, \mathbf{x_m}]^T$ with each row containing a vector of observations of dimension n. Each observation is assumed to be a normally distributed random variable with zero mean and unit variance. The covariance matrix C_x of X is defined as:

$$C_x = \frac{X^T X}{m - 1}. \tag{3.8}$$

PCA decomposes X as follows:

$$X = \hat{X} + E, \tag{3.9}$$

where $\hat{X}$ is the model prediction and E is the prediction error matrix. The estimated $\hat{X}$ is given by:

$$\hat{X} = SW^T, \tag{3.10}$$

where W is an $n \times p$ matrix. Its columns are the eigenvectors corresponding to the p largest eigenvalues of the covariance matrix which are also referred to as the p principal directions of the process data. S is the principal component score matrix and is given by:

$$S = XW. \tag{3.11}$$

Given a new sample observation $\mathbf{x}$, the PCA model prediction is calculated as:

$$\hat{\mathbf{x}} = \mathbf{x}WW^T, \tag{3.12}$$

and the prediction error $\mathbf{e}$ is calculated using:

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}. \tag{3.13}$$

The residual r corresponding to sample $\mathbf{x}$ is given by:

$$r = \mathbf{e}\mathbf{e}^T. \tag{3.14}$$

To perform process monitoring, a PCA model is first built using historial data collected durring normal process operation. Principal directions of the data are found as discussed above. The PCA model is then used for data prediction by projecting the on-line data onto the model. Residual r at each sample time is estimated by (3.13, 3.14). Whenever the residual exceeds a certain threshold, a fault alarm signal is generated.

3.4 NLPCA Neural Network Method

Most industrial processes with a large number of process variables are nonlinear processes and are usually operated at fixed points for extended periods. The monitoring method designed here is based on Kramer's NLPCA network model of the process.

Process data from the normal operating condition are normalized and used to train the parameters of the networks.

We use the Levenberg-Marquardt back-propagation (BP) algorithm to iteratively adjust the weights and biases of the network to minimize the objective function J_{NET} defined in (2.20).

Standard back-propagation is a gradient descent algorithm. The term *backpropagation* refers to the manner in which the gradient is computed for nonlinear multilayer networks. Gradient-descent-based algorithms have a first-order convergence characteristic and usually are too slow for practical problems.

A useful way to improve the performance of training is by using second-order convergence based algorithms such as Newton's method. The basic step of Newton's method is:

$$W_{k+1} = W_k - \lambda H_k^{-1} \frac{\partial J}{\partial W_k}, \tag{3.15}$$

where W_k is the current parameter matrix of the network, H_k is the Hessian matrix of second derivatives of the objective function J defined by (2.20) at the current values of weights and biases, and λ is the step size.

Unfortunately, it is usually too expensive to compute the Hessian matrix for feedforward neural networks at each iteration. There are a class of algorithms that are based on Newton's method but don't require calculations of second derivatives. These are called quasi-Newton methods. They update an approximate Hessian matrix at each iteration of the algorithm. The most successful quasi-Newton method published is the BFGS (Broyden, Fletcher, Goldfarb and Shanno) update.

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the Hessian matrix is symmetric and positive definite (which is most often the case in feedforward networks where the objective function J has the form of a sum of squares), it can be approximated as:

$$H_k = \frac{\partial^2 J}{\partial {W_k}^2} = \left[\frac{\partial \hat{\mathbf{x}}}{\partial W_k}\right] \left[\frac{\partial \hat{\mathbf{x}}}{\partial W_k}\right]^T, \tag{3.16}$$

where $\hat{\mathbf{x}}$ is the output of the network.

The Levenberg-Marquardt algorithm uses this approximation of the Hessian matrix but also introduces a factor $\mu \geq 0$ into the matrix, i.e.:

$$W_{k+1} = W_k - \lambda \left[\left(\frac{\partial \hat{\mathbf{x}}}{\partial W_k}\right) \left(\frac{\partial \hat{\mathbf{x}}}{\partial W_k}\right)^T + \mu I\right]^{-1} \frac{\partial J}{\partial W_k}. \tag{3.17}$$

When μ is zero, this is simply Newton's method using the approximate Hessian matrix. When μ is large, this becomes the gradient descent method with a small step size.

Newton's method is faster and more accurate near the minimizer, so the strategy is to shift towards Newton's method as quickly as possible. Thus, μ 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 objective function will always be reduced at each iteration of the algorithm.

The basic steps used to train a NLPCA network model using Levenberg-Marquardt method involve [22]:

1. Set initial values W_0 and a large start value of μ
2. Calculate gradient $\frac{\partial J}{\partial W_k}$ and $\frac{\partial \hat{\mathbf{x}}}{\partial W_k}$
3. Update W_k using (3.17)
4. Is $|J(W_{k+1})| \leq |J(W_k)|$?

   Yes: decrease μ for fast search in Newton direction;

   No: increase μ, search in the gradient descent direction.

5. If $|J(W_{k+1})| \leq \epsilon$, the target reached, then stops, otherwise goes to (2).

Once the NLPCA network model of the process is obtained, we can then project the on-line data $\mathbf{x}$ onto the model and measure the model predictions $\hat{\mathbf{x}}$. The residual corresponding to each observation is estimated by:

$$r = \parallel \mathbf{x} - \hat{\mathbf{x}} \parallel^2 . \tag{3.18}$$

Whenever an abrupt change in the amplitude of the residual is observed, a fault condition is declared.

3.5 Statistical Process Monitoring Illustrative Examples

Two examples are used to illustrate the performance of the CVSS method, the PCA method, and the NLPCA method in statistical monitoring of linear and nonlinear processes.

Example 1

Consider a linear auto-regressive process with two variables y_1, y_2 described as:

$$\begin{aligned} y_1(k) &= \phi y_1(k-1) + \epsilon_1(k) \\ y_2(k) &= 0.5y_1(k-1) + 0.15y_2(k-1) + \epsilon_2(k), \end{aligned} \tag{3.19}$$

where integer k denotes the current time index, and $\epsilon_1(k)$ and $\epsilon_2(k)$ are independent zero mean Gaussian noises with standard deviation 0.1. The system parameter $\phi = 0.2$ under normal conditions. The system initially operated normally and a sufficient number of samples of y_1 and y_2 were recorded to build the system model. Then a change in the dynamics was induced by increasing the value of ϕ from 0.2 to 0.5 at 200 sample time and lasted for 100 samples. Figure 3.1 plots the adjacent 500 samples of the PCA residuals, the NLPCA network residuals and the CVSS T^2 statistics in response to this change. Both the PCA and the NLPCA method have one principal component, and the NLPCA network has the size of 2-2-1-2-2.

Figure 3.1a shows that the PCA method using one principal component detects the changes in the system; however, it is significantly affected by the process noises, while the CVSS (Figure 3.1b) method, and the NLPCA (Figure 3.1c) method, both show good sensitivity to the change in the dynamics and are also robust to system noise.

Example 2

Consider a two-dimensional nonlinear dynamical system [47]

$$\begin{aligned} y_1(k) &= \theta y_1(k-1)\cos(y_1(k-1)) - 0.1y_1(k-1)^2 y_2(k-1) + 0.8y_2(k-1) + \epsilon_1(k) \\ y_2(k) &= -0.2y_1(k-1)\sin(y_2(k-1)) + 0.7y_2(k-1) + \epsilon_2(k), \end{aligned} \tag{3.20}$$

where $\epsilon_1(k)$ and $\epsilon_2(k)$ are independent zero mean Gaussian noises with a standard deviation of 0.1. The system parameter θ has the value of 0.1 under its normal operating condition. A change of θ from 0.1 to 0.5 was induced at 200 sample time and lasted for 100 samples. The process drifted away from its normal region, and returned after θ returned to 0.1. Figure 3.2 shows the responses of the PCA, NLPCA residuals, and CVSS T^2 statistics to this change. In this example, both the PCA and the NLPCA methods have one principal component, and the NLPCA network has the size of 2-15-1-15-2.

In this example, the PCA method cannot detect the change in this nonlinear process. CVSS T^2 statistics give an indication of the system change; however, the

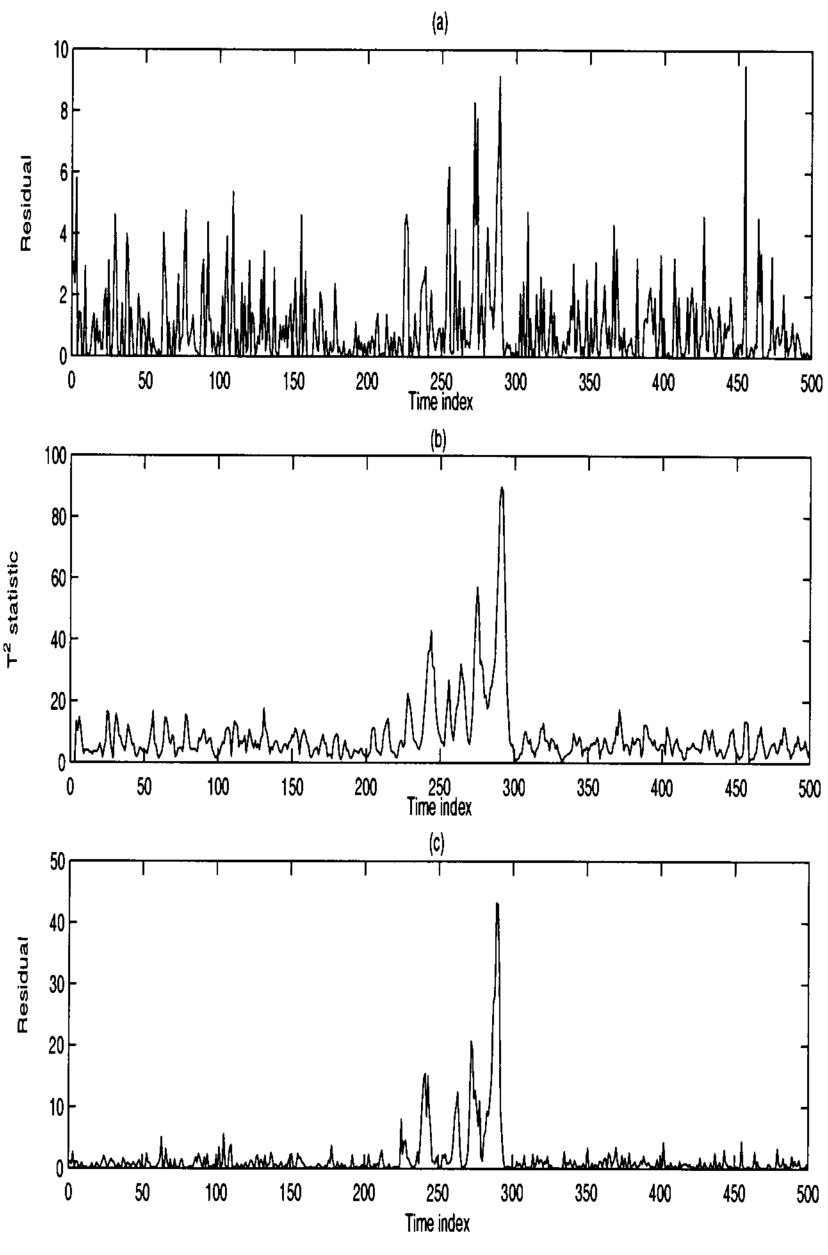

Figure 3.1: Monitoring the dynamic change of a linear process
(a)Residuals of PCA method; (b)T^2 statistic of CVSS method; (c)Residuals of NLPCA method

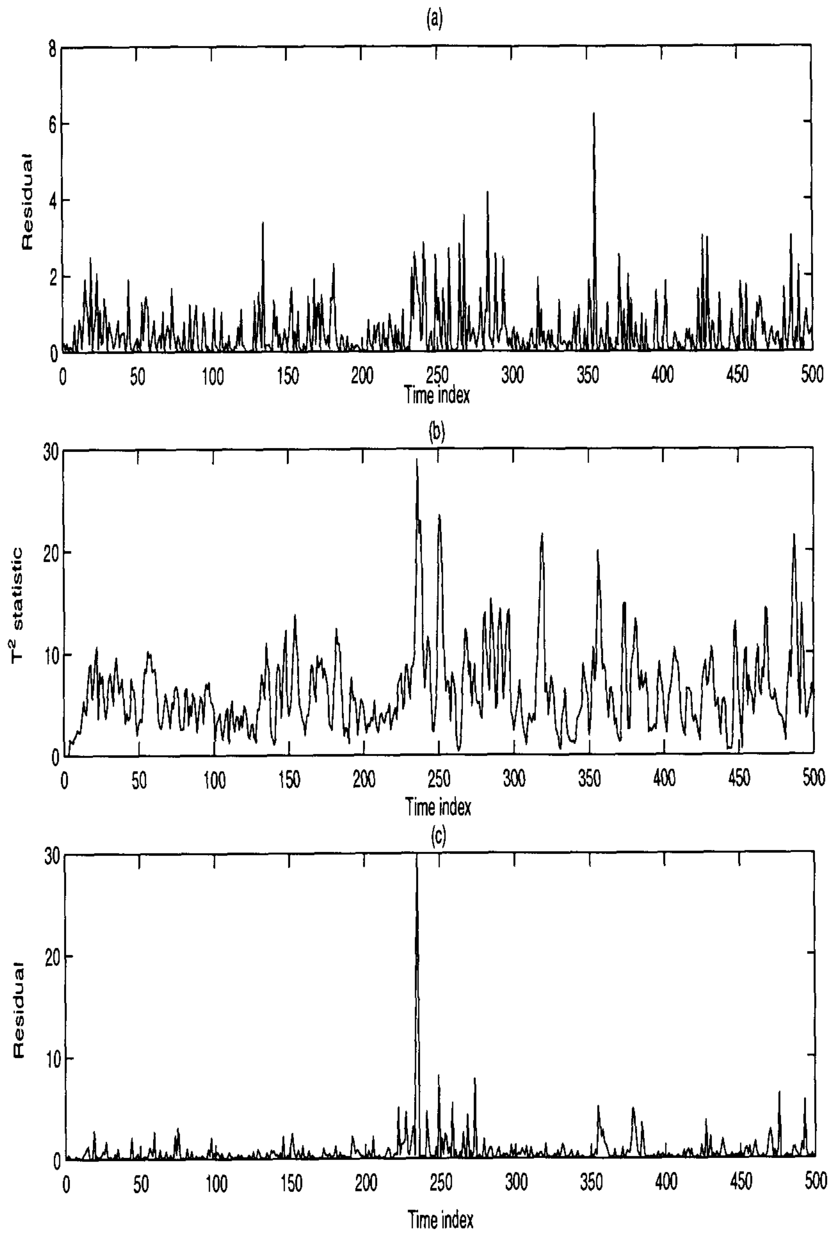

Figure 3.2: Monitoring the dynamic change of a nonlinear process
(a)Residuals of PCA method; (b)T^2 statistics of CVSS method; (c)Residuals of NLPCA method

signal to noise ratio is very low. NLPCA residuals give a good indication of the dynamic change and is robust in the presence of noise.

3.6 Conclusions

SIMs-based MSPM methods have certain features in common with that of PCA in that they all use covariance and cross-covariance information to determine the number of variables for building system models [48]. The limitation of these methods is that they assume the state variables, such as in the CVA method, or principal component features in the PCA method, are linear functions of the observations. Therefore, when the measurements are from highly nonlinear processes, these methods cannot efficiently extract the process information. NLPCA neural networks address this limitation by using nonlinear mapping functions, and treat linear dynamics as special cases when the mapping functions are linear. Simulation results show that NLPCA neural networks can detect changes in both linear and nonlinear dynamics and are robust against system noise.

Chapter 4

Actuator Benchmark Test

In this chapter, static nonlinear principal component analysis (NLPCA) neural networks are extended to model dynamic systems using a technique called time-delay neural networks (TDNNs). These networks are used to detect faults in a diesel engine actuator benchmark system, and the performance is compared with a frequency domain analytical model-based method.

4.1 The Diesel Engine Actuator Benchmark

The industrial diesel engine actuator benchmark is based on an electro-mechanical test facility which has been built at Aalborg University in Denmark [49]. The equipment simulates a part of a speed governor for large diesel engines. The governor is a device that controls the shaft speed of a diesel engine. It regulates the amount of fuel loaded into each cylinder by controlling the position of a common control rod which can be moved by an actuator motor. The current and velocity of the actuator motor are controlled by a power drive, while the position of the rod is controlled by a microprocessor based controller.

Table 4.1: Data sequences representing different models

Data Sequence No.	Description
1	Linear model, small signals, no noise.
2	Soft nonlinear model, small signals, no noise.
3	Complex nonlinear model, small signals, noise.
4	Complex nonlinear model, large signals, noise.

Simulink files provided by [49] for the diesel actuator benchmark system include nonlinear and linearized plant models, and can simulate various generic fault scenarios in electronics and other hardware. In this study, the following two realistic events are considered:

1. A sensor fault in a feedback element as a fault in the position measurement ΔS_0 (f_s). The wiper of a feedback potentiometer loses contact with the resistance element due to wear or dust. The particular fault might cause an over-speed of the diesel engine with a shut down of the engine as the result. The fault is intermediate, and lasts for only 0.2s.

2. A component fault in Δi_m (f_a). An end-stop switch suddenly malfunctions because of a broken wire or a defect in the switch element due to heavy mechanical vibration. As a result, the power drive can only deliver positive current.

The generally available signals that can be used for FDI are velocity reference n_{ref}, velocity measurement n_{mv}, and position measurement s'_{ov}. The data provided with the actuator benchmark consist of four data sets (see table 4.1), where each set is characterized in terms of the system model and signal conditions that were used to produce them. The sequences comprise a position fault, a current fault, and load disturbance.

The main task is to design algorithms that enable detection and isolation of the two faults which are also robust to load changes, noise, and model uncertainty.

4.2 Frequency Domain Analytical Model-based FDI Approach

A frequency-domain FDI approach for the actuator benchmark study is presented by E.A.García [50]. The FDI method designed here is based on the analytical linear model of the benchmark system. This approach offers powerful methods to tackle the robustness problem by using H_∞ theory.

The basic idea of the frequency domain approach is the following [50]: through a frequency characterization of all achievable residual dynamics, based on Coprime stable factorizations and Youla parametrization, the robust residual generation design is formulated as an optimization problem and solved by the H_∞ technique.

Consider a linear time-invariant system with the state equations

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) + E_d\mathbf{d}(t) + F_a\mathbf{f}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) + F_d\mathbf{d}(t) + F_s\mathbf{f}(t) \end{aligned} \tag{4.1}$$

with output $\mathbf{y} \in R^m$, input $\mathbf{u} \in R^p$, unknown input $\mathbf{d} \in R^s$, fault $\mathbf{f} \in R^q$, and matrices $A, B, C, D, E_d, F_a, F_s$ and F_d of proper dimensions. The input-output relation in the frequency domain is described as

$$\mathbf{y}(s) = G_u(s)\mathbf{u}(s) + G_f(s)\mathbf{f}(s) + G_d(s)\mathbf{d}(s), \tag{4.2}$$

where $G_u(s)$, $G_d(s)$ and $G_f(s)$ are the transfer matrices from $\mathbf{u}$, $\mathbf{d}$, $\mathbf{f}$, respectively, to the output $\mathbf{y}$ of the system.

The detailed procedure to build residuals was given in [51], and is summarized below:

1. Do coprime factorization of $G_u(s)$

$$G_u(s) = \hat{M}_u^{-1}(s)\hat{N}_u(s). \tag{4.3}$$

2. Find $Q_1(s)$ such that $Q_1(s)\hat{M}_u(s)G_f(s) = diag(T_i)$.

3. Do an extended inner-outer factorization of $Q_1(s)\hat{M}_u(s)G_d(s)$

$$Q_1(s)\hat{M}_u(s)G_d(s) = G_{do}(s)G_{de}(s)G_{di}(s) \tag{4.4}$$

4. Find the Smith-McMillan form $SM\{G_{de}(s)\}$ of $G_{de}(s)$.

5. Set $Q_2(s)$ so that $\| Q_2(s)SM\{G_{de}(s)\} \|_\infty \leq 1$

6. Set P(s) as :

$$P(s) = Q_2(s)U_{de}^{-1}(s)G_{do}^{-1}(s)Q_1(s) \tag{4.5}$$

   $U_{de}(s)$ is defined through $G_{de}(s) = U_{de}(s)SM\{G_de(s)\}V_{de}(s)$.

7. Compute the residual by:

$$\mathbf{R}(s) = P(s)(\hat{M}_u(s)\mathbf{y}(s) - \hat{N}_u\mathbf{u}(s)). \tag{4.6}$$

The continuous time state space description for the industrial actuator system is written in the standard form of 4.1 and the matrices are given by:

$$A = \begin{bmatrix} 0 & -102 & 0 \\ 181 & -171 & 0 \\ 0 & 0.0112 & 0 \end{bmatrix}; \tag{4.7}$$

$$B = \begin{bmatrix} 102 \\ 163 \\ 0 \end{bmatrix}; \qquad E_d = \begin{bmatrix} 0 \\ 4.44 \\ 0 \end{bmatrix}; \tag{4.8}$$

$$F_a = \begin{bmatrix} 0 & 0 \\ 181 & 0 \\ 0 & 0 \end{bmatrix}; \qquad C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0.978 \end{bmatrix}; \tag{4.9}$$

$$D = 0; \qquad F_d = 0; \qquad F_s = \begin{bmatrix} 0 & 0 \\ 0 & 0.978 \end{bmatrix}. \tag{4.10}$$

The state $\mathbf{x}$ is defined as

$$\mathbf{x} = \begin{bmatrix} i_2 \\ n_m \\ s_o \end{bmatrix}, \tag{4.11}$$

where i_2 is the velocity controller integral variable, n_m is the motor shaft velocity and s_o is the gear output position. and the fault vector $\mathbf{f}$ is defined as

$$\mathbf{f} = \begin{bmatrix} f_a \\ f_s \end{bmatrix}, \tag{4.12}$$

where f_a is a component fault in Δi_m, and f_s is a sensor fault in Δs_o. The input velocity reference is $u = n_{ref}$ and the measurement is $\mathbf{y} = [n_m s'_o]$, where s'_o is the position measurement.

The frequency domain residual based on [50] is given by :

$$R_a(s) = \frac{45s^3 + 7707s^2 + 836228s}{(s+100)^3} y_1(s) - \frac{7346s^2 + 836228s}{(s+100)^3} u(s) \tag{4.13}$$

$$R_s(s) = \frac{s^3 + 171s^2 + 18554s}{(s+100)^3} y_2(s) - \frac{0.0106s^2 + 1.8s + 196.7}{(s+100)^3 s} y_1(s). \tag{4.14}$$

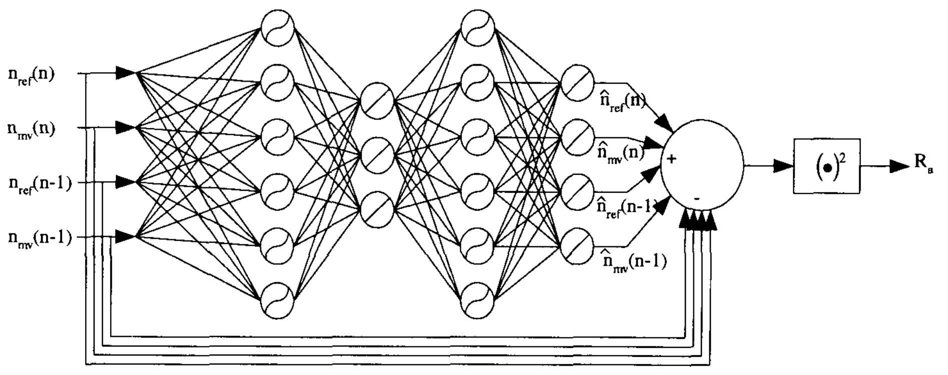

Figure 4.1: Schematic of neural network NN_a

4.3 Time-Delay NLPCA Neural Network-Based FDI Approach

Time-delay neural networks incorporate time-delay elements along their interconnections, and can be trained by back-error propagation. Time-delay neural networks are powerful tools for realizing spatiotemporal relationships between process variables.

The idea of using the time-delay neural network to model the dynamics of the diesel engine actuator system is based on the considerations that only a small number of process variables are available and spatial information between the process data may not be enough to model the system. Besides, the system involves a feedback loop. Therefore, the network designed to model the actuator system should have the ability to learn temporal relationships between the current data and the past history of events as well as the ability to learn spatial relationships between process data.

In this work, two time-delay NLPCA neural networks are designed to detect the actuator fault f_a and the position fault f_s respectively. The schematic architectures are illustrated in figure 4.1 and figure 4.2. Here the velocity reference n_{ref}, velocity

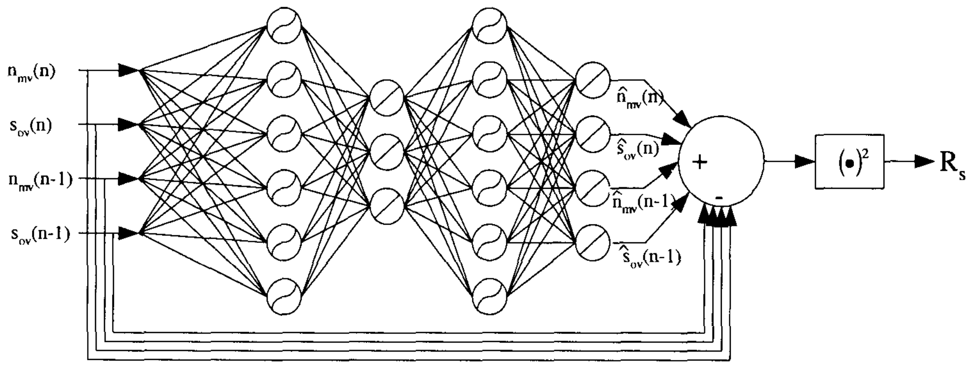

Figure 4.2: Schematic of neural network NN_s

measurement n_{mv}, and their one-step delay values are selected as the inputs to the neural network NN_a for actuator fault f_a detection (see figure 4.1). The velocity measurement n_{mv}, the position measurement s'_{ov}, and the corresponding one-step delay of these variables are the inputs of neural network NN_s to detect sensor fault f_s (see figure 4.2). Both NLPCA neural networks have the size of 4-6-3-6-4.

4.3.1 Simulation results

In this section, simulation studies are given to show how the two FDI approaches work for the detection and isolation of single faults occurring in the diesel engine actuator benchmark system. The time-delay NLPCA neural networks are obtained using data generated by the nonlinear Simulink model of the system under normal conditions with Gaussian noise and load disturbances.

The data sequences for simulation studies are described in table 4.1. The times of fault events and load step disturbances occurring in the system are listed in table 4.2. The sampling period is 0.01 sec.

Figure 4.3 shows the trends of frequency domain residuals R_s and R_a with four data

Table 4.2: The time of fault events and load step disturbances

Event	Start time	End time
Position fault	0.7s	0.9s
Load input	1.2s	2.3s
Current fault	2.7s	3.0s

sequences. Residual R_s clearly indicates that a position sensor fault has occurred at time 0.7 second in all four sequences. Residual R_a, however, is highly affected by load disturbances (data sequences 1, 2, 3, 4), and fails to detect the fault when nonlinear uncertainties are present in the system (data sequence 4). Therefore, the linear model with fixed threshold will generate false alarms for actuator fault f_a.

Figure 4.4 shows the residual plot for the time-delay NLPCA neural network approach. Residual R_s generated by neural network NN_s gives a good indication of the sensor fault f_s, and yet is completely insensitive to the position fault f_a in all four data sequences. It is more robust against system disturbances and noise than its frequency domain analogy. The result suggests that with a predefined threshold, neural network NN_s can perform effective detection and isolation of position sensor fault. Residual R_a from neural network NN_a clearly indicates the actuator fault when the system is soft-nonlinear or nonlinear (data sequences 2, and 3). Its performance decreases for data from linearized model (data sequences 1). The neural network fails to detect the actuator fault in the hard nonlinear system with large signals (data sequence 4).

4.4 Conclusions

The results reported in this chapter show that the time-delay nonlinear principal neural network based method is a viable approach to detect sensor and motor drive

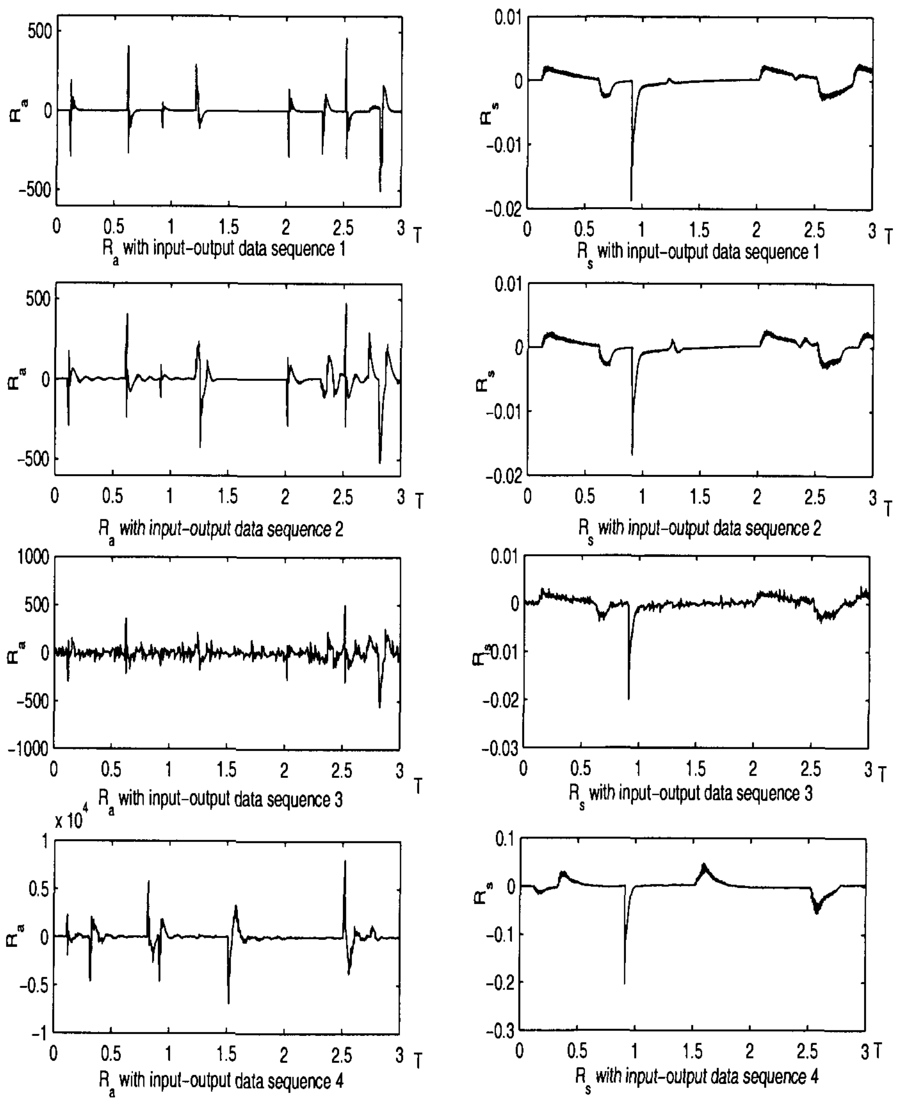

Figure 4.3: Residual generated by frequency domain approach

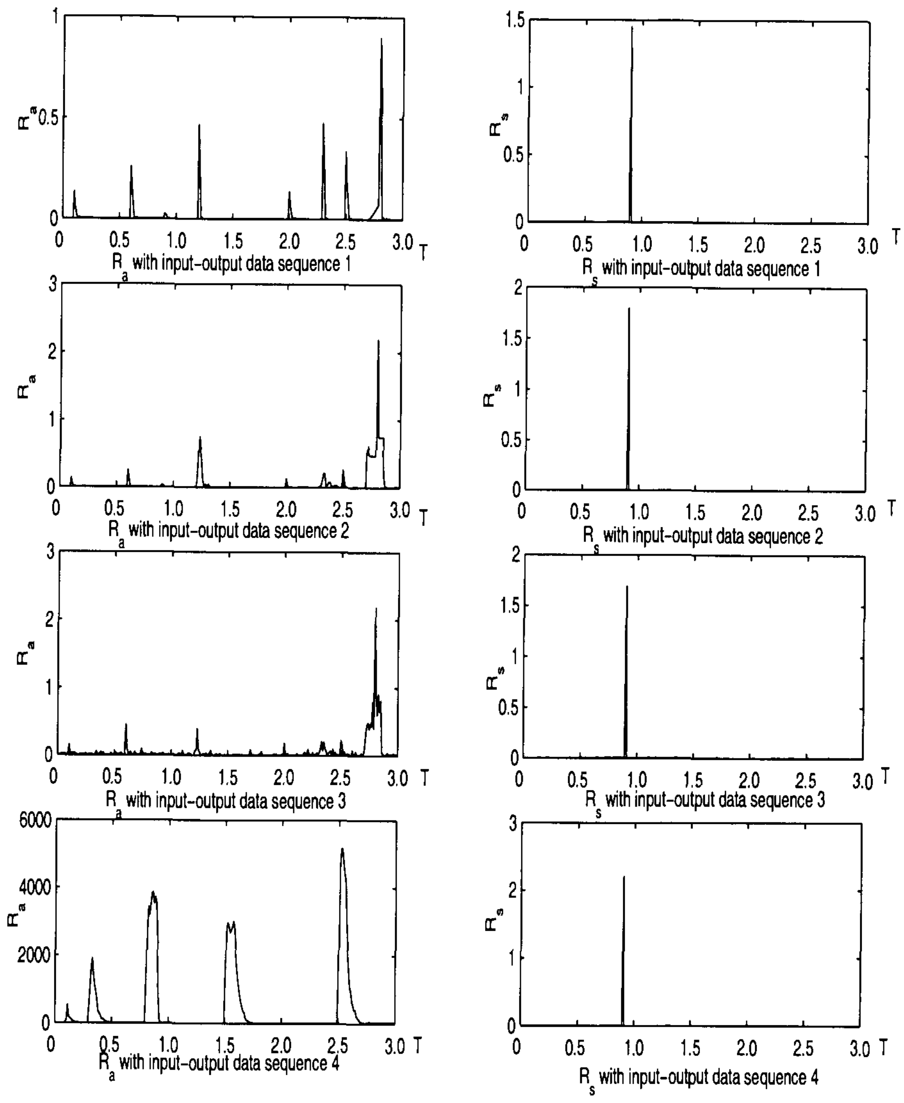

Figure 4.4: Residual generated by time-delay NLPCA neural networks

failures in the industrial actuator benchmark problem. Unlike the frequency domain method, the neural-network-based method does not need a mathematical model of the system, though at the price of a large set of training data. Furthermore, by selecting different groups of process variables as inputs to the neural network based on knowledge of the system, the NLPCA neural networks with TDNNs can be designed to be sensitive to different faults and thus make fault isolation possible.

Chapter 5

Statistical Fault Identification

5.1 Introduction

Although MSPM using statistical correlation models are very effective for fault detection, fault isolation is more difficult with this approach. The main strategy for fault isolation is the use of contribution plots. Once a fault is detected by a residual statistic, the contributions of individual variables to the residual statistic can be calculated and those variables having large contributions are examined to indicate possible causes. Contribution plots do not provide direct fault isolation. They only show which group of variables are highly correlated with the fault, and it is up to the engineers to use their process knowledge to provide feasible interpretations.

To do more, one needs to have additional information. Several approaches using prior fault histories have been proposed [24, 52]. These approaches all use some form of fault features, of which the most important are the directions of the process movement in the principal component space [24], or in the square-prediction-error space [52]. Once a new fault occurs, its features are compared with those in the fault bank in order to identify the most likely cause. Comparing the new fault with the fault bank and identifying the fault usually depends on the experience of the person reviewing the data.

Fault identification methods based on feature visualization are suitable for simple processes with a small number of variables and a small set of potential faults to be considered. As we know, different faults in a process may cause the process to drift away from its normal state and move along different trajectories until it reaches new equilibrium states. For simple processes, with a small number of variables, these trajectories may cover distinct regions in the feature space and can be easily identified by human eyes. However, for complex processes with a large number of potential faults, different faults usually cover regions that are highly overlapped and when the size of faults are small, they are most likely buried within the nominal region which makes them difficult to separate from one to another (see figure 5.1 [62]). Furthermore, depending on the magnitude of fault, the direction of the movement corresponding to the same fault may not be consistent. Figure 5.1 shows such a case where the principal component plot of a chemical process goes through different trajectories in response to different levels of clog in a valve [62]. In fact, for complicated multivariate processes, it is not effective to identify various process faults relying on human visualization of their principal component plots.

To overcome the limitation of the visualization-based approach, a fault identification system has been developed based on statistical fault distribution modelling. The difference between the fault visualization and the fault-distribution-modelling-based approach is that the former relies on human vision, while the latter is a computer-aided approach. The fault identification system is operated in two stages: a learning stage followed by an on-line working stage. In the learning stage, the system learns the distributions of various types of known faults in the feature space using prior fault

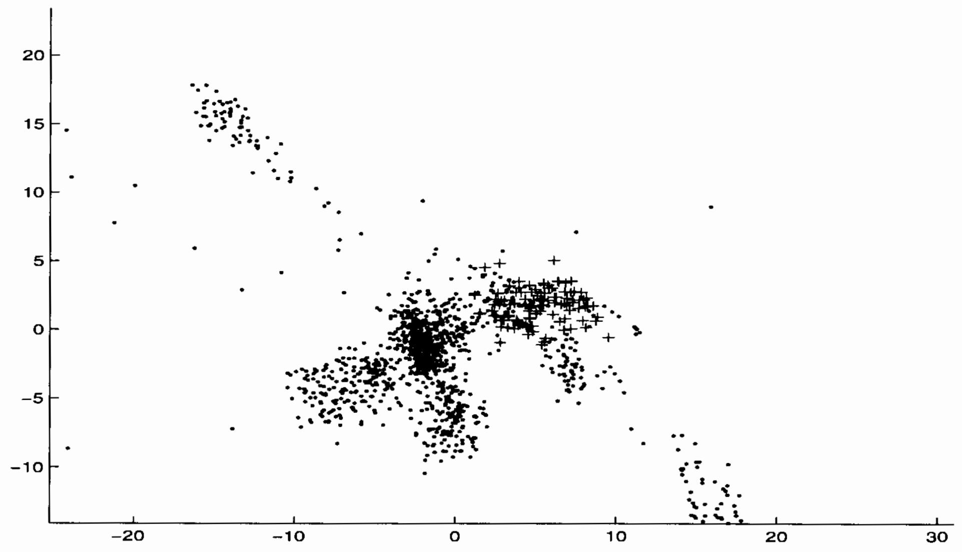

Figure 5.1: Principal component plot of a chemical process under normal and fault conditions
+ = nominal data, . = data from various fault conditions

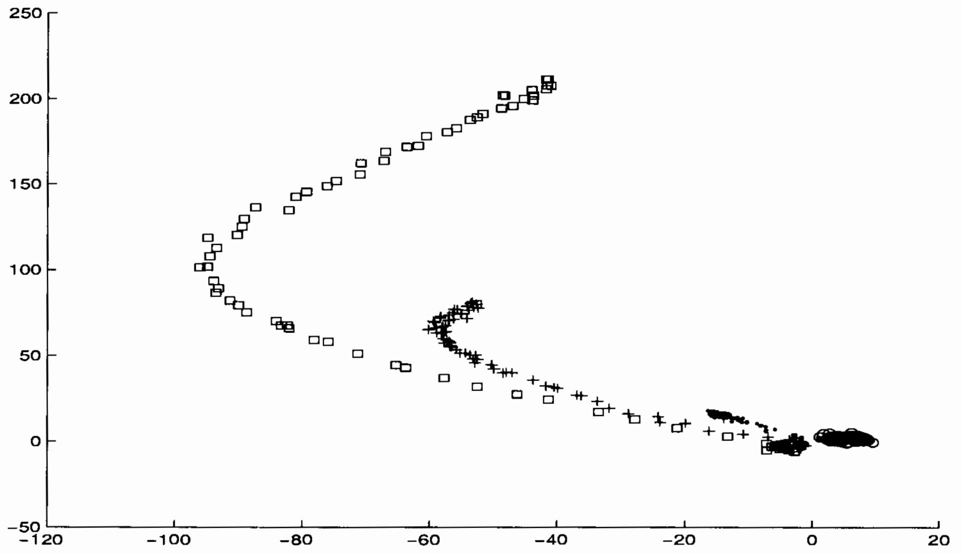

Figure 5.2: Principal component plots of a chemical process movement in response to different levels of clog in a valve
$o = nominaldata$, . = 10% clog, + = 50% clog, □ = 90% clog

histories and builds fault distribution models. This is accomplished through the use of self-organizing feature mapping (SOFM) and learning vector quantization (LVQ) techniques which will be discussed fully in subsequent sections. In the on-line working stage, unknown on-line faults are recognized by the system using the fault models built in the learning stage. The following sections introduce the fault identification system, its two operating stages, and the underlying techniques.

5.2 Fault Distribution Modelling

During the learning stage, the fault identification system builds fault distribution models with historical database record of prior faults. The information for fault modelling consists of statistical distribution of process data in the nonlinear principal component feature space during the period immediately following its detection of the fault. As mentioned before, different faults in a process may cause the process to drift away from its normal state and move along different trajectories. In other words, from the moment a fault occurs, the process operating trajectory contains some unique information about this fault. Therefore, by comparing the distribution patterns of these trajectories in the feature space, different faults can be identified. Distribution patterns in the feature space are analyzed using the techniques of self-organizing feature mapping (SOFM) and learning vector quantization (LVQ).

Figure 5.2 shows the main steps involved in the process of building fault distribution models. The first step involves extracting low-dimensional features that exist in the high-dimensional process data. The second step involves learning the distribution topography of fault data in the feature space using SOFM. In the third step, features are encoded and fault distribution models are calculated. To improve the quality of classification, distances between different fault distributions are measured using the

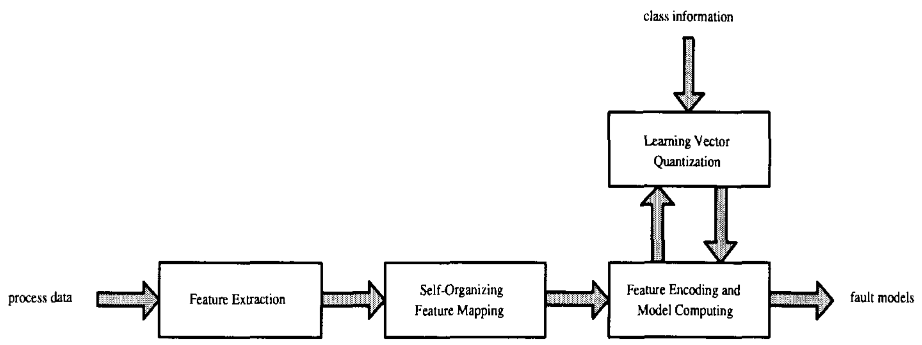
class information
Learning Vector Quantization
process data
Feature Extraction
Self-Organizing Feature Mapping
Feature Encoding and Model Computing
fault models

Figure 5.3: Block diagram of fault modelling procedure

Kullback-Leibler divergence, and LVQ is applied to improve fault models that have similar distributions.

5.2.1 Feature Extraction

In order to build a fault distribution model, historical process data are analyzed to determine the fault mechanisms. The onset of the fault is first located using the detection techniques introduced in chapter 3, then a suitable length of the subsequent process data are collected to build the model of the fault.

It is assumed that the process data contain a large amount of information for fault classification in the sense that different types of faults may have different distribution patterns. However, a variety of obstacles stand in the way of firming up this vague statement into a practical classification system. To begin with, the high cross-correlation between the process variables will make any classification inaccurate. Therefore, the first step to fault identification is to reduce the dimensionality of the process data and form a manageable set of uncorrelated variables using Kramer's NLPCA networks introduced in chapter 2. The nonlinear principal components of the process data at each sampling instant are calculated at the second layer of the NLPCA

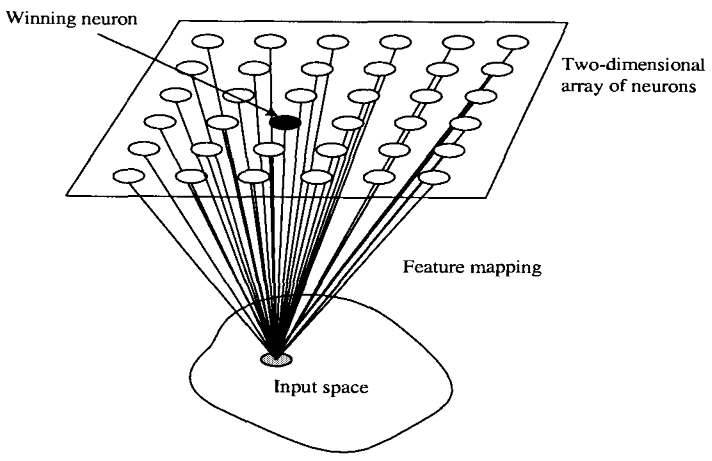

Figure 5.4: Kohonen model

networks, and form a new set of features that contain the most important information of the process variety. These features are then used to build the corresponding fault model.

5.2.2 Self-Organization Feature Mapping (SOFM)

The study of the human brain [53] has revealed that the brain keeps each incoming piece of information in its proper context, and neurons dealing with closely related information are clustered together so that they can interact via short synaptic connections. The principal of the topographic map formation in the brain may be stated as [54]:

The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature of data drawn from the input space.

Inspired by this neurobiological structure, Kohonen [55] introduced the SOFM model shown in Figure 5.4. Kohonen's model maps features from the input space to a two-dimensional array of neurons. Each neuron in the array represents a certain region of the input space. When a feature is within a region in the input space, the corresponding neuron is called a winning neuron. This model has received intensive attention in the literature and most SOFM algorithms are derived based on this model.

The SOFM introduced by Kohonen [54, 55, 56, 58] was originally designed for visualization of high-dimensional data. It converts nonlinear relationships between high-dimensional data into simple geometric relationships on a low-dimensional display which is usually a regular two-dimensional grid of nodes.

Kohonen's SOFM also belongs to the class of vector-coding algorithms, since SOFM provides a topological mapping that optimally places a fixed number of vectors into a high dimensional input space which will be demonstrated later in chapter 6, and thereby facilitates data compression [37]. In this work, the SOFM is used to perform vector quantizing and coding of the process features obtained in section 5.2.1. Fault visualization can also be achieved using the SOFM networks which will be introduced in the following section.

SOFM networks

The SOFM networks have a feedforward structure with a single computational layer consisting of neurons arranged in rows and columns. Figure 5.5 shows the schematic diagram of a two-dimensional lattice of neurons applied in this work. Each neuron in the array is fully connected to the input nodes; therefore, the weight vector $\mathbf{w}$ of each neuron has the same dimension as the input space, and is called the reference vector.

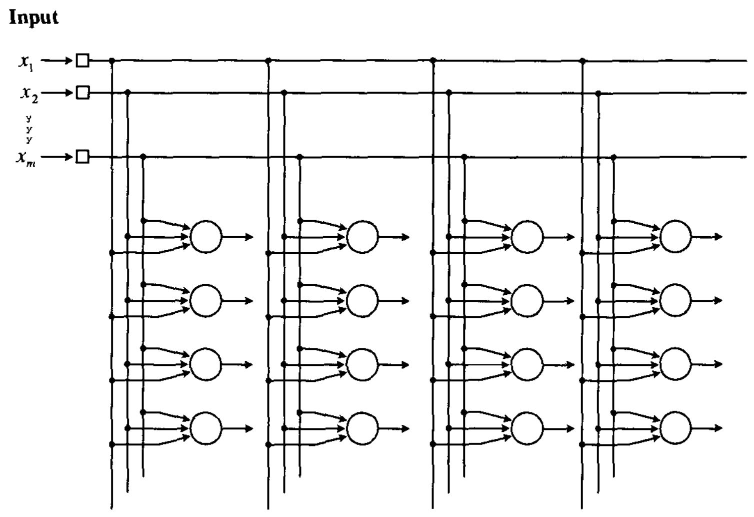

Figure 5.5: Two-dimensional array of neurons

The network maps a feature vector from the input space to a neuron in this two-dimensional array whose reference vector is at a minimal distance from the feature vector.

The formation of the SOFM involves the following three essential processes [37]:

1. Competition

   For every input vector, each neuron calculates the distance between its reference vector and the input vector. The neuron with the minimum distance is declared the winner of the competition.

2. Cooperation

   The winning neuron determines its topological neighboring neurons in the lattice, and provides the basis for cooperation within its neighborhood. Assume the SOFM array contains N units and for an input vector $\mathbf{x}$, the winning neuron index is i. The neighborhood function of the winning neuron i and its neighbor neuron j is defined by

$$h_{j,i} = exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right) \quad i,j = 1,2,\ldots,N, \tag{5.1}$$

where $d_{j,i}$ is the lateral distance between the winning neuron i and its neighbor neuron j and $d_{i,i} = 0$. The standard deviation σ defines the *effective width* of its topological neighborhood. The neighborhood function $h_{j,i}$ has a maximum value 1 at $j = i$, and decreases monotonically with increasing lateral distance $d_{j,i}$.

The SOFM algorithm also requires the size of the neighborhood to shrink with time. This is achieved by making the effective width parameter σ exponentially decay with time t [59, 60]

$$\sigma(t) = \sigma_0 exp\left(-\frac{t}{\tau_1}\right), \tag{5.2}$$

where σ_0 is the initial value of σ, and τ_1 is a time constant.

3. Synaptic adaptation

At this stage, the weights of the winning neuron and all of its neighboring neurons are updated according to the input pattern. The updated weight vector $\mathbf{w}_j(t+1)$ at time $t+1$ is defined by [55, 57, 60]

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t)h_{j,i}(t)(\mathbf{x} - \mathbf{w}_j(t)) \quad j = 1,2,\ldots,N, \tag{5.3}$$

where $\eta(t)$ is a scalar-valued *learning-rate factor*. It starts at an initial value η_0 and decreases exponentially with time t

$$\eta(t) = \eta_0 exp\left(-\frac{t}{\tau_2}\right), \tag{5.4}$$

where τ_2 is another time constant.

The adjustments have the effect of moving the weight vectors of the winning neuron and all of its neighbors toward the input vector $\mathbf{x}$, and enhancing the response of the winning neuron to a subsequent similar input. Moreover, the weight vectors are updated repeatedly in accordance with the repeated presentations of the input vectors in the training data and tend to follow the distribution of the input space due to the neighborhood updating.

SOFM networks training

The training of SOFM networks involves two phases: an ordering phase followed by a convergence phase. These two phases of adaptation are described as below [55, 57]:

1. Order phase

   It is during the first phase of the adaptive process that topological ordering of the weight vectors takes place. The ordering phase may take as many as 1000 iterations of the SOFM algorithm. The neighborhood function $h_{j,i}(t)$ and learning-rate parameter $\eta(t)$ in this phase are set as follows:

   - The neighborhood function $h_{j,i}(t)$ defined by (5.1,5.2) initially includes all neurons in the networks and shrinks slowly with time. The initial size σ_0 is set equal to the "radius" of the lattice. The time constant τ_1 is defined by:

     $$\tau_1 = \frac{1000}{\log \sigma_0}. \qquad (5.5)$$

   - The learning-rate factor $\eta(t)$ is calculated using (5.4), where the two constants are chosen as:

     $$\eta_0 = 0.1 \qquad (5.6)$$

$$\tau_2 = 1000,$$

2. Convergence phase.

   This second phase fine-tunes the feature map and provides an accurate statistical model of the input space. This phase needs many more iterations than the first phase - usually at least 500 times the number of neurons in the networks [58, 37]. The neighborhood and the learning-rate are set as follows:

   - The neighborhood function $h_{j,i}(t)$ in this phase only contains the nearest neighbors of a winning neuron and eventually reduces to zero in neighboring neurons.
   - The learning rate $\eta(t)$ is maintained at a small value of the order of 0.01.

SOFM properties

Once the SOFM algorithm has converged, the mapping from the input feature space to the discrete neuron array displays some important statistical characteristics of the input space, and has some interesting properties [37].

1. Approximation of the input space

   The set of the weight vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$ provide a good approximation of the input space.

2. Topological ordering

   The spatial location of a neuron in the lattice corresponds to a particular domain of features in the input space. The mapping computed by the SOFM algorithm preserves the most important topological and metric relationships of the input features.

3. Density matching

   The map reflects variations of the statistics of the input distribution: regions of the input space from where sample vectors are drawn with a high probability of occurrence are mapped to larger domains of the output space, and therefore with better resolution than regions where sample vectors occur with low probability.

Properties 2 and 3 supply the theoretical basis for designing a fault visualization approach using SOFM.

5.2.3 Feature Encoding and Model Computing

Since the SOFM gives a good approximation of the distribution of training features in the input space using a small set of reference vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$, it offers an approach for feature vector quantization. The set of reference vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}$ becomes the codebook, and its members are called codewords. Each reference vector $\mathbf{w}_n$ can be represented by the index of its associated neuron i_n, here $n = 1, 2, \ldots, N$. The codebook is then used to encode all feature vectors extracted in 5.2.1.

Suppose the historical database training the SOFM networks involves M classes of faults, $\theta_1, \theta_2, \ldots, \theta_M$. After feature vector encoding, all training data are encoded by N neural indices $i_1, i_2, ..., i_N$. The conditional probability function $P_m(i_n|\theta_m)$ of each fault is defined by:

$$\begin{aligned} P_m(i_n|\theta_m) &= \frac{\text{Total number of features of fault } \ \theta_m \text{ encoded as } \ i_n}{\text{Total number of features belonging to fault } \ \theta_m} \\ & \quad n = 1, 2, \ldots, N; \quad m = 1, 2, \ldots, M. \end{aligned} \tag{5.7}$$

Since each neuron is associated with a reference vector in the input feature space, the conditional probability $P_m(i_n|\theta_m)$ represents the distribution information of fault

θ_m in the feature space based on historical data records, and serves as the fault model to identify unknown faults occurring in the process.

5.2.4 Model Improvement

As we know, the SOFM matches the density of the input distributions. Regions where features have high probability of occurrence have high resolution in the output neuron lattice. One problem arises when features extracted from two different faults overlap. The overlapped regions usually have higher density than non-overlapped ones. Instead of using the information of non-overlapping regions for fault classification, the SOFM focuses on their overlapped regions. As a result, the fault models built based on SOFM are prone to have similar distributions, making fault identification difficult.

Fault modelling is enhanced by adding learning vector quantization to the detection system and improves the separation of faults with similar distributions.

Kullback-Leibler divergence

The distance between any two distributions is measured by the relative entropy or Kullback-Leibler divergence.

Given two probability functions P_p and P_q, the Kullback-Leibler divergence is defined by [61]:

$$K(p,q) = \Sigma_{n=1}^{N} P_p(i_n|\theta_p) log\left(\frac{P_p(i_n|\theta_p)}{P_q(i_n|\theta_q)}\right). \tag{5.8}$$

The Kullback-Leibler divergence has some unique properties:

1. It always has a nonnegative value and is zero only when the two distributions match perfectly, $P_p = P_q$.

2. It is invariant with respect to the permutation of the order in which the components $i_1, i_2, ..., i_N$ are arranged.

The Kullback-Leibler divergence has a large value when the two distributions are very different and has a small value when they are similar. Therefore, it can be considered as the distance between two distributions, although it is not a real distance measure because it is not symmetric. A symmetric measure

$$J(p,q) = K(p,q) + K(q,p), \tag{5.9}$$

is used here to measure the distance between two fault distribution models. Whenever two faults are found to have similar distributions, the learning vector quantization (LVQ) technique is applied to modify their fault models.

Learning vector quantization

Learning vector quantization is a supervised learning technique that uses fault class information in the training database to move the positions of reference vectors $\{\mathbf{w}_n\}$ slightly to increase the distance between any two fault distribution models defined by (5.9).

Assume that fault θ_p and θ_q occupy very close or overlapped regions of the feature space and thus have similar distributions. After SOFM vector quantization, the feature vectors of these two faults are encoded to the nearest codewords. The subset of codewords related to these two faults are first assigned to each fault class according to the following criterion:

Let $\zeta_{p,q}$ be the subset of codewords related to fault θ_p or θ_q, then a codeword $\mathbf{w}_n \in \zeta_{p,q}$ is assigned to fault class θ_p if

$$P(i_n|\theta_p) \geq P(i_n|\theta_q),$$

and vice versa.

Given an input feature vector $\mathbf{x}$ of fault θ_p or θ_q, and its nearest codeword $\mathbf{w}_i$, the learning vector quantization process is defined by the following equations:

$$\begin{aligned}
\mathbf{w}_i(t+1) &= \mathbf{w}_i(t) + \eta(t)(\mathbf{x} - \mathbf{w}_i(t)) \qquad (5.10)\\
&\quad \text{if } \mathbf{x} \text{ and } \mathbf{w}_i \text{ belong to the same fault classes,}\\
\mathbf{w}_i(t+1) &= \mathbf{w}_i(t) - \eta(t)(\mathbf{x} - \mathbf{w}_i(t))\\
&\quad \text{if } \mathbf{x} \text{ and } \mathbf{w}_i \text{ belong to different fault classes,}\\
\mathbf{w}_j(t+1) &= \mathbf{w}_j(t) \qquad \text{for } j \neq i.
\end{aligned}$$

If the class label of the input feature vector $\mathbf{x}$ and its nearest codeword $\mathbf{w}_i$ agree, then the codeword is moved in the direction of the feature vector $\mathbf{x}$. If, on the other hand, the class labels of the feature vector $\mathbf{x}$ and its nearest codeword $\mathbf{w}_i$ disagree, the codeword is moved away from the feature vector $\mathbf{x}$. Other codewords remain unchanged.

After codewords are updated according to (5.10), all fault distribution models are recalculated using (5.7). In general, by using the learning vector quantization technique in conjunction with the SOFM, similar distributions can be further separated and improved fault models are obtained.

5.3 On-line Fault Identification

The previous section introduced the learning stage of the fault identification system. It is in the learning stage that distribution models of all possible faults of an industrial process are built using prior fault histories. This section will cover another operating stage, the on-line working stage. In this stage, the fault models built in the learning stage are applied to diagnose on-line fault conditions of the process.

Assume that the industrial process involves M classes of possible faults $\theta_1, \theta_2, \ldots, \theta_M$. In the learning stage, the M fault distribution models were built. Also a codebook with N codewords labelled as $i_1, i_2, \ldots, i_N$ was obtained through the use of the SOFM network. During the on-line working stage, once an unknown fault condition is detected, the subsequent L observations $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L$ are collected for the purpose of fault identification. On-line fault identification involves the three steps listed below.

Step 1: Feature extraction

The way process features are extracted in the on-line working stage is similar to that in the learning stage. The same length of data as in the learning stage are collected after a fault is detected and their nonlinear principal component features are extracted using the NLPCA networks.

Step 2: Feature encoding

The process feature vectors obtained in Step 1 are encoded using the codebook obtained in the learning stage.

Step 3: Fault identification

The final step is to identify what kind of fault is involved in the process based on the fault distribution models built in the learning stage.

After steps 1 and 2, the L process observations $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L$ are encoded as a sequence of codeword indices $x_1, x_2, \cdots, x_L$. The fault identification problem can be described as: given a sequence of codeword indices $x_1, x_2, \cdots, x_L$, decide which one of the M fault classes they belong to.

Define an L-dimensional vector $X = \{x_1, x_2, \cdots, x_L\}$. According to the Bayes decision theory, X is assigned to class θ_{max} if it maximizes the probabilities $P(\theta_m|X)$,

$m = 1, 2, ..., M$.

$$\theta_{max} = \arg_{\theta_m} \max_{m=1}^{M} P(\theta_m|X) \tag{5.11}$$

From Bayes rule we have:

$$P(\theta_m|X) = \frac{P(X|\theta_m)P(\theta_m)}{P(X)}. \tag{5.12}$$

Assume the *a priori* probabilities of the M fault classes are equal, that is $P(\theta_1) = P(\theta_2) = \cdots = P(\theta_M)$. Then equation (5.11) becomes:

$$\theta_{max} = \arg_{\theta_m} \max_{m=1}^{M} P(X|\theta_m). \tag{5.13}$$

This is also known as the maximum likelihood estimation method. Assuming statistical independence between the L observations, we have:

$$P(X|\theta_m) = \prod_{l=1}^{L} P(x_l|\theta_m). \tag{5.14}$$

Equation (5.13) can then be written as:

$$\theta_{max} = \arg_{\theta_m} \max_{m=1}^{M} \prod_{l=1}^{L} P(x_l|\theta_m). \tag{5.15}$$

Because of the monotonicity of the logarithmic function, Equation (5.15) is equivalent to:

$$\theta_{max} = \arg_{\theta_m} \max_{m=1}^{M} \sum_{l=1}^{L} \ln P(x_l|\theta_m), \tag{5.16}$$

where $x_l \in \{i_1, i_2, \ldots, i_N\}$, $l = 1, 2, \ldots, L$.

Since the distribution models of the M faults $P(i_n|\theta_m), n = 1, 2, \ldots, N$, $m = 1, 2, \ldots, M$ have been built in the learning stage, the right side of equation 5.16 can be calculated and compared, and thus the most likely fault in the system can be identified.

5.4 Conclusions

A statistical fault identification system integrating NLPCA neural networks with SOFM neural network techniques has been developed in this work. Operations under various known types of fault conditions in the historical database are first modelled through the learning of their distribution patterns in the nonlinear principal component space. These models are then applied to identify on-line unknown faults based on their similarity to previous fault behaviors. The system enables the automation of NLPCA-based fault identification where the visual interpretation of nonlinear principal component patterns is replaced by computer-aided interpretation. Unlike the analytical model-based methods which need accurate mathematical models of the processes, the system presented here gives an alternate approach for diagnosing faults in multivariate processes when sufficient historical databases are available.

Chapter 6

Three-Tank Benchmark Test

6.1 The Three-Tank Benchmark System

Leaks, clogs, valve blockages and sensor faults are examples of some common faults in chemical processes and process control applications. To study the corresponding fault diagnostic problems, a laboratory desktop plant, composed of three tanks interconnected by various hydrodynamic paths, was built by the Department of Computer Automation and Control at Jozef Stefan Institute. The plant mimics some common processes for transporting fluids in chemical plants [62].

The schematic of the process is shown in Figure 6.1 [62]. It consists of three tanks R1, R2, and R3 connected with flow paths which serve to supply water from the reservoir R0. Two of the paths have built-in pumps, P1 and P2, driven by DC motors with permanent magnets. There is one servo valve V5 in the plant, driven by a DC motor. Valves V1 and V2 are on-off while V3 and V4 are manual valves. The purpose of valve V3 is primarily for realizing a leakage fault of the tank R1.

The capacity of the reservoir R0 is much greater than the capacity of tanks so that its level is practically constant during the operation.

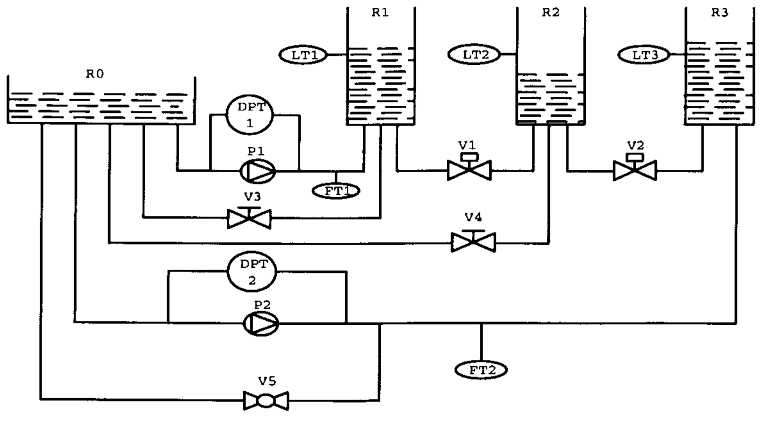

Figure 6.1: Schematic of the three-tank process

The three-tank system can be studied in different configurations and operating modes. In this study, R1 and R3 take over the role of buffers for supplying R2. Contents from R1 and R3 are mixed in R2 and then fed back to reservoir R0. The level in tank R1 is controlled by manipulating the speed of pump P1, while the level in R3 is adjusted by controlling the voltage command signal of valve V5 around a given working point defined by the constant speed of pump P2. Proper ratios of flows from R1 into R2 and from R3 into R2 are achieved by adjusting the difference of the reference values of levels in the tanks.

A nonlinear simulation model for the entire operating range of the system is built in Simulink. The model can simulate 20 different faults in sensors, actuators and components. Table 6.1 lists the 20 faults realized in the simulation model. The intensity of each fault is in the range of 0 to 1.

The system can be operated in both open loop and closed loop. In this study, the closed-loop model is used for simulation of the plant under influence of faults. Its Simulink scheme is shown in figure 6.2 [62]. There are two feedback loops in the system. The first loop controls the level of tank R1 by adjusting the speed w_1 of pump

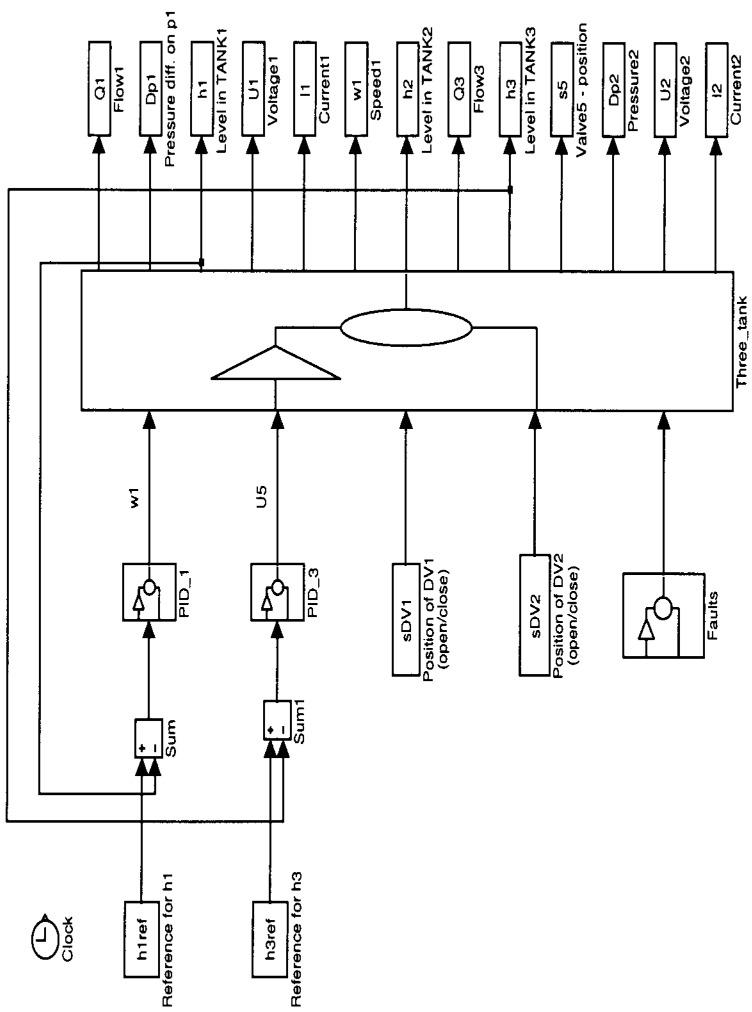

Figure 6.2: Simulink scheme of the three-tank system

Table 6.1: List of process faults

Index	Fault
1	leak in R1
2	clog in branch with V1
3	clog in branch with V2
4	clog in branch with V4
5	clog in branch with P1
6	clog in branch with P2
7	increased friction in DC motor of P1
8	clog in branch with V5
9	bias in sensor of h_1
10	bias in sensor of h_2
11	bias in sensor of h_3
12	bias in sensor of Q_1
13	bias in sensor of Δ p_1
14	bias in sensor of I_1
15	bias in sensor of U_1
16	bias in sensor of Q_3
17	bias in sensor of Δ p_2
18	bias in sensor of I_2
19	bias in sensor of U_2
20	bias in sensor of ω_1

P1. The second loop controls the level of tank R3 by adjusting the position of valve V5. The overall model consists of two control inputs and eleven outputs. Table 6.2 lists the variables of the plant and their nominal ranges. Zero-mean Gaussian noise with known standard deviations is added to the measured values.

6.2 Monitoring the Three-Tank System

This section studies the performance of the NLPCA neural network approach for monitoring the three-tank system. The Simulink model of the plant was used to

Table 6.2: Nominal value of measured process variables

Symbol	Variable	Unit	Range	Description
LT1	h_1	cm	0 - 60	level in tank R1
LT2	h_2	cm	0 - 60	level in tank R2
LT3	h_3	cm	0 - 60	level in tank R3
DPT1	Δ p_1	cm H_2O	0 - 100	pressure difference in the pump1
DPT2	Δ p_2	cm H_2O	0 - 100	pressure difference in the pump2
FT1	Q_1	cm^3/s	0 - 30	flow through the branch with pump1
FT2	Q_2	cm^3/s	0 - 30	flow through the branch with pump2
-	U_1	V	0 - 10	voltage on the DC motor in pump1
-	I_1	-	0 - 0.3	current to the DC motor in pump1
-	U_2	V	0 - 10	voltage on the DC motor in pump2
-	I_2	-	0 - 0.3	current to the DC motor in pump2
-	w_1	-	0 - 6	"Speed" of rotation of pump1
-	s_5	-	0 - 10	position of the stem of the continuous valve V5

simulate the operation of the system. The reference level h_{1ref} of tank R1 and the reference level h_{3ref} of tank R3 were kept constant in the study. Measurements of process variables listed in table 6.2 were collected for FDI purpose.

The NLPCA neural network model of the system was obtained with a set of measurements collected for 3000 seconds from the process under routine operation with no system fault. (In real processes, much more data would be required to capture all sources of common-cause variations.) The NLPCA neural network has 13 inputs; its four-layer structure is N1=N3=N4=13 and N2=3.

The simulation study shows how well the NLPCA neural network works for detecting single faults occurring under normal operating conditions. The plant was initially operating in a normal and well-behaved manner. A process fault with intensity of 0.1 was introduced into the plant at 210 seconds and lasted for 100 seconds. At 510 seconds, the same fault, with intensity of 0.5, was introduced to the plant and lasted for 100 seconds. The same method was repeated for all faults listed in table 6.1.

Figure 6.3 - 6.7 plot the residuals of the NLPCA-based approach in response to 20 different faults with intensities of 0.1 and 0.5.

Test results show that the NLPCA neural network approach effectively detects component faults, such as a leak in tank R1 (see figure 6.3 (1)) or a clog in the main branches (see figure 6.3 (2,3,4), 6.4 (5,6)). Biases in the sensors (see figure 6.5, 6.6, 6.7) were detected including biases in the feed-back loops (see figure 6.5 (9,11)). The results show that sensitivities of the residuals to different faults vary from case to case. The NLPCA residuals are not sensitive to the clog in branch with P1 and increased friction in DC motor of P1 (see figure 6.4 (5,7)). Further study of these two cases shows that these two faults are located in the actuator of the system and their nonlinear principal component features lie very close to that of the normal region.

6.3 SOFM-Based Fault Identification

Monitoring the residuals from the NLPCA neural network determines the overall status of the three-tank system; however, the individual process variables responsible for the occurrence of the fault are not determined by residual charts. In this section, simulations are made to test the SOFM-based fault identification system introduced in chapter 5 in performing single-fault identification.

6.3.1 Fault Distribution Modelling

Training database preparation

Building distribution models of process faults requires a fault training database. A fault database for the three-tank system was generated from the Simulink model and consists of fault conditions caused by the 20 faults listed in table 6.1 at two severity

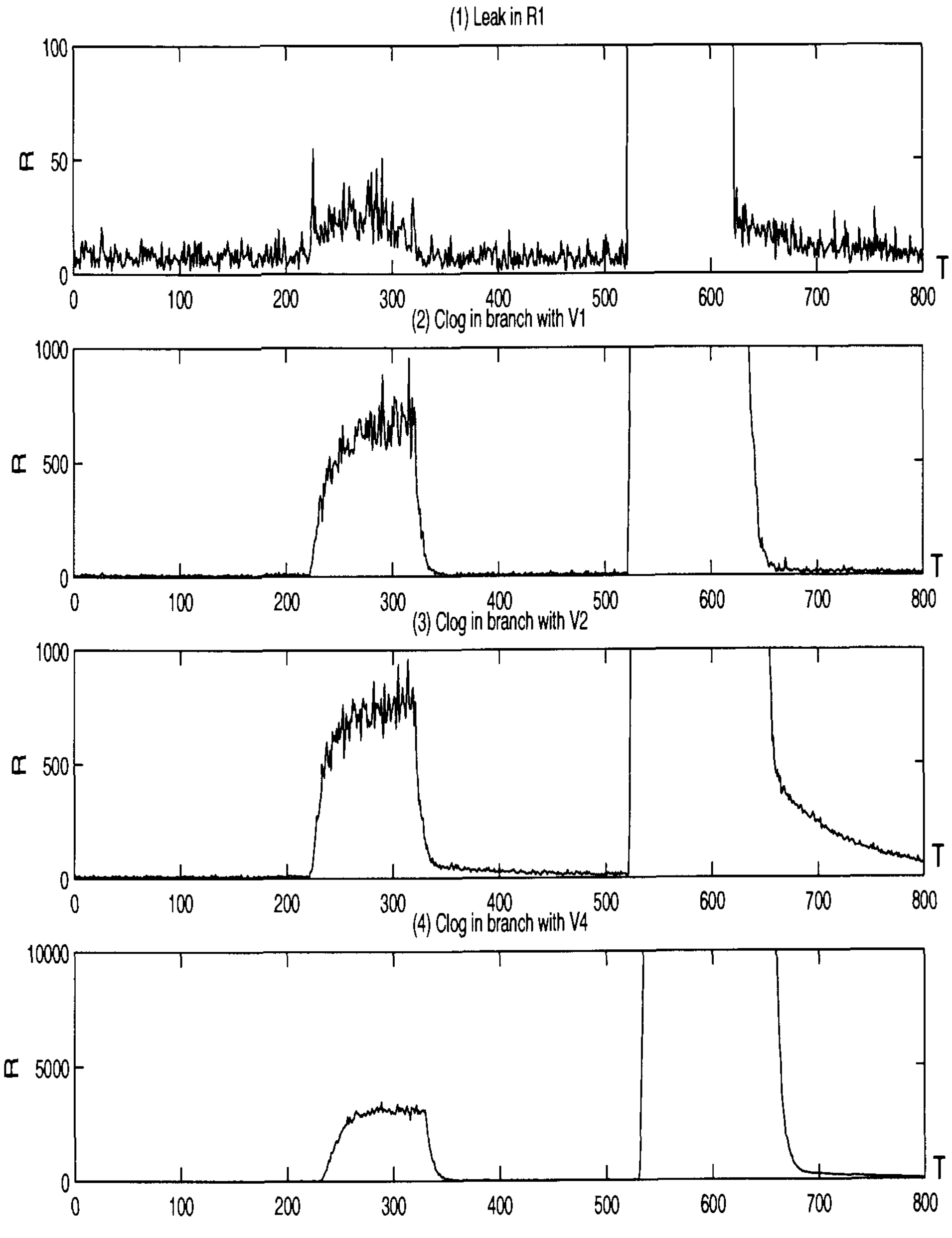

Figure 6.3: Residual plots for a single fault with intensity of 0.1 and 0.5

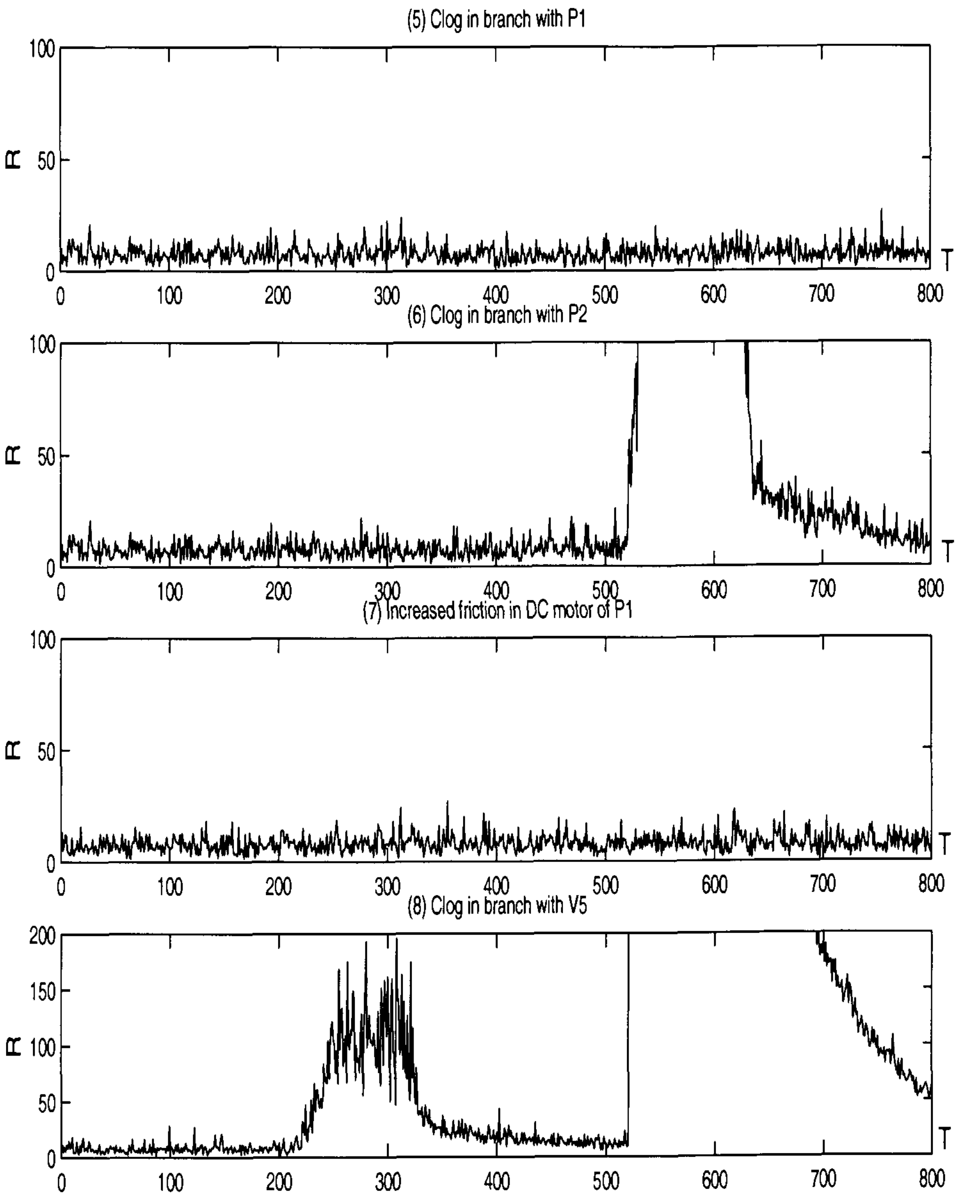

Figure 6.4: Residual plots for a single fault with intensity of 0.1 and 0.5

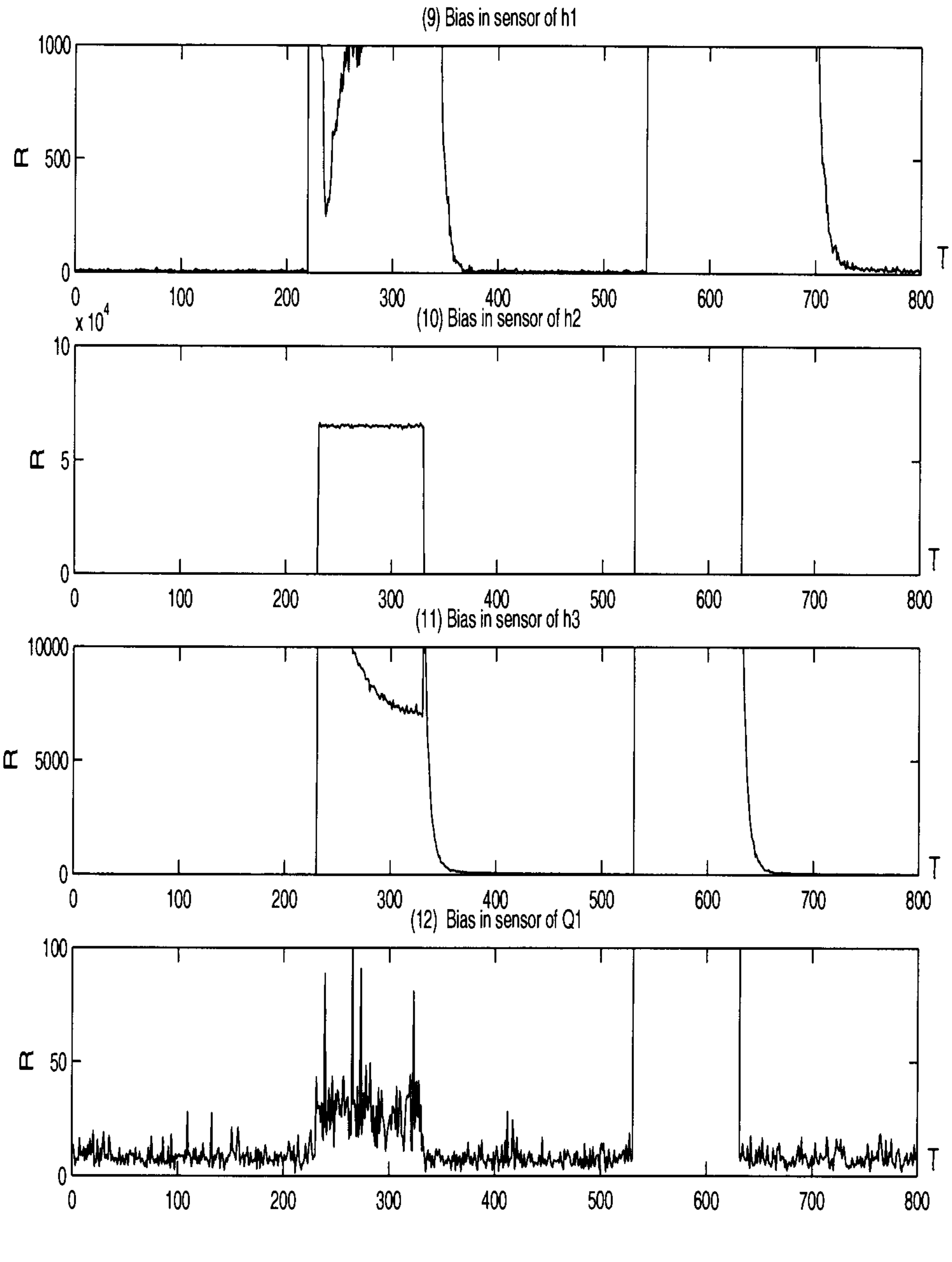

Figure 6.5: Residual plots for a single fault with intensity of 0.1 and 0.5

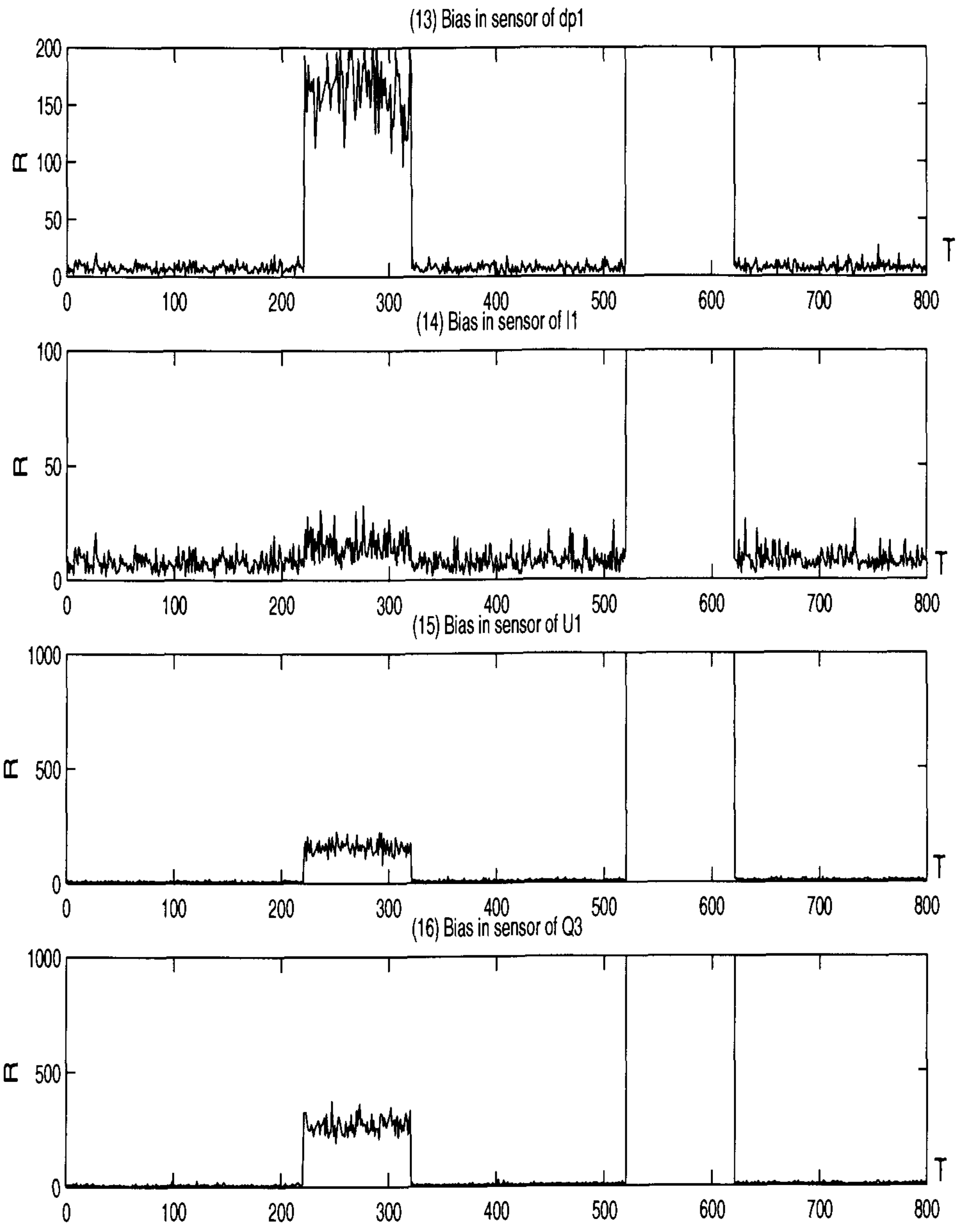

Figure 6.6: Residual plots for a single fault with intensity of 0.1 and 0.5

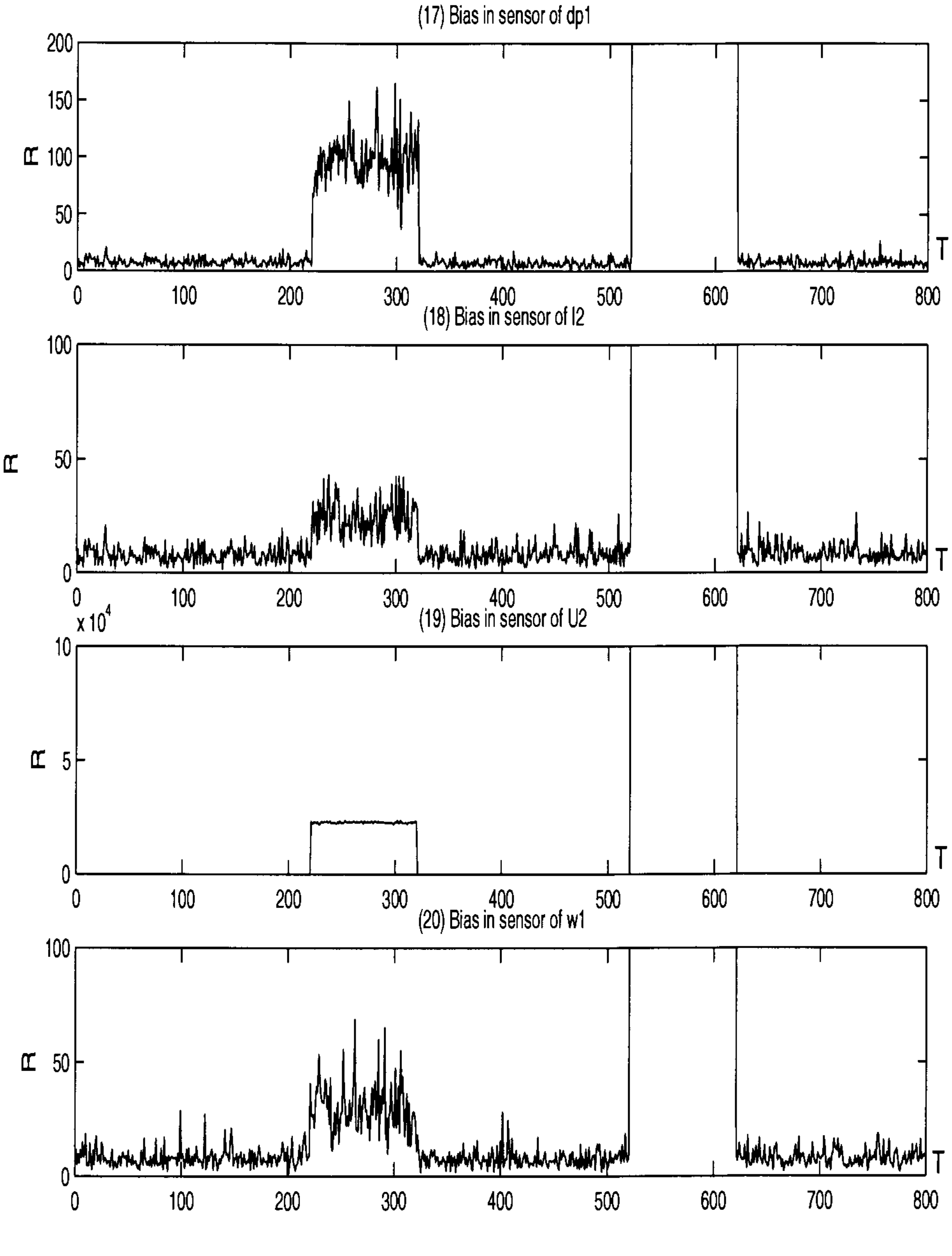

Figure 6.7: Residual plots for a single fault with intensity of 0.1 and 0.5

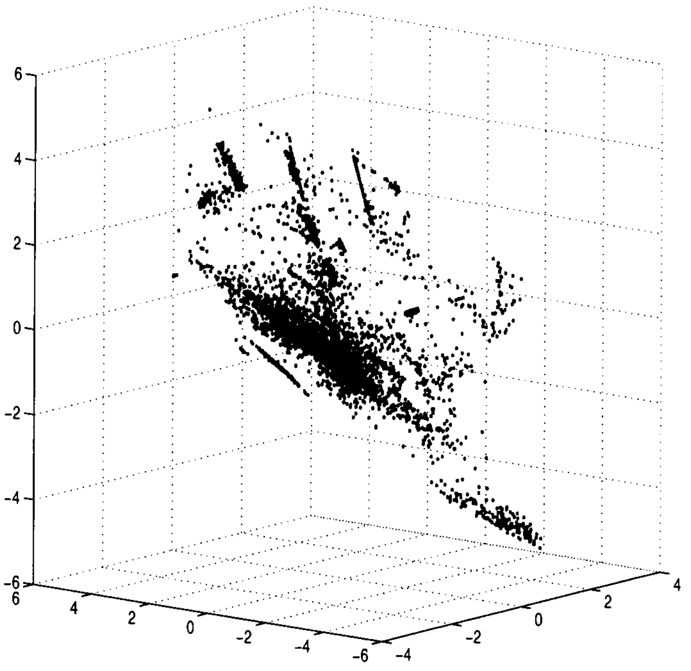

Figure 6.8: Nonlinear principal component plots of the training database

levels, 0.1 and 0.5.

Feature extraction

The training database was preprocessed by the NLPCA network model built in section 6.2. The network has three neurons in its nonlinear principal component layer (second layer). For a given input process observation, the outputs of the second layer are collected and form a nonlinear principal component feature vector of the observation. Figure 6.8 plots the principal component features of the training database in a 3-dimensional feature space.

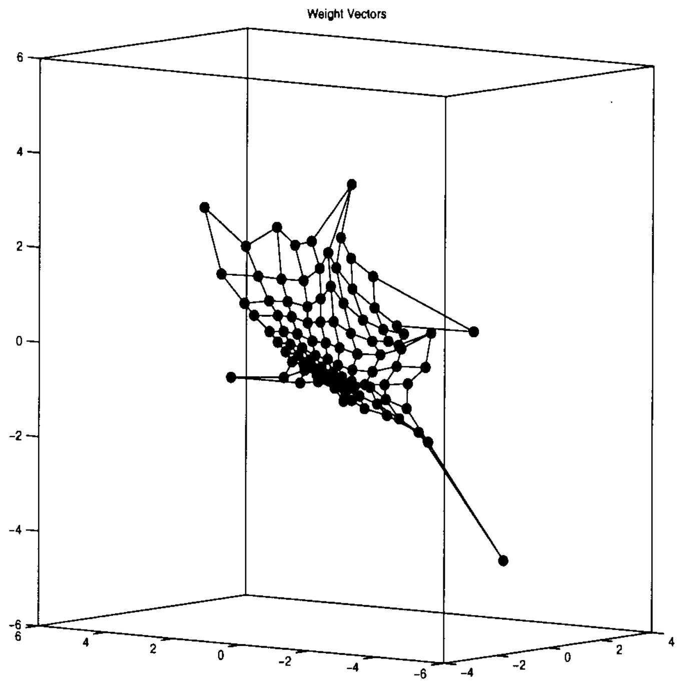

Figure 6.9: Reference vector structure of SOFM

Self-Organizing Feature Mapping

SOFM networks with a 10 $\times$ 10 lattice were applied to learn the distribution of the principal component features of the training database. After training, the set of reference vectors of the SOFM networks provided a good approximation of the training feature space. Figure 6.9 plots the structure of the reference vectors associated with the two-dimensional array of neurons after the learning process.

The LVQ algorithm adjusts the position of the reference vectors to increase the distance between any two similar fault distributions. These reference vectors act as codebook vectors to quantize all features of the training database. Fault distribution models were built using equation 5.7.

6.3.2 On-line Fault Identification

After building fault distribution models for the three-tank system, the models were tested for fault sensing with unknown fault records. An unknown fault was simulated and the process response to the fault was measured and used as the input to the NLPCA-SOFM-based fault identification system. Table 6.3 - 6.8 compares the performance of the NLPCA-SOFM-based fault identification system with the actual system fault.

Extensive tests show that the SOFM-based fault identification system performs accurate identification of single faults in the three-tank system. Faults were identified correctly 80% of the time. Fault identification was the most accurate when the level of the introduced fault is close to the level of the fault recorded in the training database. The system fails to identify faults when the fault intensities are lower than 0.05.

An interesting result is that the SOFM-based fault identification system identifies actuator faults (F5, F7) which are not detectable by the NLPCA neural network. Therefore, the system can be used for fault detection as well as identification. The disadvantage of the SOFM-based method is that data buffering is required while the NLPCA networks can perform real-time monitoring.

6.3.3 Fault Visualization

SOFM-based methods have also been applied as a visualization tool for mapping high-dimensional process data to a two-dimensional uniform grid. For the three-tank system model, a grid with 10×10 internally ordered units is constructed to visualize the topology of the input space. Data association in the input space is visualized in terms of the relative positions of the units. The gray scale of each unit is proportional to the quantity of process data in the region represented by the unit. Light areas correspond to large quantity of process data, while dark areas correspond to small quantity of process data.

Figures 6.10 and 6.11 show the patterns of the 20 fault conditions with intensity of 0.1 and 0.5. The pattern of normal conditions is shown on the left top of these two figures. These patterns prove the assumption in chapter 5 that different faults in a process may cause the process to drift away from its normal state and move along different trajectories and different faults have different distribution patterns. For simple systems, fault identification can be achieved by comparing the pattern of an unknown fault with those of fault banks based on human vision. However, for complex systems, such as the three-tank system, involving large amount of faults, human vision-based approach becomes difficult. In such cases, the NLPCA-SOFM-based fault identification system is more effective.

6.4 Conclusions

The implementation of the NLPCA-SOFM-based fault diagnosis approach was tested on the three-tank system. The results show that in most cases, the NLPCA neural network performs early and accurate detection of an abnormality in the system when the abnormality is due to a root cause in the system components or sensors. The results demonstrate the capability of the SOFM-based fault identification system to correctly identify single faults. An advantage of this approach is the automation of the SOFM-based fault identification where manual interpretation of the nonlinear principal components is replaced by automated interpretation based on SOFM. This approach also enables fault visualization in a two dimensional lattice, which can provide the operator with visible knowledge of the system bias.

Table 6.3: Testing results for faults at intensity of 0.05

Fault introduced		**Fault diagnosed**	**Correct**
Leak in R1	0.05	Clog in branch with P1	No
Clog in branch with V1	0.05	Clog in branch with V5	No
Clog in branch with V2	0.05	Clog in branch with V2	Yes
Clog in branch with V4	0.05	Bias in sensor of w1	No
Clog in branch with P1	0.05	Clog in branch with P1	Yes
Clog in branch with P2	0.05	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.05	Increased friction in DC motor of P1	Yes
Clog in branch with V5	0.05	Clog in branch with V5	Yes
Bias in sensor of h1	0.05	Bias in sensor of h1	Yes
Bias in sensor of h2	0.05	Bias in sensor of h2	Yes
Bias in sensor of h3	0.05	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.05	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.05	Bias in sensor of h2	No
Bias in sensor of I1	0.05	Clog in branch with V5	No
Bias in sensor of U1	0.05	Bias in sensor of w1	No
Bias in sensor of Q3	0.05	Increased friction in DC motor of P1	No
Bias in sensor of dp2	0.05	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.05	Bias in sensor of I1	No
Bias in sensor of U2	0.05	Bias in sensor of U2	Yes
Bias in sensor of w1	0.05	Bias in sensor of dp2	No
Correct Rate			**55%**

Table 6.4: Testing results for faults at intensity of 0.1

Fault introduced		**Fault diagnosed**	**Correct**
Leak in R1	0.10	Leak in R1	Yes
Clog in branch with V1	0.10	Clog in branch with V1	Yes
Clog in branch with V2	0.10	Clog in branch with V2	Yes
Clog in branch with V4	0.10	Clog in branch with V4	Yes
Clog in branch with P1	0.10	Clog in branch with P1	Yes
Clog in branch with P2	0.10	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.10	Increased friction in DC motor of P1	Yes
Clog in branch with V5	0.10	Clog in branch with V5	Yes
Bias in sensor of h1	0.10	Bias in sensor of h1	Yes
Bias in sensor of h2	0.10	Bias in sensor of h2	Yes
Bias in sensor of h3	0.10	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.10	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.10	Bias in sensor of dp1	Yes
Bias in sensor of I1	0.10	Bias in sensor of I1	Yes
Bias in sensor of U1	0.10	Bias in sensor of U1	Yes
Bias in sensor of Q3	0.10	Bias in sensor of Q3	Yes
Bias in sensor of dp2	0.10	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.10	Bias in sensor of I2	Yes
Bias in sensor of U2	0.10	Bias in sensor of U2	Yes
Bias in sensor of w1	0.10	Bias in sensor of w1	Yes
Correct Rate			**100%**

Table 6.5: Testing results for faults at intensity of 0.2

Fault introduced		**Fault diagnosed**	**Correct**
Leak in R1	0.20	Leak in R1	Yes
Clog in branch with V1	0.20	Clog in branch with V1	Yes
Clog in branch with V2	0.20	Clog in branch with V2	Yes
Clog in branch with V4	0.20	Clog in branch with V4	Yes
Clog in branch with P1	0.20	Clog in branch with P1	Yes
Clog in branch with P2	0.20	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.20	Increased friction in DC motor of P1	Yes
Clog in branch with V5	0.20	Bias in sensor of w1	No
Bias in sensor of h1	0.20	Bias in sensor of h1	Yes
Bias in sensor of h2	0.20	Bias in sensor of h2	Yes
Bias in sensor of h3	0.20	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.20	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.20	Bias in sensor of dp1	Yes
Bias in sensor of I1	0.20	Bias in sensor of I1	Yes
Bias in sensor of U1	0.20	Bias in sensor of w1	No
Bias in sensor of Q3	0.20	Bias in sensor of I2	No
Bias in sensor of dp2	0.20	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.20	Clog in branch with P2	No
Bias in sensor of U2	0.20	Bias in sensor of U2	Yes
Bias in sensor of w1	0.20	Bias in sensor of w1	Yes
Correct Rate			**80%**

Table 6.6: Testing results for faults at intensity of 0.3

Fault introduced		**Fault diagnosed**	**Correct**
Leak in R1	0.30	Leak in R1	Yes
Clog in branch with V1	0.30	Clog in branch with V1	Yes
Clog in branch with V2	0.30	Bias in sensor of I1	No
Clog in branch with V4	0.30	Clog in branch with V4	Yes
Clog in branch with P1	0.30	Clog in branch with P1	Yes
Clog in branch with P2	0.30	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.30	Increased friction in DC motor of P1	Yes
Clog in branch with V5	0.30	Clog in branch with V5	Yes
Bias in sensor of h1	0.30	Bias in sensor of h1	Yes
Bias in sensor of h2	0.30	Bias in sensor of h2	Yes
Bias in sensor of h3	0.30	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.30	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.30	Bias in sensor of dp1	Yes
Bias in sensor of I1	0.30	Bias in sensor of I1	Yes
Bias in sensor of U1	0.30	Bias in sensor of U1	Yes
Bias in sensor of Q3	0.30	Clog in branch with V5	No
Bias in sensor of dp2	0.30	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.30	Bias in sensor of I2	Yes
Bias in sensor of U2	0.30	Bias in sensor of U2	Yes
Bias in sensor of w1	0.30	Bias in sensor of w1	Yes
Correct Rate			**90%**

Table 6.7: Testing results for faults at intensity of 0.5

Fault introduced		Fault diagnosed	Correct
Leak in R1	0.50	Leak in R1	Yes
Clog in branch with V1	0.50	Clog in branch with V1	Yes
Clog in branch with V2	0.50	Clog in branch with V2	Yes
Clog in branch with V4	0.50	Clog in branch with V4	Yes
Clog in branch with P1	0.50	Clog in branch with P1	Yes
Clog in branch with P2	0.50	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.50	Increased friction in DC motor of P1	Yes
Clog in branch with V5	0.50	Clog in branch with V5	Yes
Bias in sensor of h1	0.50	Bias in sensor of h1	Yes
Bias in sensor of h2	0.50	Bias in sensor of h2	Yes
Bias in sensor of h3	0.50	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.50	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.50	Bias in sensor of dp1	Yes
Bias in sensor of I1	0.50	Bias in sensor of I1	Yes
Bias in sensor of U1	0.50	Bias in sensor of U1	Yes
Bias in sensor of Q3	0.50	Bias in sensor of Q3	Yes
Bias in sensor of dp2	0.50	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.50	Bias in sensor of I2	Yes
Bias in sensor of U2	0.50	Bias in sensor of U2	Yes
Bias in sensor of w1	0.50	Bias in sensor of w1	Yes
Correct Rate			**100%**

Table 6.8: Testing results for faults at intensity of 0.6

Fault introduced		**Fault diagnosed**	**Correct**
Leak in R1	0.60	Leak in R1	Yes
Clog in branch with V1	0.60	Clog in branch with V1	Yes
Clog in branch with V2	0.60	Bias in sensor of I1	No
Clog in branch with V4	0.60	Clog in branch with V4	Yes
Clog in branch with P1	0.60	Clog in branch with P1	Yes
Clog in branch with P2	0.60	Clog in branch with P2	Yes
Increased friction in DC motor of P1	0.60	Bias in sensor of w1	No
Clog in branch with V5	0.60	Clog in branch with V5	Yes
Bias in sensor of h1	0.60	Bias in sensor of h1	Yes
Bias in sensor of h2	0.60	Bias in sensor of h2	Yes
Bias in sensor of h3	0.60	Bias in sensor of h3	Yes
Bias in sensor of Q1	0.60	Bias in sensor of Q1	Yes
Bias in sensor of dp1	0.60	Bias in sensor of dp1	Yes
Bias in sensor of I1	0.60	Bias in sensor of I1	Yes
Bias in sensor of U1	0.60	Bias in sensor of U1	Yes
Bias in sensor of Q3	0.60	Clog in branch with V2	No
Bias in sensor of dp2	0.60	Bias in sensor of dp2	Yes
Bias in sensor of I2	0.60	Bias in sensor of I2	Yes
Bias in sensor of U2	0.60	Bias in sensor of U2	Yes
Bias in sensor of w1	0.60	Bias in sensor of w1	Yes
Correct Rate			**90%**

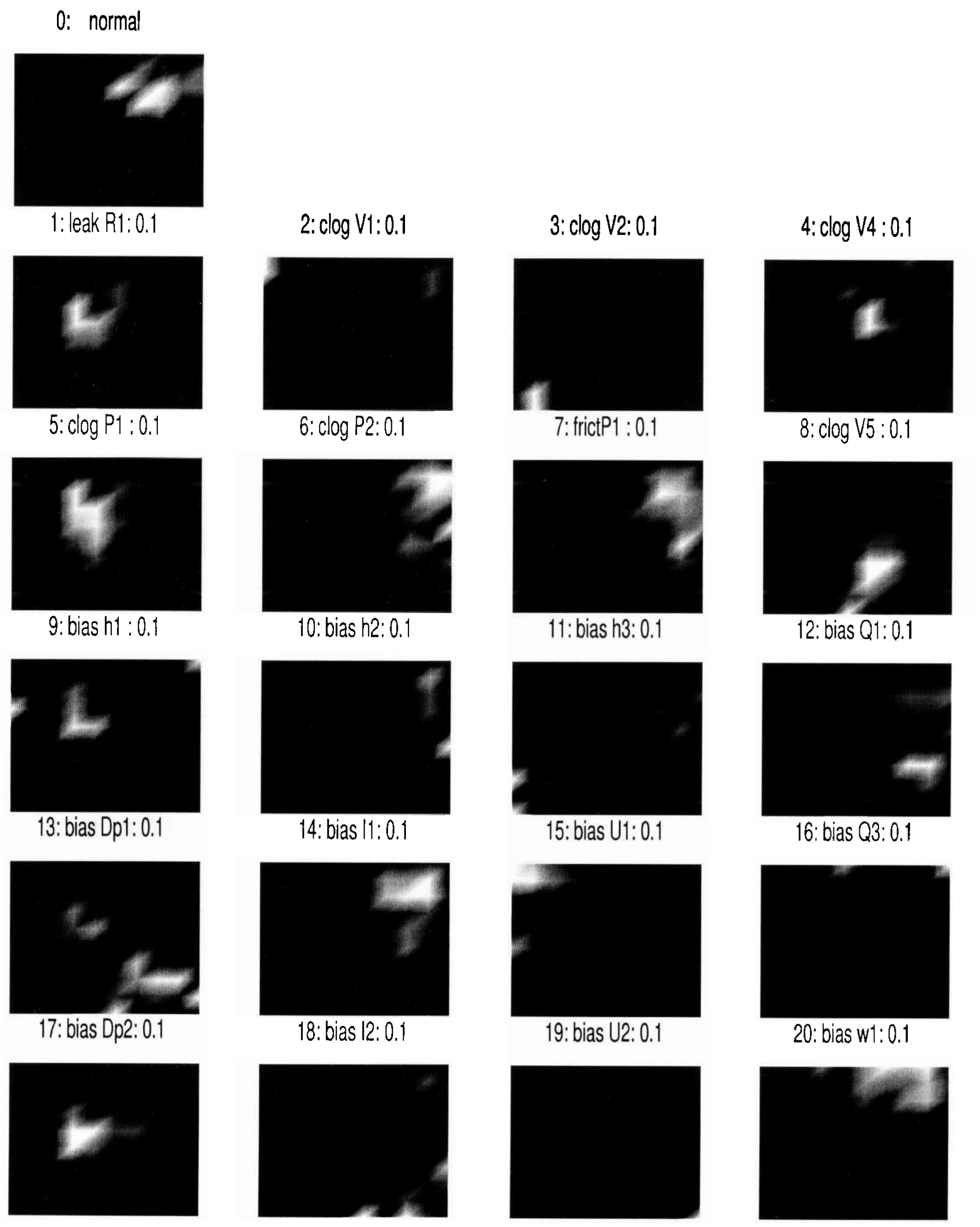

Figure 6.10: Fault representations in the SOFM lattice

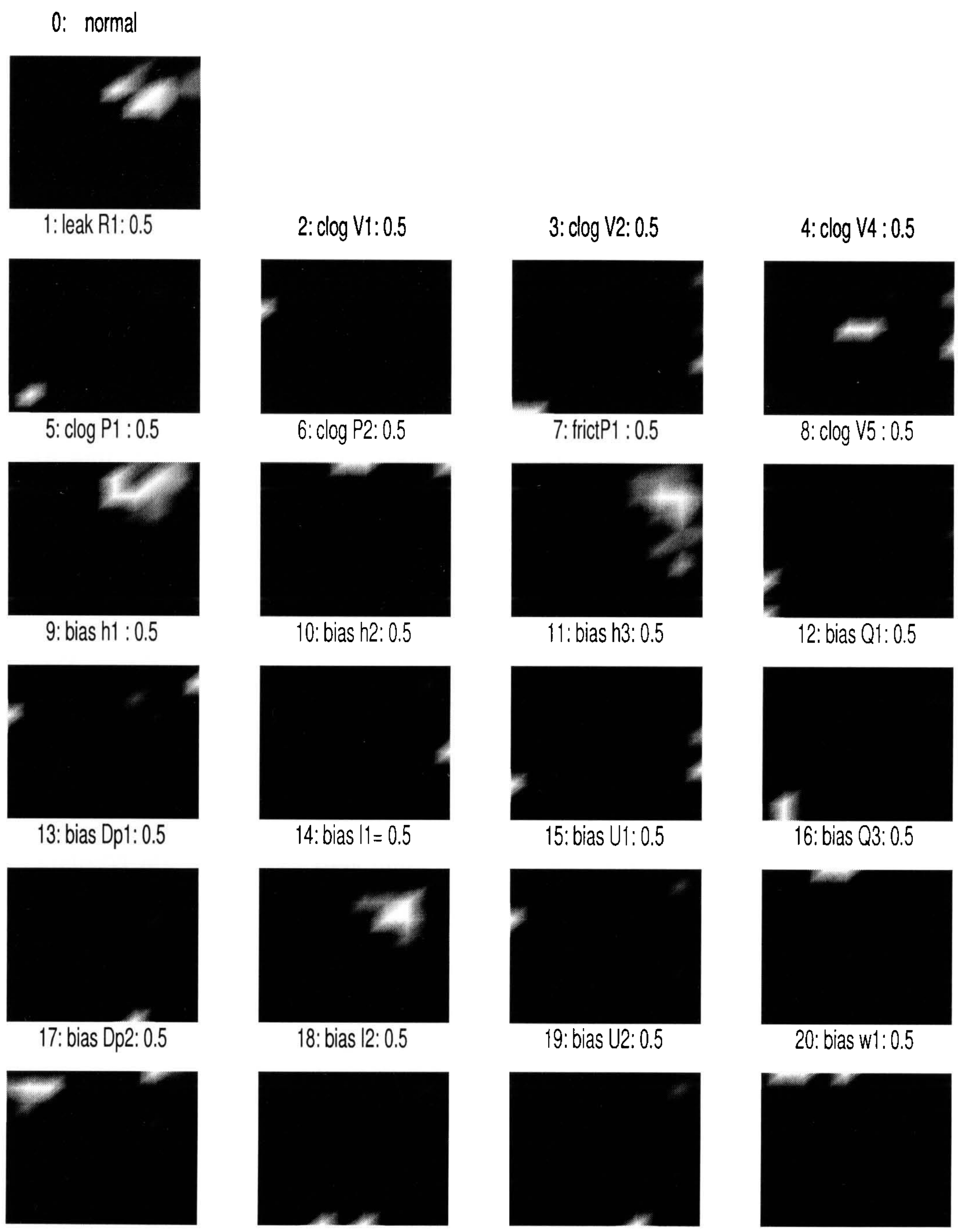

Figure 6.11: Fault representations in the SOFM lattice

Chapter 7

Conclusions

An automated framework for statistical multivariate process monitoring and fault diagnosis based on nonlinear principal component analysis (NLPCA) has been developed. Unlike most analytical model-based methods which need accurate mathematical models of the system, multivariate statistical methods are much easier to develop because of the availability of routine operating data. Also multivariate methods can handle a large number of measured variables. The approach addresses the linearity limitation of most statistical methods by assuming the hidden features are nonlinear functions of the measured observations. The NLPCA neural network and its dynamic extension are techniques that make this approach computationally attractive. The approach also enables the automation of NLPCA-based fault identification where the manual interpretation of the nonlinear principal component patterns is replaced by automated interpretation based on SOFM. These advantages were illustrated with an industrial actuator benchmark as well as an three-tank benchmark system. The concepts and techniques used in this work can also be extended to design large data-based intelligence systems.

Although the NLPCA-SOFM-based fault diagnosis system has many advantages over other methods, there are some limitations. One of the key assumptions made in building the NLPCA model is that the data from the normal operating region of the process have a Gaussian distribution. This condition is often not true in practical industrial processes where the data follow non-Gaussian distribution. In such situations, the NLPCA model could lead to over-generalization, and as a result, data pertaining to some faults could be declared as normal. One way to overcome this problem is to use fuzzy clustering methods [63] to identify the number of Gaussians and their centers. NLPCA is then applied on each of the Gaussians. Another limitation of this system is that the fault identification method based on SOFM could fail to identify different faults that change the measured variables in the same direction. This is due to the qualitative nature of the SOFM-based diagnosis and may be overcome by using knowledge of the process.

Future improvements to the NLPCA-SOFM-based fault diagnosis system could be made in several directions. One direction is to integrate the qualitative knowledge of the process in the system, since the qualitative knowledge of the process could be obtained from various channels in most situations. They could be applied in a proper way to improve the system's diagnosing ability. For example, the cause-effect relation of the process could be useful in choosing measured variables for the NLPCA model and could also be useful in fault diagnosis. Another direction is to incorporate more temporal information in the fault diagnosis system. Sequence detection techniques such as Markov models and dynamic modelling techniques such as recurrent neural networks may improve the fault detection ability of the system. Other improvements could also be made by using fuzzy logic for multiple fault diagnosis.

Bibliography

[1] S. Yoon and J. F. MacGregor, "Statistical and Causal Model-Based Approaches to Fault Detection and Isolation," *AIChE Journal*, Vol. 46, No. 9, pp. 1813-1824, 2000.

[2] P. M. Frank, "Analytical and qualitative model-based fault diagnosis - a survey and some new results," *European Journal of Control*, vol. 2, pp. 6-28, 1996.

[3] E. Y. Chow and A. S. Willsky, "Analytical redundancy and the design of robust failture detection systems," *IEEE Trans. Automatic Control*, vol. 29, pp. 603-619, 1984.

[4] J. Gertler and D. Singer, "A new structural framework for parity equation based failure detection and isolation," *Automatica*, vol. 26, pp. 381-388, 1990.

[5] R. J. Patton, P. M. Frank, R. N. Clark (eds), "Fault diagnosis in dynamic systems, theory and application," Prentice Hall, Englewood Cliffs, NJ, 1989.

[6] P. M. Frank, "Advances in observer-based fault diagnosis," *Proc. of the Conference TOOLIDIAG 93*, Toulouse, CERT, France, vol. 3, pp 817-836, 1993.

[7] P. M. Frank, X. Ding, "Frequency domain approach to optimally robust residual generation and evaluation for model-based fault diagnosis," *Automatica*, vol. 30, pp. 789-804, 1994.

[8] R. Isermann, "Fault diagnosis of machines via parameter estimation and knowledge processing," *Automatica*, vol. 30, pp. 815-836, 1993.

[9] O. O. Oyeleye, F. E. Finch, M. A. Kraner, "Qualitative modeling and fault diagnosis of dynamic processes by MIDAS," *Chemical Engineering Communications*, vol. 96, pp. 205-228, 1990.

[10] D. C. Montgomery, "Introduction to statistical quality control", 2nd edition, Wiley, New York, 1991.

[11] L. C. Alwan, "CUSUM quality control-multivariate approach," *Commun. Stat. - Theory Method*, vol. 15, pp. 3531-3535, 1986.

[12] C. A. Lowry, W. H. Woodall, C. W. Champ, and S. E. Rigdon, "A multivariate exponentially weighted moving average control chart," *Technometrics*, vol. 34, pp. 46-51, 1992.

[13] H. Tong, C. M. Crowe, "Detecting persistent gross errors by sequential analysis of principal components," *Computers and Chemical Engineering* vol. 20, pp. 733-738, 1996.

[14] S. Wold, K. Esbenson, P. Geladi, "Principal component analysis," *Chemometric and Intelligent Laboratory Systems*, vol. 2, pp. 37-52, 1987.

[15] P. Geladi, B. R. Kowalski, "Partial least squares regression: A tutorial", *Analytica Chimica Acta*, vol. 185, pp. 1-17, 1986.

[16] W. E. Larimore, "System identification, reduced-order filtering and modelling via canonical variate analysis," *Proc. 1983 Automatic Control Conf.* pp. 445-451, 1983.

[17] W. E. Larimore, "Canonical variate analysis in identification, filtering, and adaptive control," *Proc. of the 29th Conference on Decision and Control*, Hawaii, pp. 596-604, 1990.

[18] M. Verhagen, P. Dewilde, "Subspacemodel identification part 1. the output-error state-space model identification class of algorithm," *International Jounal of Control*, vol. 56, pp. 1187-1210, 1992.

[19] P. VanOverschee and B. DeMoor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, pp. 75-93, 1994.

[20] D. J. H. Wilson, G. W. Irwin, and G. Lightbody, "RBF Principal manifolds for process monitoring," *IEEE Transactions on Neural Networks*, vol. 10, No. 6, pp. 1424-1434, 2000.

[21] K. S. Narendra, "Neural networks for control: theory and practice," *Proc. IEEE*, vol. 84, No. 10, pp. 1385 - 1406, 1996.

[22] Y. Tan, M. Saif, "Neural-networks-based nonlinear dynamic modeling for automotive engines," *Neurocomputing,* vol. 30, pp. 129-142, 2000.

[23] T. J. Hastie, W. Stuetzle,"Principal curves," *J. Am. Stat. Assoc.,* vol. 84, pp. 502 -516, 1989.

[24] E. B. Martin, A. J. Morris and J. Zhang, "Process performance monitoring using multivariate statistical process contol," *IEE Proc. -Control Theory Appl.,* vol, 143, No. 2, pp. 132 - 144, 1996.

[25] G. W. Cottrell, J. Metcalfe, "EMPATH: face, emotion recognition using holons," *Advances in Neural Information systems III,* pp. 564-571, 1991.

[26] B. A. Golomb, D. T. Lawrence and T. J. Sejnowski, "SEXNET: a neural network identifies sex from human faces," *Advances in Neural Information processing Systems III* (Denver, 1990), pp. 572-577, Morgan Kaufmann San Mateo, CA, 1991.

[27] S. Tan and M. Mavrovouniotis, "Reducing data dimensionality through optimizing neural-network inputs," *AIChE J.,* vol. 41, No. 6, pp. 1471-1480, 1995.

[28] T. Stamkopoulos, K. Diamantaras and N. Maglaveras, "ECG analysis using nonlinear PCA neural networks for ischemia detection," *IEEE Transactions on Signal Processing,* vol. 46, No. 11, pp. 3058-3067, 1998.

[29] R. Rico-Martinez, K. Krischer, and I. G. Kevrekidis, "Continuous-time nonlinear signal processing of Cu elect data," *Chem. Eng. Commun.,* vol. 118, pp. 25-48, 1992.

[30] D. Dong, T. J. Mcavoy, "Non-linear principal component analysis based on principal curve and neural networks," *Proc. ACC,* Baltimore, USA, pp. 1284-1288, 1994.

[31] B. D. O. Anderson, J. B. Moore, editors, *Optimal filtering,* Prentice Hall, Englewood Cliffs, NJ, 1979.

[32] K. I. Diamantaras, S. Y. Kung, "Principal components neural networks: Theory and Applications," New York: Wiley, 1996.

[33] J. Karhunen, E. O. L. Wang, R. Vigario, and J. Joutsensalo, "A class of neural networks for independent component analysis," *IEEE Transactions on Neural Networks,* vol. 8, no. 3, pp. 485-504, 1997.

[34] J. Karhunen, J. Joutsensalo, "Representation and separation of signals using NLPCA type learning," *Neural Networks,* vol. 7, no. 1, pp. 113-127, 1994.

[35] J. Karhunen, "Optimization criteria and NLPCA neural networks," *Proc. IEEE Int. Conf. Neural Networks,* Orlando, FL, pp. 1241 - 1246, Jun. 1994.

[36] J. Karhunen, J. Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks," *Neural Networks,* vol. 8, no. 4, pp. 549-562, 1995.

[37] S. Haykin, "Neural Network," Prentice Hall, Upper Saddle River, New Jersey 07458, 1999.

[38] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.,* vol. 37, No. 2, pp. 233-243, 1991.

[39] B. Lerner, H. Guterman, M. Aladjem, and I. Dinstein, "A comparative study of neural network based feature extraction paradigms," *Pattern Recognition Letters* vol. 20, No. 1, pp. 7-14, 1999.

[40] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, no. 5, pp. 359-366, 1989.

[41] G. Cybenko,"Appoximation by superpositions of a sigmoidal function," *Math. Control, Signal Systems,* vol. 2, pp. 303-314, 1989.

[42] E. C. Malthouse, "Limitations of NLPCA as performed with generic neural networks," *IEEE Transactions on Neural Networks,* vol. 9, no. 1, pp. 165-173, 1998.

[43] H. Hotelling, "Relations between two sets of variates," *Biometrika,* vol. 28, pp. 321- 377, 1936.

[44] H. Akaike, "A new look at statistical model identification," *IEEE Trans. Auto. Cont.,* vol. 19, pp. 716-723, 1974.

[45] A. Raich, A. Çinar, "Statistical process monitoring and disturbance diagnosis in multivariable continuous processes," *AICHE J.,* vol. 42, pp. 995-1009, 1996.

[46] A. Negiz, A. Çinar, "Statistical monitoring of multivariable continuous processes with state space models," *AICHE J.,* vol. 43, pp. 2002-2020, 1997.

[47] A. J. Fossard, D. Normand-Cyrot, editors, "Nonlinear Systems Stability and Stabilization," Chapman & Hall, NY 10119, New York, USA, 1996.

[48] C. D. Schaper, W. E. Larimore, D. E. Seborg, and D.A. Mellichamp, "Identification of chemical processes using canonical variate analysis," *Comput. Chem. Eng.,* vol. 18, pp.55 - 62, 1994.

[49] M. Blanke, S. Bϕgh, R. B. Jϕgensen, and R. J. Pattern, "Fault detection for diesel engine actuator - a benchmark for FDI," *Proc. of IFAC Symposium on Fault Detection, Supervision and Safety for Industrial Processes,* vol.2, pp. 498-506, 1994.

[50] E. A. García, B. Köppen-Seliger, P. M. Frank, "Frequency domain approach to residual generation at the industrial actuator benchmark test," *Proc. of IFAC Symposium on Fault Detection, Supervision and Safety for Industrial Processes,* vol. 2, pp. 514-518, 1994.

[51] X. Ding and P. M. Frank, "Fault detection and identification via frequency domain observation approaches," *mathematical and Intelligent Models in System Simulation,*, IMACS, pp. 471-476, 1991.

[52] J. V. Kresta, J. F. Macgregor, and T. E. Marlin, "Multivariate statistical monitoring of process operation performance," *Canadian J. Chem. Eng.,* vol. 69, pp. 35-47, 1991.

[53] E. I. Knudsen, S. DuLac, and S.D.Esterly, "Computational maps in the brain," *Annual Review of Neuroscience,* vol. 10, pp. 41-65, 1987.

[54] T. Kohonen, "The self-organizing map," *Proc. of the Institute of Electrical and Electronics Engineering,* vol. 78, pp. 1464-1480, 1990.

[55] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics,* vol. 22, pp. 159 -168, 1982.

[56] T. Kohonen, "Self-Organizing and Associative Memory," Third edition, New York: Springer-Verlag, 1988.

[57] T. Kohonen, "Exploration of very large databases by self-organizing maps," *International Conference on Neural Networks,* vol. I, pp. 1-6, Houston, 1997.

[58] T. Kohonen, "Self-Organizing Maps," Third edition, New York: Springer, 2001.

[59] K. Obermayer, H. Ritter, and K. Schulten,"Development and spatial structure of cortical feature maps: A model study," *Advances in Neural Information Processing Systems,* vol. 3, pp. 11-17, 1991.

[60] H. Ritter, T. Martinetz, and K. Schulten, "Neural computation and self-organizing maps: An introduction," Reading, MA: Addison-Wesley,1992.

[61] S. Kullback, "Information Theory and Statistics," Gloucester, MA: Peter Smith, 1968.

[62] G. Dolanc, A. Rakar and et al.,"Three-tank Benchmark Tesk," Jozef Stefan Institute, issue C, June, 1997.

[63] G. Beni, X. Liu, "A least biased fuzzy clustering method," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 16 , no. 9, pp. 954-960, 1994.