A COLLABORATIVE, MULTI-AGENT BASED METHODOLOGY FOR ABNORMAL EVENTS MANAGEMENT

NG YEW SENG

NATIONAL UNIVERSITY OF SINGAPORE

2006

A COLLABORATIVE, MULTI-AGENT BASED METHODOLOGY FOR ABNORMAL EVENTS MANAGEMENT

NG YEW SENG

(B. Eng., UTM, Malaysia)

A THESIS SUBMITTED

FOR THE DEGREE DOCTOR OF PHILOSOPHY

DEPARTMENT OF CHEMICAL AND BIOMOLECULAR ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgements

This thesis is by far the most significant scientific accomplishment in my life and it would be impossible without the people who supported me and believed in me. I would like to take this opportunity and thank them here.

First, I would like to express my deepest gratitude towards my research supervisor, Prof. Raj. Srinivasan for his continued guidance and support throughout the course of this research. He is not only a scientist with great vision but also most importantly a resourceful thinker, whose ideas stimulate developments in many areas throughout the course of this research. His trust and scientific excitement inspired me and I am glad to work with him.

I sincerely thank Prof. Rangaiah Gade Pandu and Prof. Lim Khiang Wee, whom constituted and chaired my research panel. Their frank and open suggestions shed light into new interesting research topics, sometimes remedying my shortsightedness in my research work.

I would like to express my thanks to some academic staffs whom I have worked with while serving as a teaching assistant, they include Prof. I.A. Karimi, Prof. L. Samavedham, and Prof. K. Hidajat. Special thanks are also extended to the ever-helpful departmental staffs Mr. Qin Zhen, Ms. Tay Chun Yen, and the collaborators at Bioprocessing Technology Institute (BTI), Dr. Steve Oh, Mr. Ow Siak Wei Dave, and Ms. Lee Chai Lian, for their help in the MRI Imaging and fermentation projects.

I would like to thank all my lab mates, Jonnalagadda Sudhakar, Arief Adhitya, Mukta Bansal, Nguyen Trong Nhan, Mohammad Iftekhar Hossain, Li Jie, Manish Mishra, Qian Mingsheng, and Wang Cheng for maintaining a healthy, enjoyable and pleasant working environment. I would like to place my thanks to friends at Institute of

Chemical & Engineering Sciences (ICES), Seema Manuja, Iskandar Halim, Naraharisetti Pavan Kumar, Zhou Ying, and Doan Xuan Tien.

I am also very grateful to my friends in National University of Singapore, whom I have enjoyed spending most of my leisure time with. They include Ayman Daoud Allian, Rao Raghuraj, Liu Yu, Naveen Bhutani, Sukumar Balaji, Murthy Konda, Balla Ganesh, Cheng Cheng, etc.

Finally, I would like to express my deep gratitude and love for my parents, my brother, my sister-in-law, and my fiancée Jessica Zhang Xin, who wholeheartedly supported me in my work. Without their best wishes and blessings, I would not have been where I am currently.

Table of Contents

Summary

Modern chemical plants have complicated unit operations with considerable recycles. The complex controls and instrumentation installed often compensate and conceal faults, causing many faults in the process to remain undetected, until serious consequences occur. This thesis strives to explore new methodologies suitable for fault detection and identification (FDI) during transient mode of operations. Though the emphasis of this thesis is mainly on transient operations, the proposed methodologies are generic and can be applied to steady-state operations as well.

A novel framework based on multi-agent approach has been developed for detecting and diagnosing faults in the process industries by integrating various data-driven fault detection and identification techniques. Three major data-driven approaches, namely, self-organizing map (SOM), principal components analysis (PCA), and kernel density estimator (KDE) were extended in this thesis to the domain of transient operations.

The SOM belongs to the category of unsupervised neural-networks and is able to project high-dimensional data to two dimensions. The proposed SOM methodology utilizes cluster analysis approach for data representation, in which process operations (both steady-state and state transition) can be tracked and abstracted as a one-dimensional sequence. These sequences provide a unique signature for a given operation and are used for identifying known process faults based on syntactic pattern recognition.

The PCA approach has been popular in process monitoring. However, an in-depth analysis of PCA-based approaches reveals that the method is unsuitable for transient states since the associated statistics for monitoring are prone to errors during these mode of operations. These shortcomings are overcome through a novel modeling

strategy based on multiple overlapping PCA models. This allows each model to overlap with its neighbors to enable continuity in modeling transient operations. An optimal PCA model is then chosen at every instant for online monitoring.

A new monitoring statistic has also been proposed based on KDE to substitute the widely used Hotelling's T^2 statistic for monitoring of transient operations. Since Hotelling's T^2 statistic is based on *F*-distribution in data density modeling, it is unsuitable for transient operations. Therefore, a KDE-based statistic, which does not require a parametric model, is proposed. The KDE-based statistic can be used with any arbitrary distribution, and is suitable for most process operations.

Finally, a collaborative, software multi-agent based framework is developed to integrate these heterogeneous FDI methods. The framework, which is designated as Collaborative Agents for Managing Efficient Operations (CAMEO), contains different FDI methods, each modeled as a software agent in an interactive multi-agent environment. Each monitoring agent observes the process in real-time and flags abnormalities independently. Collaboration among these methods is achieved through a standardized communication formalism. The resulting conflicts between the agents are resolved through decision fusion algorithms that consider the results of FDI agents and fuse them. The decision fusion strategy is the logic for enforcing consistency among different agents within CAMEO and the bedrock for the collaboration mechanisms. Extensive testing of the proposed method to multiple case studies demonstrates the method's ability to reduce both Type-I and Type-II errors, and speed up time of fault detection and diagnosis considerably compared to any single application of FDI technique. The four developments reported above have been tested extensively using various case studies – the Tennessee Eastman problem, pilot scale distillation unit startup, and a simulated fed-batch operation.

List of Figures

List of Tables

Nomenclature

Subscript

i index used for process time representation (row of a given matrix)

j index used for representing various fault classes

r index used for number of classifiers, agents in the multi-agent architecture

n index used for representing process variable (column of a given matrix)

Parameters

C Total number of principal components retained

I The total number of samples (length) of a training data X

J Total number of fault patterns available in a fault database

K Total number of partitions used for clustering algorithm (fuzzy clustering)

N Total number of variables in a multivariate data

Variables

σ Standard deviation of the training data x

A A software agent used in the multi-agent environment

C_j The class representation of data

c_k Centroid of k^{th} cluster obtained from k-means algorithm

e Residual matrix after PCA decomposition

$E(x)$ Combined classification results after decision fusion

$F(\alpha)$ F-distribution used in evaluating upper control limit for T^2 statistic

F_j Faulty data used for pattern matching

H Alignment matrix for identifying optimal alignment between two sequences

h Bandwidth selector / smoothing parameters

H Scoring matrix for matrix propagation during aligning sequences

h^{opt} Final smoothing parameters used for online monitoring

K	A kernel function
K^p	Cohen's Kappa Statistic for measuring inter-classifier agreements
l	Time-lag dynamics incorporated in DPCA model
$m^{\exp}$	Optimal operating conditions identified for current signal x_i
M^R	BMUs of normal operation identified from SOM
p	Loading matrix identified from principal components decomposition
$P(A\|B)$	Conditional probability of A given evidence B
p_c	Proportion of agreement among classifiers expected by chance
p_o	Observed proportion of agreement among classifiers
$Q(\alpha)$	Upper control limit for SPE with (1-α) confidence
r	Radius of the neighborhood function h used during training of SOM
S_{pca}	Similarity factor for comparing two PCA models
SPE_i	Squared prediction error associated with PCA decomposition
t	Scores matrix from principal components projection
$T^2(\alpha)$	Upper control limit for T^2 statistic
T_i^2	Hotelling's T^2 statistic at time i^{th}
x	A one dimensional vector $x \in \Re^N$
X	An autoscaled matrix with I samples and N variables
$X(t)$	Measurements obtained till current time t
X_i^D	Autoscaled data, X augmented with time-lag samples
Y	Filtered signals obtained from plant sensors
Γ	Similarity degree between two data
δ_j	Variables contribution residual for D-statistic
λ_c	Eigenvalue of c^{th} principal component
Π	Fault maturity degree

τ Fault distortion measurement index

τ^{β} Upper control limits used for τ

$\epsilon(p)$ Total square error for p^{th} replicate of k-means iteration

κ *C*lassifier used for classification of input sample

Chapter 1 Introduction

1.1 Introduction to Monitoring and Fault Diagnosis

As chemical plants and refineries grow in complexity, the process of detection, diagnosis and correction of abnormal situations becomes increasingly difficult for plant engineers and operators. Modern chemical plants have long sequential unit operations with considerable recycles. The complex controls and instrumentation installed often compensate and conceal faults. Consequently, most faults in processes are often remain undetected until serious consequences occur, *i.e.*, shutdowns, equipments malfunctions, or catastrophic accidents such as fires or explosions. Methods of fault detection and diagnosis that improve unit availability, and reduce maintenance costs thus merit serious attentions (Himmelblau, 1978).

The chemical industries have rated abnormal events management (AEM) as a major problem with huge economic impact every year. Early detection and diagnosis of process faults while the plant is still in a controllable region can prevent progression of abnormal events into accidents and the resultant losses. From an economic viewpoint, Nimmo (1995) reported that approximately 20 billion dollars of annual losses in U.S. was due to poor Abnormal Event Management (AEM) while Laser (2000) reported that the impact of AEM on British economy was estimated at 27 billion dollars. Industrial statistics also show that minor accidents are very common, occurring on a day-to-day basis, causing injuries, illness to plant personnel, and costing investors billions of dollars each year. It was also reported that about 70% of the industrial accidents are caused by human errors (Bureau of Labor Statistics, 1998; McGraw-Hill Economics, 1985; National Safety Council, 1999; Venkatasubramanian *et al.*, 2003a). The severe consequences of abnormal events on humans safety and

economics offer motivation for this PhD work, which aims to resolve the difficulties faced by people in the process industries. The main area of investigation for major part of this thesis centers on the more challenging transient operations. Definitions for some of the important terms in this thesis are first established before a more thorough discussion on transient operations.

- *Transitions*: Operations that induce large changes to plant operating conditions. The magnitude of the state-variables usually alters significantly when a process undergoes transition.
- *Modes*: Operating regions correspond to steady-state operations. The variables magnitude often fluctuates within a small limit when a process is operating in a mode.
- *Fault*: Any departure from an acceptable range of an observed variable or calculated parameter associated with a process (Himmelblau, 1978).
- *State identification*: The task of locating the current process status, or state, based on measurable variables obtained from plant sensors.
- *Fault detection*: The task of determining the health of a process. A process can be either in the state of normal or abnormal.
- *Fault diagnosis*: The task of locating the root cause of an abnormal behavior, which constitutes the main reason for the deviations among process variables from the acceptable range of normal plant operations.
- *Fault candidate*: A set of possible explanations for the plant's abnormal behaviors. Explanations are usually derived using some analytical or artificial intelligence techniques.
- *Type*-I *Errors*: False positives resulting from a fault detection or diagnosis algorithm, usually associated with the wrong prediction of abnormality.

- *Type*-II *Errors*: False negatives resulting from a fault detection or diagnosis algorithm, usually associated with inability to correctly detect or diagnose a fault.

1.2 Introduction to Transient Operations

Increasingly, manufacturing facilities operate at a multitude of states and frequently switch between them. The switch from one state to another is termed as a process transition. Plant startups and shutdowns are common examples of transitions in the process and allied industries including refining, petrochemicals, paper & pulp, steel, and cement manufacture. Other transitions occur due to feedstock, throughput, or product slate changes as well as maintenance operations such as furnace decoking or absorber regeneration. Transient operations are also common in high-value added specialty and pharmaceutical plants which commonly operate in batch and fed-batch phases. Particulate operations such as crystallization, drying, filtration, etc, whose monitoring and control is becoming increasingly important in the pharmaceutical and formulated product industries, are also operated under transient states.

Process transitions commonly entail large changes in the plant operating conditions. Plant operators therefore perform transitions manually following predefined standard operating procedures (SOP), which clearly state the sequence of actions that need to be taken, e.g.: open or close valves, activate or deactivate equipments, reconfigure controllers, etc. However, owing to the lack of effective automation and the high cognitive load for operators, the occurrence of human errors during transitions is quite common. Survey conducted in the oil and gas industries also revealed that human errors, especially during transitions, are the leading cause of abnormal situations (Nimmo, 1995). A key feature of transient operations is that small changes in the plant operating conditions during critical periods can degrade the

quality of the final product; this is especially obvious in biological processes. Due to the numerous complexities in these modes of operations, effective techniques for online monitoring are essential since timely corrective action can prevent fault propagation and allow a batch or product to be saved. Furthermore, online monitoring would also engender safe operations as the occurrence of abnormal events can be minimized.

1.3 Challenges in Monitoring Transient Operations

Operators and control engineers face tremendous challenges during process transitions. These challenges range from day-to-day operational challenges to transition modeling and monitoring. Control and operation challenges relate to the difficulties faced by plant personnel when operating and maneuvering logic controllers and plant equipments during transitions. On the other hand, modeling challenges relate to the difficulties in developing models (either first principle or data-based) suitable for control and monitoring transitions.

1.3.1 Control and Operation Challenges

The control and operational challenges for monitoring transient operations are as follows:

1) *State and time-based events*: Transient operations can be both condition and time-based. The tasks to be followed are often governed by a prespecified condition, e.g.: pH reaches a certain threshold or a predefined "milestone" during operation. Other tasks may be associated with time based actions, e.g.: heating for a prespecified duration, fixed time crystallization, etc.

2) *Multivariate multi-scale processes*: A process plant is usually observed through hundreds or thousands of sensors. Each recorded variable might display a trend that is

unique. Multi-time scale effects also become important where some variables change quickly (order of seconds) and others respond over hours. Consequently, monitoring and tracing the root cause in the event of a fault can be difficult.

3) *Inadequacy of regulatory control*: Most current-day DCS are configured for steady-state control and are not effective during process transitions. Therefore, it is very common for plant operators to perform transitions manually following standard operating procedures (SOP) and transfer the control to the automation system only upon reaching steady-state.

4) *Run-to-run deviations*: In many occasions, a process run (especially in batch operations) might be completed much earlier or later compared to another, as operating practices might be very dissimilar due to differences in initial condition, impurity profile, etc. At times, even the strictest adherence to SOP by plant operators might result in deviations of final product quality due to such exogenous environmental or process factors (pharmaceutical processes).

5) *Manual operations*: Operators need to attend to numerous tasks during process transitions, which include tracking of important trajectories, executing standard operating procedures, attending to important alarms, synchronizing actions with other operators, etc. In addition, operators need to constantly watch out for other business related factors which include: (i) minimizing operating cost, (ii) ensuring process is safely operated, (iii) adhering to emission limits, and (iv) ensuring final products adhere to regulatory specifications, and customers / consumers expectations. The resulting high workload of plant operators increases the likelihood of human errors.

1.3.2 Modeling Challenges

The modeling challenges for monitoring transient operations are as follows:

1) *Nonlinearity and discontinuity*: Transient operations often display nonlinear and discontinuous behaviors. Such behaviors obviate simple linear models and complicate the task of process model development.

2) *Non-stationary states*: Process dynamics during transitions are often displayed as large changes in plant operating conditions with normal operation state as trajectories rather than a fixed setpoint. The switching of process modes or evolution of process phases often require different flowsheet configuration through shutting down or starting up new equipments. Consequently, constructing a comprehensive model for process monitoring during transient states can be difficult.

3) *Inability to model human interactions*: Human interactions with the process are often complex and difficult to predict. There is a lack of modeling techniques to-date that can be used to detect and rectify abnormal operations caused by human errors.

The above challenges are unique to transient operations. Hybrid discrete-continuous behavior of the process therefore has to be considered when monitoring these modes of operations. This thesis seeks to overcome the difficulties encountered during transient mode of operations. A formal description of the thesis objective is stated in the following section.

1.4 Objective of Thesis

This thesis strives to explore new methodologies suitable for fault detection and identification (FDI) during transient mode of operations. Though the emphasis of this thesis is on transient operations, the proposed methodologies are generic and can be applied to steady-state operations as well. The FDI methodologies developed are

centered on data-driven approaches. The attractive features offered by data-driven approaches such as the ability to be scaled-up (deployed) in short duration when there are an abundance of information rich data, and their potential of finding wider range of applications (beyond the domain of chemical processes) are the primary motivations for their selection. A conceptually sound means of integrating heterogeneous FDI methods is sought to improve the performance of FDI during transient operations. Towards this end, an efficient and scalable multi-agent based methodology is sought to integrate the strengths of numerous FDI methods. Since each FDI method is computationally complex, an efficient means of speeding up the response time of the integrated FDI system is also desirable.

1.5 Thesis Overview and Organization

The rest of this thesis is organized as follows. In Chapter 2, a categorization of existing FDI methods is presented followed by a literature review of various data-based modeling techniques. Most methods in the literature are compared and shown to be inadequate for transient operations. There is hence a need to develop new methodologies capable of covering this critical region of plant operations. Three different data-driven approaches, namely self-organizing map, statistical process monitoring, and kernel density estimator are selected in this thesis for further improvements. The conceptual review of these methods are also presented. Since each FDI method exhibits strengths and drawbacks that are process dependent, there is a strong motivation for the development of a collaborative approach for FDI to bring together the strengths from different classes of FDI methods. The rationale for such an approach is based on the precept that the strengths of different methods can be brought to bear on the problem and the drawbacks of any individual method overcome through collaboration.

The development of Self-organizing Map (SOM) to represent and compare process operations is a major contribution in this research work and is presented in Chapter 3 and Chapter 4. SOM belongs to the category of unsupervised neural-networks and has been gaining much attention lately for its ability to project high-dimensional data to two dimensions. The trained SOM can serve as a high-fidelity model of process operations and different types of operation can be visualized as a series of best matching units (BMUs). Steady-state operations are often represented as a cluster on SOM while transient operations are represented as trajectories. With additional clustering, process operations can be abstracted as one-dimensional sequences on SOM. The generated sequences provide an unique identity for a particular operation and can be used for identifying known process faults based on syntactic pattern recognition. The method also supports the detection and identification of novel faults based on the variable residuals while comparing two process runs: a reference run compared to the actual operation.

Principal Components Analysis (PCA) has been a popular method for process monitoring. Mathematically, PCA relies upon eigenvector decomposition of the covariance or correlation matrix to capture the major trends of process variables. However, in-depth analysis of PCA-based approaches revealed that the method is unsuitable for transient operations. Though PCA shows high accuracy in data-modeling, its associated statistics for process monitoring are subjected to errors during transient mode of operations. The existing PCA-based statistics assume that the training data follows a standard normal distribution, which does not hold for most transient processes. The resulting consequences of such assumptions are a significant increase in Type-I and Type-II errors when these statistics are applied during transitions. In Chapter 5, an adjoined multi-dynamic PCA (ADPCA) modeling

approach is developed to overcome this shortcoming. The proposed technique uses multiple PCA models that are allowed to overlap with their neighbors to enable continuity in modeling transient operations. For online application, an optimal PCA model is selected at every instant for process monitoring. Extensive testing of the proposed method demonstrates the methods' ability to reduce both types of errors (Type-I and Type-II errors) compared to existing PCA-based monitoring technique. Detailed comparison between the proposed ADPCA technique with some other popular variants of PCA, *i.e.*, multiway-PCA and dynamic-PCA are also presented.

In Chapter 6, a Kernel-Density Estimation (KDE) based PCA approach is developed for fault detection and identification during transient processes. Density estimation is the construction of an estimation of the density function (data distribution) from the observed data. The conventional means for process monitoring is based on parametric form of density estimation, by assuming that the data will follow a known density function, *i.e.*, normal, *F*-distribution, χ^2, etc. Appropriate bounds on the density model are used to select the confidence limits for monitoring. Unlike the parametric approaches, the KDE does not require any prior assumption of the data distribution, instead the density model is estimated from the data itself. In this chapter, the KDE approach is extended to the transient operation regime. A new monitoring statistic is proposed to substitute the widely used Hotelling's T^2 statistic. Hotelling's T^2 statistic follows a *F*-distribution in data density modeling, and is unsuitable for modeling transient operations. Since KDE is bi-variate in nature, different combinations of the latent variables can be unified and integrated for multi-dimensional KDE analysis. An attractive feature of KDE is that it can be used with arbitrary distributions. The method can hence be generalized to most process operations, *i.e.*, in both the domain of steady-state and transient operations. Since

KDE-based monitoring statistic exhibits high accuracy in distinguishing data classes through unique confinement of data boundary, the KDE-based statistic is found to be very effective for fault identification.

Each FDI method has its corresponding strengths and shortcomings that are process dependent. A method that works well under one circumstance might not work well under another when different features of the process come to the fore. Since each developed FDI methods can be considered as an independent entity with similar objective (timely, accurate FDI during process operations), a collaborative, multi-agent based framework is developed in Chapter 7 to integrate heterogeneous diagnostic classifiers. A software agent can be viewed as an identifiable computational entity that automates some task or decision making to benefit humans. The framework developed in this thesis, which is designated as Collaborative Agents for Managing Efficient Operations (CAMEO), models each FDI method as an agent, located in an interactive multi-agent environment. Collaboration among these methods is achieved through a standardized communication formalism. The agents within the multi-agent framework can be distributed across a cluster of computer nodes to exploit multiple processors. Each agent communicates with other agents through message passing, this allows the integration of computationally demanding FDI methods.

When multiple FDI methods are used in parallel, a conflict resolution strategy is needed to arbitrate among the contradictory decisions proposed by the various FDI methods, so that one consolidated solution can be presented to the plant personnel. The resulting conflicts within the multi-agent system can often be resolved through decision fusion where incongruous opinions among the FDI agents are weighted and fused. Three of the popular decision fusion strategies, namely, voting, Bayesian-

combination, and Dempster-Shafer fusion approaches are studied in Chapter 8. The strengths and shortcomings of each decision fusion strategy are critically evaluated.

Finally, a summary of the research is presented in Chapter 9 along with recommendations for future work in the area of fault detection and diagnosis. Some comments on the integration of the proposed work with other parts of plant operations are also provided.

Chapter 2 Literature Review

2.1 Monitoring of Transitions – An overview

Operations of a process can be classified into modes and transitions. A mode corresponds to the region of continuous operations under fixed flowsheet conditions; *i.e.*, no equipment is brought online or taken offline. During a mode, the process operates under steady state and its constituent variables vary within a narrow range (Srinivasan *et al.*, 2004). In contrast, transitions correspond to large changes / discontinuities in the plant operations; *i.e.*, change of setpoints, turning on or idling of equipments, maneuvering manual valves in a plant, etc. Due to the large alterations in magnitude of the observable plant variables, process transitions thus induce additional complexity to the complicated task of monitoring and fault diagnosis compared to its steady-state counterpart (see Section 1.2 for more thorough discussion on process transitions).

Despite the abundance of literature on fault detection and identification (Venkatasubramanian *et al.*, 2003a,b,c), only a few of these methods have been explicitly designed for process transitions. In this thesis, existing FDI methods for transient operations are categorized as two classes: namely qualitative models and quantitative models. The classification of the FDI methods is based on the functional form of the diagnostic model, *i.e.*, methods that are based on abstracted, trend, and causal analysis of process data are categorized as qualitative methods, while methods that use statistical or mathematical means for analysis of process data are categorized as quantitative methods (see Figure 2-1). A review of some of these FDI methods is presented next.

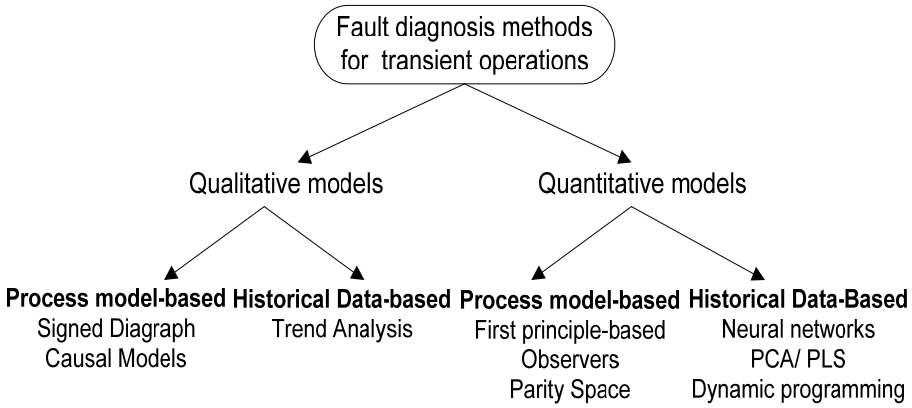


Figure 2-1: Existing approaches for monitoring transient operations

2.2 Taxonomy of Existing FDI Methods

As noted in previous section, existing FDI methods can be broadly classified into two categories namely qualitative model-based and quantitative model-based methods.

2.2.1 Qualitative Model-based Methods:

Qualitative model based methods include techniques such as trend analysis, rule-based systems and signed-digraphs. *Trend analysis* is based on the abstraction of process data into a set of trends (Cheung and Stephanopoulos, 1990). Monitoring is then performed on the identified trends, which are made up of primitives that describe the qualitative behavior of the process variables. Classical trend analysis approaches are based on monitoring an ordered set of primitives that describe the evolution of a process variable. When a fault occurs, process variables vary from their nominal ranges and exhibit trends that are characteristic of the fault. Hence, different faults can be mapped to their characteristic trend signatures. Extension of trend analysis to fuzzy reasoning is reported in Dash *et al.* (2003). However, the above mentioned trend characterization is not true during process transitions since each variable might display a different trend during different phases of the transition. There are also occasions where process exhibits different trends during transitions due to normal operating variations, thus complicating trend comparison. The same trends observed during different stages could have different implications. Classical trend analysis is therefore

not sufficient to monitor transitions adequately. Sundarraman and Srinivasan (2003) overcome the above problems through enhanced trends. In their approach, enhanced trends are composed of an ordered sequence of enhanced atoms. The later consists of shape, duration of trend manifestation, and magnitude of the starting and ending of a trend. The enhanced trends computed from real-time were compared to a trend dictionary computed offline from normal operating dataset. Three types of matching degrees: shape matching degree, magnitude matching degree, and duration matching degree were also introduced to facilitate trend comparison during transition. The main shortcoming of trend analysis is that it is designed for monitoring individual variables. For instance, it does not take into account the correlation between the variables in the process.

Rule-based systems, sometimes referred to as expert systems, use rules to perform monitoring. They are best suited to situations where plant operators have a good knowledge regarding the nuances of the transitions and the underlying process. Honda and Kobayashi (2000) used a fuzzy rule-based inference system for the direct control of batch operations. The process phase is first recognized by fuzzy inference, and then a fuzzy neural network based control system is used to control the batch process. They have illustrated their methods on three processes - mevalotin precursor production, Vitamin B2 production, and sake mashing. In Muthuswamy and Srinivasan (2003), a rule-based expert system is developed for automation and supervisory control of semi-batch fermentation processes. They characterized transitions using features in process variables and represented them as multivariate rules. These rules track the process across phases and automatically detect the current active phase using online data. Different monitoring rules are formulated for each phase of a transition. The rule-based transition characterization method was shown to be robust to measurement noise

and easily comprehendible to the operators. Nevertheless, rule-based systems are process specific, and at times it might be hard to extract rules to adequately model complex processes.

2.2.2 Quantitative Model-based Methods

First-principle models, statistical models, signal processing models, and neural networks are grouped under model-based systems. These are built either from first-principles knowledge or using input-output data (Venkatasubramanian *et al.* 2003c). Bhagwat *et al.*, (2003a) presented a non-linear model-based approach to monitor process transitions. Estimation of process states and residuals is achieved through open-loop observers and extended Kalman filters. To address the issues arising from the discontinuous nature of transition, the scheme uses knowledge of the standard operating procedure and divides each transition into phases. For the purpose of monitoring, each phase is associated with a model component and different filters and observers are selected for fault detection in that phase. However, accurate models of highly complex processes operating in multiple regimes are seldom available and difficult to develop, thus limiting their practical applicability. Multiple model-based approaches have therefore been used to model, control, and monitor transitions. Banerjee and Arkun (1998) proposed a strategy to control transient processes through an identification method that builds linear models for different operating regimes, and then interpolates nonlinear models in between these local models to match plant dynamics during transitions. Kosanovich *et al.* (1997) designed different linear controllers for different operating regions of a reactor. A supervisory control strategy that assesses plant-model mismatch was used to determine the switching logic when different scheduling policies are demanded. The use of multi-linear models to predict process trajectory during fermentation is illustrated by Azimzadeh *et al.* (2001). They

used Model Predictive Control (MPC) in cascade with PID controllers for driving the transition along the optimal trajectory. In Bhagwat *et al.* (2003b), a multi-linear model-based fault detection scheme was proposed based on decomposition of operation of a non-linear process into multiple locally-linear regimes. Kalman filters and open-loop observers were used for state estimation and residuals generation in each regime. Analysis of residuals using thresholds, faults maps, and logic-charts enabled on-line detection and isolation of faults.

Signal processing methods can be applied to analyze the normal/abnormal status of a process by comparing the online profile of process variables with those of previously known runs. The underlying methods perform time synchronization between process signals from different runs before comparing them based on predefined similarity metrics. Methods for signal processing include dynamic time warping (DTW) and dynamic programming (DP). DTW originated from the area of speech recognition and has found application in the chemical engineering domain recently. Some applications of DTW for process monitoring can be found in Gollmer and Posten (1996) and Kassidas *et al.* (1998a,b). One known shortcoming of DTW is its high computational cost which grows exponentially with the length of process data. This can be minimized by using landmarks such as peaks or local minima in the signals to reduce the complexity of signal comparison (Srinivasan and Qian, 2005). These landmarks, called singular points, can be used to align different runs. Singular points can be used first to decompose a long continuous signal into multiple, short, semi-continuous ones. DTW is subsequently applied on the short segments to perform monitoring during transitions. However, one known shortcoming of DTW algorithm is the essential requirement that the starting and ending points of the signals to be compared should coincide. Such shortcomings obviate their direct practice for online

applications since the points in the historical database that should be matched with the starting and ending points of the online signal are unknown. To overcome these shortcomings, Srinivasan and Qian (2006) proposed dynamic locus analysis which is an extension of the Smith and Waterman (1981) discrete sequence comparison algorithm for online signals comparison.

With the increasing availability of inexpensive sensors, the number of measured variables for most industrial processes easily ranges in thousands. This has lead to the popularity of multivariate statistical methods, which bring forth powerful means to monitor transitions. *Principal components analysis* (PCA) is one such multivariate dimensionality reduction technique that is widely used for developing data-driven models (Jackson, 1991). Applications of PCA and its variants for process monitoring can be found in Chiang and Braatz, (2003); MacGregor and Kourti, (1995); and Chen and Liu, (2002). Most of the reported work in multivariate statistical analysis is directed to processes where the correlation between the process variables remains the same. They are not directly applicable to transitions due to the statistical non-stationarity of the process and time-varying dynamics. In order to overcome this, an extension called dynamic PCA (DPCA) has been proposed (Ku *et al.*, 1995). In Srinivasan *et al.* (2004), DPCA has been used to classify process states based on historical operating data. Process data is first segmented into modes and transitions. Steady state modes are identified by using a moving window approach which is capable of rejecting outliers. A DPCA based similarity factor is used to compare transitions with historical data, which can be used for online FDI.

Neural-network based approaches are another popular area for fault diagnosis in continuous processes (Kavuri and Venkatasubramanian, 1993). They have been popular for classification and function approximation. Kapil *et al.* (2005) used neural-

network to control a fed-batch yeast fermentation process. They tracked the trajectory of fermentation with a recurrent neural-network that allows online adaptation, and showed that such adaptation allows the network to be used over a wide region outside its training domain. In Fabro *et al.* (2005), recurrent neural-networks were used to identify process states and predict process behavior. Control actions for different phases of transition are provided through sets of fuzzy controllers. They illustrated their approach through a distillation-column startup case study. Theoretically, artificial neural networks can approximate any well defined non-linear function with arbitrary accuracy. But unfortunately, there is no universal criterion for selecting a specific structure of neural-network for a practical application. Usually the structure of the network is decided based on the input dimensionality and the complexity of the underlying classes. The construction of an accurate neural classifier for such multivariate, multi-class temporal classification problem suffers from the "curse of dimensionality". To overcome the above drawbacks, Srinivasan *et al.* (2005a) proposed the use of two new neural network architectures, namely One-Variable-One-Network (OVON) and One-Class-One-Network (OCON). In both structures, the original classification problem is decomposed into a number of simpler classification problems. The OVON uses a sub-state identification layer where a set of neural networks are used to identify simpler univariate, temporal patterns. A unification layer is subsequently used to infer the process state based on the sub-states, through multi-dimensional, static pattern recognition. A state-identification layer is used to identify the presence or absence of a temporal pattern in multi-dimensions for OCON; the state of the process is inferred by analyzing the static, multi-dimensional outputs from the state-identification layer. Comparisons with traditional networks indicate that the new neural networks architectures are simpler in structure, faster to train, and yield

substantial improvement in classification accuracy. However, a priori knowledge of the sub-states of each variable is needed to derive the sub-state identification layer of OVON, which can be cumbersome for processes with large number of variables. High misclassification rate is also reported during state change when there is no clear separation between the states. Another way of improving classification accuracy of temporal patterns using neural-networks is to incorporate context-based information (previous process state). In Srinivasan *et al.* (2005b), a context-based neural network architecture called operating state identification neural-network (OSINN) was proposed for online state identification. They showed that feedback of context information to the input nodes of the network can improve classification accuracy.

Subsequent sections within this chapter will review important concepts and literature that are deemed necessary for new FDI methodologies development in this thesis.

2.3 Visualization Methods for Multivariate Temporal Data Analysis

Advancements in sensors and database technologies in chemical plants have resulted in the availability of huge amount of process data. Visual exploration methods, which facilitate humans to uncover knowledge, patterns, trends, and relationships from the data is hence crucial in understanding process operations, especially when multi-state operation and transitions between them is common.

Visualization techniques use graphical representation to improve human's understanding of the structure in the data. These techniques convert data from a numeric form into a graphic that facilitates human understanding by means of the visual perception system. The simplest approach by far is the coordinate plot using mutually orthogonal axis. However, this is useful only for 2- or 3-dimensional data.

Other techniques have been proposed to facilitate visualization of higher dimensional, including Andrews' curves (Andrews, 1972) and Chernoff's face (Chernoff, 1973). The Andrews' curve forces all variables onto a two-dimensional curve by transforming each N-dimensional observation $x_i = \{x_{i1}, ..., x_{in}, ..., x_{iN}\}$ to

$$f_i = x_{i1}/\sqrt{2} + x_{i2} \cdot \sin(t) + x_{i3} \cdot \cos(t) + x_{i4} \cdot \sin(2t) + x_{i5} \cdot \cos(2t) + ...$$

$$\text{where } t \in [-\pi \ \pi]. \qquad \text{(Eq 2-1)}$$

Chernoff's face uses different parts of the face such as ears, mouth, and eyes to represent different values observed from the data. However, these approaches have limited applicability to high-dimensional, temporal data.

The scatter diagram or scatter plot has been a popular tool for visualizing the correlation among multivariate data using two dimensional graphs by displaying all pairs of variables against each other (Rauwendaal, 1993). If the relationship between two variables is linear, the points on the corresponding subplot would fall on a straight line. An illustration is shown in Figure 2-2 based on 8 variables from the distillation column case study described in Section 4. The diagonal plots have been replaced by the histogram of the variable. From the figure, variables 1 and 2, 2 and 3, and 16 and 18 appear to be linearly correlated, while no direct correlation is apparent between variables 6, 9 and 11. Scatter diagrams have been used for visualizing gene expression data by Zhang *et al.*, (2003) and Craig *et al.*, (2005). However, they have limited used even for correlation analysis as the number of subplots to be analyzed grows as ${}^{N}C_{2}$. Consequently, more efficient techniques are required for high-dimensional data.

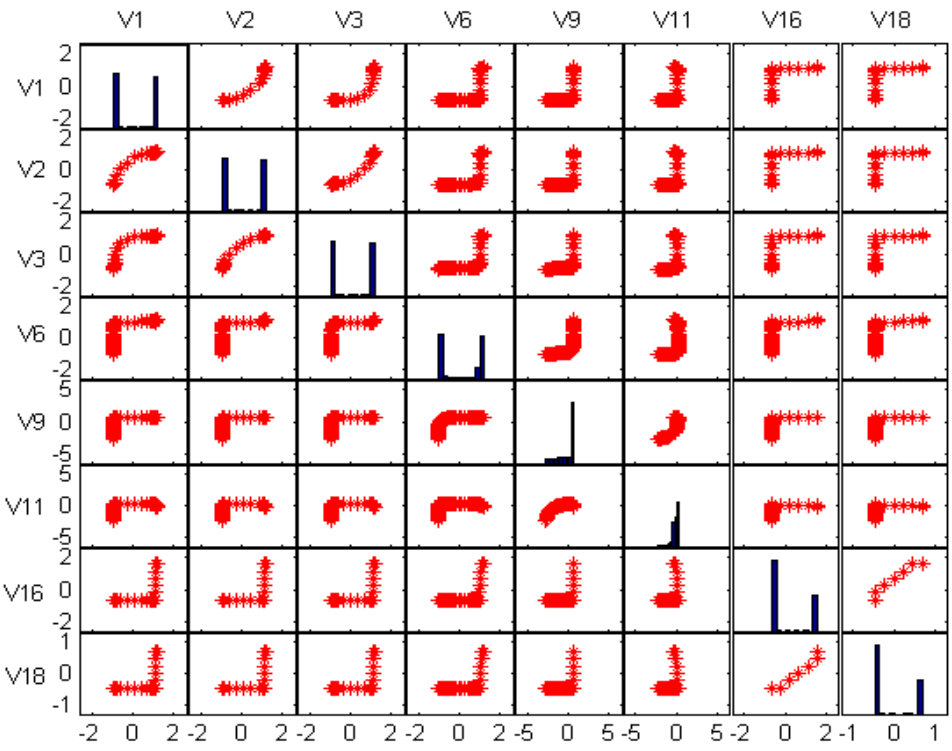


Figure 2-2: Scatter plot of eight variables from the distillation column data

Parallel coordinates was first proposed by Inselberg (1985) and is based on a dual representation of the Cartesian coordinates. Each dimension in parallel coordinates is represented by a vertical line, so all the coordinate axis are parallel instead of being mutually perpendicular. Each observation is represented by locating its i^{th} variable along the i^{th} axis and connecting all the N points for the observation by a line. A high-dimensional dataset is then captured completely in a 2-dimensional envelope of the polygonal lines representing all the observations (Inselberg *et al.*, 1987). An example of parallel coordinates is shown in Figure 2-3 for the same distillation column dataset as above. Inselberg (2002) applied the parallel coordinates for mining operational data from a Very Large Scale Integration (VLSI chip) production plant. From the representation, operating conditions that give higher yield during manufacturing could be extracted visually. Albazzaz *et al.* (2005) used parallel coordinate to visualize multidimensional data from a wastewater treatment plant.

However, they concluded that automating the parallel coordinates method remains a difficult task, so it was not found suitable for on-line visualization of large scale data.

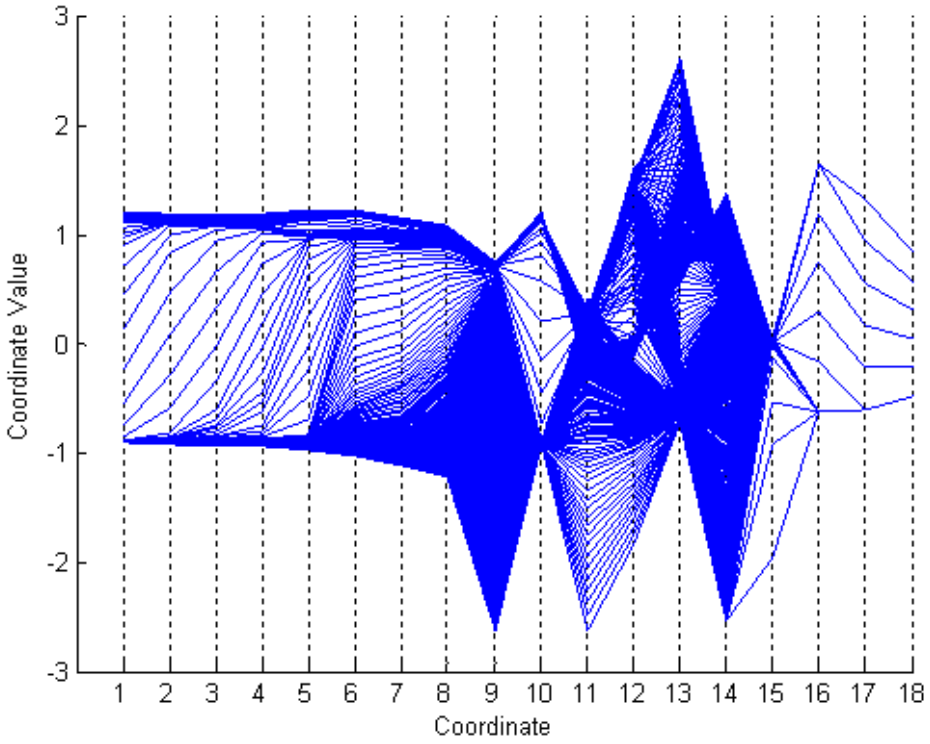


Figure 2-3: Parallel coordinate visualization of the distillation unit startup

Principal Components Analysis (PCA) is a popular statistical technique for dimensionality reduction and information extraction. PCA finds combination of latent variables that describe major variation in the data (Jackson,1991; Wise *et al.*, 1990). In general, only a few principal components are necessary to adequately represent the data. In such cases where the dimensions of multivariate data can be reduced effectively through PCA, visualization can be achieved through the biplot of the first few scores as they would explain the most important trends in the data. In Jokinen (1994), PCA was used for the visualization of an industrial continuous stirred tank reactor (CSTR) and distillation column. Six fault classes could be distinguished from the normal operation as clusters on a biplot. Sebzalli and Wang (2001) used a similar

technique to discover the various operating zones of an FCC process. Some other examples of PCA-based visualization can also be found in Mandenius *et al.* (1998) for visualizing state transitions in biopharmaceutical processes. Martin and Morris (1996) and Fourie and de Vaal (2000) applied it for process visualization and monitoring. Srinivasan *et al.* (2004) showed that for multi-state operations, process modes form clusters on the scores plot while transitions are depicted as trajectories. However, the use of PCA is not without limitation. Firstly, the linear approximation of PCA might not be sufficient to capture nonlinear relationships in the multivariate data. Also, often the first two or three principal components are not adequate for capturing all the important variance in the data, so depiction of observations in a 2- or 3-dimensional coordinate plot is not adequate. One solution for such cases is the use of parallel coordinates to visualize a larger number of scores as proposed by Wang *et al.* (2004). Finally, when multi-state operations are visualized, the scaling of each variable is dominated by the large variation during transitions; significant changes within a steady state would be obscured by the depiction. To overcome these shortcomings, a self-organizing map based methodology is developed in Chapter 3 for visualizing high-dimensional, multi-state operational data.

2.4 Process Modeling with Self-Organizing Map

Most of the existing methods for FDI are based on supervised learning. Application of such methods during transitions can be difficult when the state/class information of process transitions is unavailable. Also, different operators might demarcate a process into different states based on their own interpretation, rendering state annotation a difficult operation. To overcome this, Self-Organizing Map that is based on unsupervised learning can be used when such state critical information is unavailable.

The Self-Organizing Map (SOM) is an unsupervised neural network first proposed by Kohonen (1982). It is capable of projecting high-dimensional data to a two dimensional grid and thus can serve as a visualization tool. Self-organization means that the net orients and adaptively assumes the form that can best describe the input vectors (Kohonen, 1993). The SOM employs nonparametric regression of discrete, ordered reference vectors to the distribution of the input feature vectors. A finite number of reference vectors are adaptively placed in the input signal space to approximate the input signals.

Consider a dataset *X* containing *I* samples, each *N*-dimensional. *X* is therefore a 2-dimensional matrix of size *I x N*, with the i^{th} sample $x_i = \{x_{i1},...,x_{in},...,x_{iN}\}$. A SOM is an ordered collection of neurons. Each neuron has an associated reference vector $m_j \in \Re^n$ so, $m_j = \{m_{j1},...m_{jn},...m_{jN}\}$. Consider a SOM $M^{SOM} = \{m_1,...,m_j,...,m_J\}^T$ with *J* neurons, which has to be trained to represent and visualize *X*. This involves the calculation of the reference vector of every neuron. Initially, let each m_j be assigned a random vector from the domain of *X*. When a sample $x_i \in X$ is presented to the SOM for training, the neuron whose reference vector has the smallest difference from x_i is identified and defined as the winner or the *Best Matching Unit* (BMU) for that input:

$$b_i = \arg\min_j \left\| x_i - m_j \right\|, \forall j \in [1, J] \qquad \text{(Eq 2-2)}$$

The distance $\| \ \|$ between x_i and m_j is measured here using the Euclidean metric, but other metrics can also be used.

The neurons in M^{SOM} are usually placed in a two-dimensional grid. Let the location of the j^{th} neuron be r_j, where $r_j \in \Re^2$. A distance metric can be defined on the two-dimensional grid and all neurons up to a certain distance from the j^{th} neuron

considered its topological neighbors $\mathcal{N}_j$ in the grid. This concept of neuron neighborhood is the key differentiating feature of SOM and is responsible for its unique properties (described below). During training, when each sample $x_i \in X$ is presented, the reference vector of the *BMU*, m_{b_i}, *as well as those of its topological neighbors* in the grid are updated by moving them towards the training sample x_i. In its simplest form, the SOM learning rule at the t^{th} iteration is given as:

$$m_j(t+1) = m_j(t) + \alpha(t) h_{b_i j}(t) \left\| x_i(t) - m_j(t) \right\| \qquad \text{(Eq 2-3)}$$

where $\alpha(t)$ is the learning rate factor, and $h_{b_i j}(t)$ is a neighborhood function centered on the BMU b_i but defined over all the neurons in M^{SOM}. The Gaussian neighborhood function is commonly used and is given by:

$$h_{b_i j}(t) = \exp\left(-\frac{|| r_{b_i} - r_j ||^2}{2\sigma^2(t)} \right) \qquad \text{(Eq 2-4)}$$

where $\sigma(t)$ is the neighborhood width. During training, the neighborhood width is varied from iteration to iteration by changing $\sigma(t)$. A large value of σ and α is employed initially and is usually decreased monotonically with t (Kohonen, 2000). To guarantee convergence, it is necessary that as training proceeds and $t \to \infty$, $\alpha(t) \to 0$ and $\sigma(t)$ to a small value, typically 1. Other variants of the neighborhood function as well as training algorithms have been proposed in literature (Vesanto, 2002).

The updating of the reference vector of the neighborhood neurons *along* with that of the BMU's provides the topology-preserving feature of SOM, i.e., the neighboring neurons are activated and learn from the training input x_i and thus acquire similar reference vectors. The neighboring units are thus, in a sense, more similar to each other and the trained SOM maps similar input samples onto nearby neurons.

The training of the SOM requires the specification of two parameters – the number of neurons in the SOM, J and the aspect ratio of the two-dimensional grid. As recommended by Vesanto (2002), the following heuristic is used:

$$J = 8\sqrt{I} \qquad \text{(Eq 2-5)}$$

to specify the number of neurons and the square root of the ratio between the two largest eigenvalues of the covariance matrix of X to specify the aspect ratio.

Once the SOM has been trained, it can be directly used for classification of new samples. In this case, the index of the BMU can be considered as the class index. For any testing data x_i, a class can be assigned by finding its BMU. The goodness of fit of a testing sample x_i to its BMU, b_i, can be measured based on the *quantization error* (Vesanto, 2002):

$$E_i^q = \left\| x_i - m_{b_i} \right\| \qquad \text{(Eq 2-6)}$$

The quantization error measures the goodness of projection. A large value of E_i^q indicates a large difference between the sample x_i and the identified BMU, b_i, i.e., the x_i is not well represented by the SOM and is not very similar to any of the training samples.

The SOM can be used for visualization in two different ways. Firstly, the trained SOM can be thought of as a mapping from $X \in \Re^n$ to 2-D. The neurons in a trained SOM are not equally distributed among the whole input space $\Re^n$, rather more neurons are designated for regions with more samples in X (high density) and fewer ones for lower density regions in $\Re^n$. Therefore, one way of visualizing clusters in X is by means of the distance between a neuron and each of its neighbors (Ultsch and Siemon, 1990).

$$\mathcal{D}_{jj'} = \left\| m_j - m_{j'} \right\| \quad j' \in \mathcal{N}_j \qquad \text{(Eq 2-7)}$$

The unified distance matrix (U-Matrix) visualizes the SOM by depicting the boundary between each pair of neurons by a color or gray shade that is proportional to their $\mathcal{D}_{jj'}$. Alternatively, the average distance between the neuron and its neighbors can be used to color the neuron.

$$\mathcal{D}_j = \frac{1}{|\mathcal{N}_j|}\sum_{j'}\|m_j - m_{j'}\| \quad j' \in \mathcal{N}_j \qquad \text{(Eq 2-8)}$$

where, $|\mathcal{N}_j|$ is the size of the neighborhood (Ultsch and Siemon, 1990). Cluster borders are then indicated as "mountains" of high distances separating "valleys" of similar neurons. An example of a U-matrix is shown in Figure 3-1a of Chapter 3. From the U-matrix, three distinct clusters can be seen, as distinct valleys separated by mountains.

Secondly, clusters in the trained SOM can be labeled and directly used as a three-dimensional display to depict a new sample x_i from the space of X. This is particularly useful for identifying the class (cluster) of a new sample. The BMU corresponding to this new x_i is often referred to as its "hit". If new samples are regularly available from an online process, the state of the process at a point in time can be identified from the hit; the evolution of the process state can also be visualized by the sequence of the hits, as discussed in Section 4.

SOM has been used successfully in diverse fields. A comprehensive review of SOM applications is reported by Kaski *et al.* (1998). Xiao *et al.* (2003) used SOM for microarray data analysis and visualization of transcriptional changes in tumors. López-Rubio *et al.* (2003) adapted SOM for principal components analysis. Kolehmainen *et al.* (2003) used SOM to identify phases in the growth of yeast. While the previous applications of SOM are mainly for data clustering, recently SOM has also been applied to process monitoring and fault diagnosis.

Deventer *et al.* (1996) demonstrated how disturbances in a froth flotation plant can be visualized with SOM. They tracked changes in operating conditions by utilizing features extracted from froth images and visualizing the degree of dispersion through SOM. In Chan *et al.* (2001), a constrained Kohonen network was proposed to overcome the problem of monitoring redundant sensors by constraining the weight vectors in the parity space. Srinivasan and Gopal (2002) showed that SOM can be used to extract features during the operation of a fluidized catalytic cracking unit. Jämsä-Jounela *et al.* (2003) presented a SOM based fault diagnosis system for a smelter using heuristic rules. SOM was used to determine the coefficient for oxygen enrichment and detection of aggregations in various parts of the plant. Abonyi *et al.* (2003) applied SOM to estimate product quality in a polyethylene process. SOM was used initially for knowledge extraction of the historical data, and process monitoring was achieved through manual supervision of variables projection to SOM subspace. They also used SOM for quality estimation by incorporating quality information into SOM.

In Chapter 3 and Chapter 4, SOM-based visualization and FDI methods are described for diagnosing faults in transient operations. A SOM-based data representation structure is used to represent process operations.

2.4.1 Dynamic programming approaches to discrete sequence comparison

The discrete sequence comparison problem is generally formulated as follows: given two long sequences, find the largest number of elements from one sequence that can be matched with those from the second while allowing for local variations in either. Several approaches have been proposed in literature including the dynamic programming based techniques of Needleman and Wunsch (1970) and Smith and Waterman (1981). The former performs a global alignment while the latter local.

Global alignment implies that the two sequences are aligned over their entire length, while local alignment requires discovering and aligning only a locally conserved subsequence.

The Needleman-Wunsch algorithm takes two input sequences, say $A = \{a_1, a_2, ..., a_i, ..., a_I\}$ and $B = \{b_1, b_2, ..., b_j, ..., b_J\}$, and generates an alignment together with a score as an output. A distance between two sequences elements *a* and *b* is defined as $d(a,b)$. Typically, each exact match gets a positive score, each mismatch gets a penalty; so $d(a,b) \geq 0$ if $a = b$ and $d(a,b) = -\Phi$ if $a \neq b$. In order to find the maximum matches between the two sequences, an alignment matrix, *H* of size (*I+1*) x (*J+1*), is setup, whose element H_{ij} measures the similarity between substring $A = \{a_1, a_2, ..., a_i\}$ and $\{b_1, b_2, ..., b_j\}$. *H* is initialized with:

$$H_{i0} = i \times \Phi, \; i \in [0, I] \qquad \text{(Eq 2-9)}$$

$$H_{0j} = j \times \Phi, \; j \in [1, J] \qquad \text{(Eq 2-10)}$$

Other elements of *H* are calculated recursively as (Gusfield, 1997):

$$H_{(i+1)(j+1)} = \max(H_{ij} + d(a_{i+1}, b_{j+1}), H_{(i+1)j} + \Phi, H_{i(j+1)} + \Phi) \qquad \text{(Eq 2-11)}$$

The last element of *H*, $H_{(I+1)(J+1)}$, is normally referred to as the score of an alignment and is a measure of similarity between *A* and *B*. The best alignment between the two sequences can be located by sequentially back tracing from this element until the first, H_{11}, while selecting the best predecessor in each step. The Needleman-Wunsch algorithm has a computational complexity of *O*(*IJ*).

The Smith-Waterman algorithm is a variation of the above and replaces Eq (2-9) and (2-10) with:

$$H_{i0} = H_{0j} = 0 \qquad \text{(Eq 2-12)}$$

Therefore, local alignments become visible and the best matching subsequences can be identified from the highest score in the entire matrix (as versus the element H_{IJ}). In this thesis, a sequence comparison based approach is propsoed for process monitoring and diagnosis.

2.5 Process Modeling with Principal Components Analysis

Principal Components Analysis (PCA) is a popular statistical technique for process monitoring (Kourti, 2002). Mathematically, PCA relies upon eigenvector decomposition of the covariance or correlation matrix to capture the major tendencies of process variables. Let a mean centered dataset X be represented as $X = \{x_1, x_2, x_3, ..., x_m\}^T$ with I rows and N columns. The covariance matrix of X, is defined as:

$$Cov(X) = \frac{X^T X}{I-1}. \qquad \text{(Eq 2-13)}$$

PCA linearly decomposes the data matrix X as the sum of scores, t, loadings, p and a residual matrix e in the following way (Wise and Gallagher, 1996):

$$X = t_1 p_1^T + t_2 p_2^T + ... + t_k p_k^T + ... + t_K p_K^T + e. \qquad \text{(Eq 2-14)}$$

Here, K is the number of principal components that a user wants to retain. The value of K is at most equal to the dimension of X:

$$K \le \min(I, N). \qquad \text{(Eq 2-15)}$$

The scores vector, t contains information on how the samples relate to each other, while the loadings vector, p contains information regarding the correlation among the variables. The p vectors are normally computed from the eigenvectors of the covariance matrix:

$$\text{Cov}(X) p_k = \lambda_k p_k, \qquad \text{(Eq 2-16)}$$

where λ_k is the eigenvalue associated with eigenvector p_k. The t, p are then sorted in descending order with respect to λ_k. The λ_k are a measure of the amount of variance described by the k^{th} principal components, or can be thought of the information being retained from the matrix X by p_k. In most cases, only a few rows of p_k that capture enough variance of X need to be retained for process monitoring (Wise and Gallagher, 1996).

2.5.1 PCA Associated Monitoring Statistics

Fault detection using PCA or its variants is achieved through monitoring of Squared Prediction Error (SPE) and/or Hotelling's T^2 statistic. The SPE measures the variation of a sample $\boldsymbol{x}_i$ from the PCA model, *i.e.*, lack-of-fit (Jackson and Mudholkar, 1979):

$$SPE_i = e_i e_i^T = x_i(\mathbf{1} - p_k p_k^T)x_i^T. \qquad \text{(Eq 2-17)}$$

The process is considered normal if $SPE_i < Q_{1-\alpha}$, where $Q_{1-\alpha}$ denotes the upper control limit for confidence level, $1-\alpha$ (Jackson and Mudholkar, 1979):

$$Q_{1-\alpha} = \theta_1\left[1 + \frac{c_\alpha\sqrt{2\theta_2 h_0^2}}{\theta_1} + \frac{\theta_2 h_0(h_0-1)}{\theta_1^2}\right]^{1/h_0} \qquad \text{(Eq 2-18)}$$

where $\theta_{w=}\sum_{j=k+1}^{n}\lambda_j^w$ with $w \in [1, K]$, $h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2}$ and c_α is the standard normal deviation corresponding to the upper $(1-\alpha)$ percentile.

The T^2 statistic measures the variation of the sample within the PCA model:

$$T_i^2 = t_i\lambda^{-1}t_i^T = x_i p\lambda^{-1}p^T x_i^T \qquad \text{(Eq 2-19)}$$

where λ^{-1} is the diagonal matrix containing the inverse of the eigenvalues associated with K eigenvectors retained in the PCA model (Wise *et al.,* 1990). An upper control limit $T^2_{1-\alpha}$ similar to $Q_{1-\alpha}$ can also be derived for the T^2 statistic (Jackson, 1991):

$$T^2_{1-\alpha} = \frac{K(I-1)}{I-K} F(K, I-K, \alpha) \qquad \text{(Eq 2-20)}$$

where $F(K, I-K, \alpha)$ is a F distribution, α is the probability point, K and I-K are the numerator and denominator used in evaluating the degree of freedom.

The SPE and T^2 monitoring statistics are complementary in nature, since the SPE measures the lack-of-fit while the T^2 statistic measures the variation of a sample within the model. Nevertheless, there have been suggestions that the use of a single index, rather than two different indexes, is preferrable in practice (Qin, 2003). A combined index for fault detection that combines SPE and T^2 statistics have been reported by Raich and Çinar (1996) and Yue and Qin, (2001). Raich and Çinar (1996) proposed a combined index called *combined discriminant similarity index*, Φ_i, given as:

$$\Phi_i = \beta\left(\frac{SPE_i}{Q_{1-\alpha}}\right) + (1-\beta)\left(\frac{T_i^2}{T^2_{1-\alpha}}\right) \qquad \text{(Eq 2-21)}$$

where $\beta \in [0,1]$. They further suggested that $\Phi_i < 1$ for a process to be considered normal. As an alternative, Chen *et al.* (2004) suggested synthesizing SPE and T^2 statistics on a new bi-variate plot with their joint probability density function estimated through kernel density estimation. The joint projection technique was found to be more sensitive in detecting process fault and is able to reduce the number of monitoring charts used.

There exist some variants of PCA methods termed Multiway-PCA and Dynamic-PCA (described in Appendix B). Previous literature on monitoring based on MPCA/DPCA for transient operations normally require the batch-length to be equal in order to rescale the online measurements at each time instant to reduce the occurrence of false positives, e.g.: Birol *et al.* (2002), Lee *et al.* (2004b), Chen and Liu (2002), Nomikos and MacGregor (1995), Rännar *et al.* (1998), etc. Monitoring limits which are generated based on fixed time approach might give reasonable performance if the underlying events are highly synchronized, e.g.: activation of fed-batch at fixed time-point, all growth of microbiology to be consistent throughout, fixed duration of batch operation, etc. However, run-to-run variations are often significant in transient operations due to process environmental or human factors. For example, a low ambient temperature might cause a longer duration of heating for a process reboiler, different run lengths might occur due to unpredictable biomass characteristics, operation time-lag induced by plant operators, etc. Such varying behavior cannot be adequately modeled by MPCA and DPCA techniques. To overcome these shortcomings, multiple models are needed for a more flexible monitoring of transient operations.

2.5.2 Multi-model Approach for Process Monitoring

The use of multiple local models has been a popular approach in system identification (Böling *et al.*, 2004), advanced control (Palma and Magni, 2004), and monitoring (Bhagwat *et al.*, 2003b). In Chapter 4 of this thesis, a *divide-and-conquer* strategy is used to overcome the Type-I and Type-II errors outlined in the previous section by using multiple PCA models to monitor transient processes. Numerous benefits have been reported when such strategies are adopted to monitor multiple process modes. In the area of PCA, the use of distinct models for pattern recognition was first introduced by Wold (1976). The author suggested performing pattern

recognition based on modeling each distinct class by separate PCA models. While disjoint PCA models might yield greater accuracy during pattern recognition, their application to transient operations is prone to Type-I errors as these processes often follow a smoothly varying trajectory rather than abrupt changing from one state to another. Hwang and Han (1999) developed a method to monitor processes with multiple operating modes. Their method is based on clustering using a self-organizing map and a super-PCA model to capture the average covariance structure of the different modes. However, their approach is valid for modes that do not show strong nonlinearities, and the modes must be located near to one another in the PC subspace for the method to work. Lu and Gao (2005) showed that dividing a batch process into stages of different process characteristics could improve the quality of prediction of an injection molding process. Their method is based on a stage-based PLS modeling approach. The above methods used limited number of models for modeling processes that are not stationary. As an alternative, Rännar *et al.* (1998) built separate PCA models for each time slice of a batch data. They collected samples at each time interval across different batches of training data, and constructed a PCA model for each time interval. Online monitoring is then based on the PCA model created for a particular time instant. Such method of creating multiple PCA models requires every batch to be monitored have equal batch length, which is often not practical as run-to-run deviations can be quite common in practice. In summary, the existing PCA approaches with multiple models are limited to equal run-length, and inadequate handling of discontinuities, and thus prone to false positives. To overcome these shortcomings, a novel monitoring technique based on overlapping PCA models is proposed in Chapter 4. As opposed to previous approaches, the proposed method is *robust to varying run*

length, *gives good monitoring resolution*, and *sensitive to new operating regions* (*i.e.,* able to detect novel faults).

2.6 Fault Isolation through Principal Components Analysis

In Chapter 5 of this thesis, a pattern recognition approach for fault isolation based on PCA augmented kernel density estimator (KDE) is developed. In literature, fault isolation in MSPC often relies on the monotonic increase/decrease of some function with respect to an appropriate measure. Two PCA-based approaches for generating such functions are based on angle discrimination, and statistical discrimination approaches.

2.6.1 Fault Isolation based on Angle Discriminant

PCA angle discriminant relies on comparing the angles of the latent variables of a faulty dataset with respect to another PCA model. Classification of fault classes is quantified through a similarity measurement, S_{PCA}, which quantifies the resemblance in shape between two PCA models in the latent subspace (See Appendix C). S_{PCA} measures the similarity between two PCA models based on the angles between the vectors of the first k PCs (Krzanowski, 1979). A modified form of S_{PCA} is given by Singhal and Seborg (2002) through weighting each principal component by the square root of its corresponding eigenvalue. A small value of S_{PCA} normally indicates low similarity between two models whilst large value signifies high similarity. Applications of S_{PCA} for fault diagnosis can be found in Raich and Çinar (1997), Singhal and Seborg (2002). However, one major drawback with angle-based discriminant is the difficulty in selecting window width from any given dataset. A snapshot collected from real-time is often an incomplete signal of a fault class. The non-availability of extensive fault data usually leads to poor recognition rate even

though the right class of fault is selected for comparison. As a solution, statistical discriminants which do not require the specification of snapshot can be used.

2.6.2 Fault Isolation based on Statistical Discriminant

Statistical discriminants based on Q statistic and T^2 can be used to characterize the similarity between various dataset. If a comparison of similarity is needed between two datasets, denoted *S* and *H* here respectively, a PCA model can be constructed based on dataset of *H*. *S* can be classified as belonging to class *H* if the majority of *SPE* and T^2 statistic generated from *S* does not violate the upper control limit generated from PCA model of class *H*. When a dataset is compared against *J* fault classes, *J* number of PCA models is usually generated with their frequencies of in-control status (*SPE* and T^2) quantified. Such method of fault discrimination has been termed as fault reconstruction by Dunia and Qin (1998). If the PCA model shows good representation of the fault classes, T^2 statistic can be used to discriminate the fault classes. If the important variations for discriminating faults are contained in the residual space, fault model that minimizes Q statistic can be identified. Alternately, if the variations are seen in both the score and residual space, then the fault candidate F^c that minimizes the combined discriminant among W fault classes, $F_1,...,F_w,...,F_W$, can be chosen (Yue and Qin, 2001):

$$F^c = \arg\min_w b[T^2 / T^2_{1-\alpha}]_w + (1-b)[SPE / Q_{1-\alpha}]_w \qquad \text{(Eq 2-22)}$$

where *b* is a weighting factor between 0 and 1. If the majority of fault samples identified shows:

$$[T^2 / T^2_{1-\alpha}]_w << 1 \text{ and } [SPE / Q_{1-\alpha}]_w << 1 \qquad \text{(Eq 2-23)}$$

the observation is deemed a good match to the fault class *w* identified (Chiang *et al.*, 2001). Fault discrimination based on *SPE* and T^2 has been reported to be effective in

the domain of continuous processes (Qin, 2003; Dunia & Qin, 1998; Chiang *et al.*, 2001), as the training data is usually composed of single mode of operation, where any deviation from the targeted mode is usually considered faulty. However, when a process undergoes transition, the magnitude of most variables vary considerably. Process transitions contravene the necessary conditions for application of conventional methods for statistical discriminant, *i.e.*, Hotelling's T^2 statistic. Such deviations often cause significant Type-I and Type-II errors. As an illustrative example, consider the trajectory produced from a time-varying fermentation process and represented in the PC subspace as shown in Figure 2-4. Hotelling's T^2 statistic (Eq 2-12) quantifies the deviation from PCA models through a F-distribution (shifted normal distribution) of the training data, which corresponds to an ellipse in the PC subspace. As can be seen from Figure 2-4, the normal operating data is not fully bounded with Hotelling's T^2. The use of $\alpha = 1\%$ (99% confidence, i.e, $T^2_{0.99}$) still dictates a large portion of the normal operating data as faulty (false positives). In this example, the rate of false positives is 9.06% contributed from regions *t*=[1 49], [84 87], [94 100], [790 806]. Apart from the high rate of false positives, Hotelling's T^2 is also prone to false negatives during transient operations. As shown in Figure 2-4, the statistic misclassifies operating spaces not belonging to the training data, which might be abnormal region. In Chapter 5 of this thesis, the class separability of temporal data in the score space is explored based on Kernel Density Estimator to overcome the shortcomings of Hotelling's T^2 statistic.

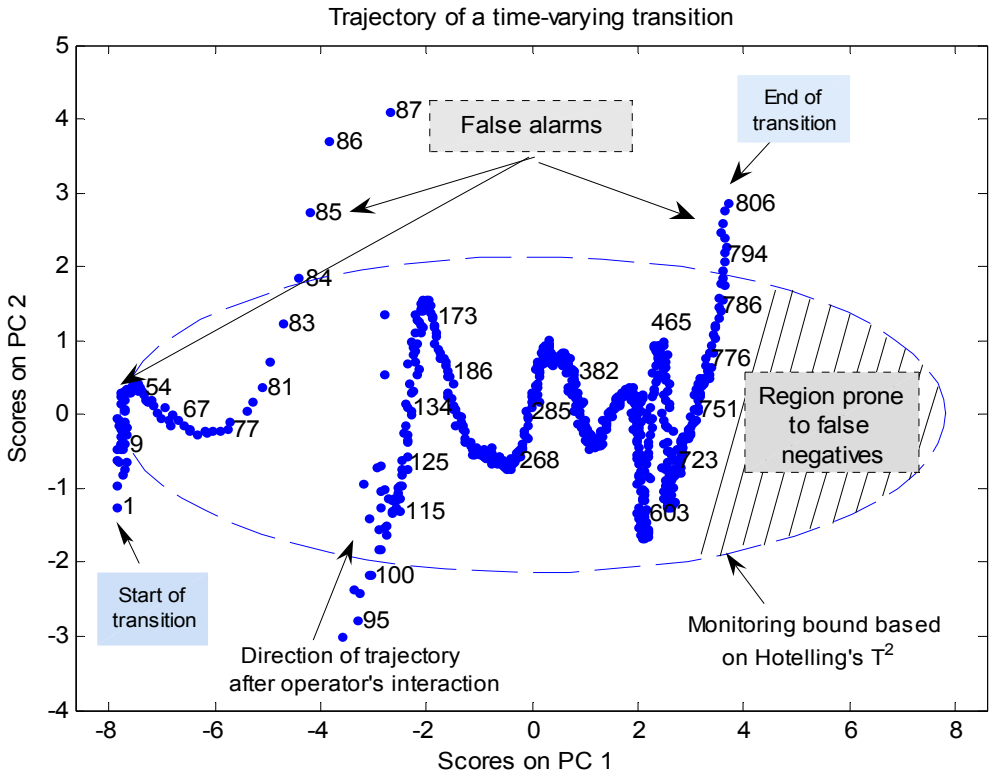


Figure 2-4: Limitations of Hotelling's T^2 statistic

2.6.3 Nonparametric Bounds for Pattern Recognition with KDE

Kernel Density Estimator (KDE) is a density estimation technique which creates bounds based on the training data without the imposition of any priori assumption of density distribution of the training data (distribution of training data during transition is often not known to control engineers and difficult to predict). The monitoring bounds constructed can thus describe the behavior of process transitions more accurately.

KDE is a data-driven technique to estimate the density contour of a function. The main motivation for applying nonparametric approach to monitor transient operations is straightforward, i.e., when the dataset is showing no discernible functional form, then one would want to let the data decide which function fits them best. The basic algorithm of nonparametric density estimation was first introduced by Fix and Hodges (1951) when they addressed the problem of statistical discrimination

when the parametric form of the sampling density is not known. Several improved algorithms and alternative theoretical modes of analysis were later introduced by Rosenblatt (1956), Watson and Leadbetter (1963), Epanechnikov (1969), and Wahba (1971). The practical applications of nonparametric density estimation can be found in Scott (1979) and Silverman (1978).

A non-parametric model gives better indication of features such as skewness and multimodality in a dataset which is usually unavailable from parametric model. Consider a *N*-variate data *X* with *I* samples given by $X = (x_1, ..., x_i, ..., x_I)^T$, and a generic vector $x_i \in \Re^N$ having representation $x_i = (x_{i1}, ..., x_{in}, ..., x_{iN})$. The *N*-dimensional kernel density estimator for *X*, $\hat{f}$, can be expressed as (Wand and Jones, 1994):

$$\hat{f}(x,h) = I^{-1} \sum_{i=1}^{I} K_h(x - x_i) \qquad \text{(Eq 2-24)}$$

where *h* is a symmetric positive definite $N \times N$ matrix called the *bandwidth matrix*,

$$K_h(x) = |h|^{-1/2} K(h^{-1/2} x), \qquad \text{(Eq 2-25)}$$

and *K* is a *N*-variate kernel function satisfying

$$\int K(x)dx = 1. \qquad \text{(Eq 2-26)}$$

The bandwidth matrix is an important parameter in determining the final form of the density model and is described further in Appendix D. KDE methods have been popular in process monitoring though its application for disturbance identification has not been reported. Martin and Morris (1996) used KDE to generate confidence bounds for analysis of process data produced from a batch polymerization reactor. They showed that bounds generated from non-parametric approach was able to provide better accuracy in detecting faulty batches when visualized on the first two principal

components. The application of kernel density estimation to monitor a steady-state section of a glass melter is presented by Chen *et al.* (2000). They evaluated three bandwidth selector techniques, namely, MISE, adaptive MISE (AMISE), and biased asymptotic MISE (BAMISE) techniques. Their study showed that the BAMISE outperformed the other two methods in estimating data density for samples with small variation. Goulding *et al.* (2000) highlighted the issues of control under process uncertainty through KDE approaches. They used KDE to augment and support the functionality of closed-loop control for uncertainties caused by sensor errors. In Doymaz *et al.* (2001) nonlinear-PCA augmented KDE was used for real-time process monitoring. They executed their strategy in three steps: 1) noise suppression and outliers rejection through wavelet and moving median filter, 2) dimensional reduction through wavelet, and 3) pair-wise comparison of scores extracted from principal components through KDE. An approach for joint analysis of the T^2 and *SPE* statistics for statistical process monitoring was introduced by Chen *et al.* (2004). The authors used KDE on the scatter diagram produced from T^2 and *SPE* statistics in order to increase the sensitivity of both statistics. Lopes and Menezes (2004) used a non-linear principal components analysis for monitoring fermentation processes. They constructed a five layer auto-associative neural network for non-linear principal components extraction before KDE was applied to monitor the extracted latent variables. The application of KDE to monitor wastewater treatment process was described in Lee *et al.* (2004a) where independent component analysis was used to recover the source signals form a process data before KDE was applied to detect process faults.

As described earlier, most KDE-based approaches for monitoring were usually conducted through inspection of the density contours produced on the 2D subspace of

the latent components. This usually involves looking at all possible pairs of latent components obtained. In Chapter 5 of this thesis, a method is proposed to combine the bi-variate KDE into a single, unified index by aggregating information from various combinations of the latent components obtained. Performance of the proposed index to disturbance detection and identification is also studied.

2.7 Collaborative Decision Support with Multi-Agent System

A software agent-based framework is proposed in Chapter 6 to integrate various FDI methodologies into a coherent, unified system to support decision making during process operations. The term *agent* is defined as a computer system that is situated in some environment, and is capable of performing autonomous actions in that environment in order to meet its design objectives (Wooldridge and Jennings, 1995). A software agent can thus be viewed as an identifiable computational entity that automates some aspect of a task or decision making to benefit humans. Agent-based computing is regarded as the next significant breakthrough in software development with capabilities to perform autonomous actions in open, dynamically changing environment. It offers opportunities to solve a complex problem collaboratively using heterogeneous methods across multiple platforms. An agent is generally characterized by its underlying knowledge/methods, and is allowed to interact with other agents using a common agent communication language.

Agent-based computing has been used in various fields since its introduction in the early 90s. Some popular areas in which agent-based computing have been used include supply-chain analysis (García-Flores *et al.*, 2000; Julka *et al.*, 2002a,b), legacy systems integration (Allsopp *et al.,* 2002; Sheremetov *et al.*, 2004; Tabuada and Pappas, 2005), and information fusion (Tso *et a.,* 2000).

In the area of monitoring and fault diagnosis, Mangina *et al.* (2001) used multi-agent approach to perform condition monitoring of industrial gas turbine startup sequences. Their agents used knowledge-based method to recognize the different phases of turbine operations. Abnormal operations of turbine were detected by placing appropriate linear threshold across state variables. The final diagnostic results were produced after validating the consistency with other sensors (implemented as Sensor collaborative agents and Meta-knowledge Reasoning Agent). Cho *et al.* (2003) developed an agent-based method to monitor network configuration. They incorporated their agents with different sets of knowledge-based reasoning systems to diagnose and recover known network faults. In Maturana *et al.* (2004), the multi-agent architecture was proposed to integrate various automation systems to control a chilled water system that is used frequently in US-navy vessels. Their diagnostic component includes a suite of data acquisition, signal processing, diagnostic, and prognostic algorithms. Under abnormal event, each agent can interrogate the diagnostic component of other agents to validate a fault hypothesis. In Ng and Srinivasan (2004b), the feasibility of combining different FDI methods based on multi-agent architecture was studied. They showed that various FDI methods can be integrated into a cohesive, and interacting entity through the introduction of an agent wrapper (see Chapter 6 for details).

2.7.1 Needs for Collaborative & Distributed Agents

It is worth noting that each FDI method has its corresponding strengths and shortcomings that are process dependent. A method that works well under one circumstance might not work well under another when different features of the process come to the fore. A comparison of the strengths and shortcomings of these methods is shown in Table 2-1. As can be seen from the table, no single FDI method has the

capability to address the numerous facets of transient processes. Collaboration among heterogeneous methods is therefore needed to bring forth the benefits of each method to improve monitoring resolution and robustness of a FDI system. The rationale of such an approach is based on the precept that the strengths of various methods can be brought together to bear on the problem and the drawbacks of an individual method can be overcome through *collaboration*.

It has already been shown in the pattern recognition literature that a judicious combination of classifiers generally outperforms a single one (Al-Ghoneim and Kumar, 1998; Rahman and Fairhurst, 1999; Lin *et al.*, 2003; McArthur *et al.*, 2004). The main reason for combination of classifiers is based on the axiom that different types of classifiers can often compliment the shortcomings of others and hence improve classification performance through combined prediction. When diagnosing faults in complex processes, designing a perfect classifier for all possible scenarios can be difficult, and combining different fault diagnostic methods can be a good alternative by extracting different features from heterogeneous diagnostic classifiers. Combining multiple classifiers can also simplify the tedious process of parameters tuning. In most pattern recognition algorithms, the final model is affected by various parameters that need to be tuned offline. The objective of offline training is thus determination of the optimal parameters that minimizes prediction errors, e.g.: select optimal combination of variables, historical records in time-lag model, numerous thresholds, etc. Each combination could lead to different performance when the model is used online, and thus exhibit different prediction capabilities for different fault classes. One way to mitigate this problem is to train a group of classifiers, where each classifier utilizes different parameters to tackle a specific class of interest, and produces predictions based on the combination of all results (Lim and Harrison, 2003).

The response time from a collaborative FDI system could constitute a bottleneck for its real-time applicability. Since most monitoring and fault diagnosis algorithms are computationally complex, a tremendous amount of computational resources would be required for monitoring and fault diagnosis, whereas the FDI results are often needed in real-time. There exist two main factors that contribute to the computational complexity in any FDI program: complexity due to algorithms, and complexity due to data structure. Algorithm complexity is caused by user commands that implement a high level of search activities at each iteration cycle. Some examples of such algorithms are dynamic time warping, dynamic locus analysis, and expert systems where a high number of CPU flops are needed. The data structure complexity is associated with the volume of data that need to be processed. Some simple routines can turn out to be extremely time consuming when the volume of data is high, e.g.: when iterating over a multivariate database of few hundred Gigabytes, the processing time for simple multivariate techniques such as principal components analysis (Wise and Gallagher, 1996) and partial least squares (Wold *et al.*, 1989) approach could turn out to be significant. Consequently, the resulting collaborative FDI approach could easily overwhelm the load of a single processor. Such limitation is resolved in Chapter 6 by integrating parallel computing method into the multi-agent framework by distributing the FDI methods (agents) across a networked computing cluster.

Table 2-1: Strengths and shortcomings of different FDI methods

Methods	**QTA**	**Expert Systems**	**PCA/PLS**	**Kalman-filters**	**DTW/ DLA**	**Neural Networks**	**SOM**
Multivariate Analysis	×	×	√	×	√	√	√
Speed of Detection	√	√	√	√	√	√	√
Fault Isolation	√	√	√	√	×	√	√
Novel Fault Detection	√	×	√	×	√	√	√
Explanation facility	√	√	×	×	×	×	×
Visualization of results	×	×	√	×	×	×	√
Recovery automation	×	√	×	×	×	×	×
Ease of development	√	×	√	×	√	√	√
Adaptation & Robustness	√	×	√	×	√	√	√

2.8 Decision Fusion Strategies for Conflicts Resolution

When multiple FDI classifiers are integrated, there is a need to synchronize the results generated from each FDI classifier with the conflicts among them resolved. The results synchronization in the proposed multi-agent framework through decision fusion is described in Chapter 7 of this thesis. Decision fusion is a subdivision of data fusion, which is defined as a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from single and multiple sources (DSTO, 1994). Decision fusion has been used in applications ranging from earth resource monitoring, weather forecasting, vehicle traffic control to military target classification and tracking, etc. The review paper from Clemen (1989) provides a good compilation of early work in the area of decision fusion, where the conclusions obtained from the decision fusion community were virtually unanimous: most authors reported dramatic performance improvements by combining multiple forecasters or classifiers. Hall (1992) distinguished the level of decision fusion into three categories, namely data-level fusion, feature-level fusion, and identity-fusion. Data-level fusion involves the process of merging data from coextensive sensors prior to analysis to eliminate sensing errors. Feature-level fusion combines different information from different sensors for identity declaration, while identity-level fusion performs class declaration by combining class specific predictions from different classifiers. This thesis focuses on decision fusion at the identity declaration level (Figure 2-5). The decision fusion process involves combination of predictions from different classifiers to achieve a joint declaration on data identity (classes) by merging predictions from all classifiers.

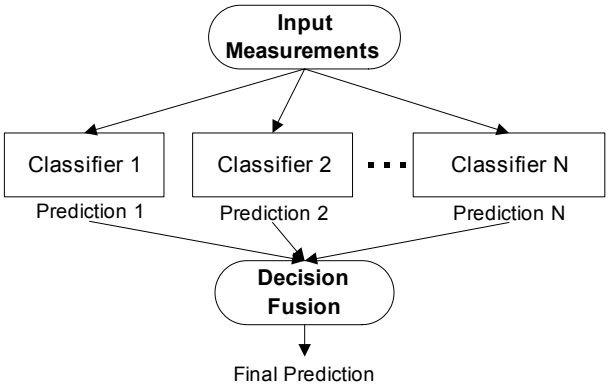


Figure 2-5: Schematic diagram of a decision fusion process involving N classifiers

In this thesis, existing approaches for decision fusion are classified into utility-based and evidence-based methods. *Utility-based* methods provide the simplest way to fuse decisions. These methods do not utilize prior knowledge or evidence from previous predictions, but are based on some aggregating techniques which evaluate the combined utility functions generated from each classifier. Methods based on utility techniques include simple average, voting techniques, and their variants. Early studies in the area of economic variables estimation (Ringuest and Tang, 1987) showed that taking simple average from all forecasters gave higher accuracy in predictions as the error distributions from forecasters normally deviates evenly on the positive and negative side. A more popular form of utility-based approach is voting. There exist various variants of voting schemes, e.g.: plurality voting, approval voting, cumulative voting, borda count, etc. Each of these voting schemes differs in the way that the utility function is aggregated:

1. *Plurality voting*: Plurality voting is a winner-take-all system that is currently used in most countries for election. During decision fusion, each classifier assigns votes for their desired candidates and the candidate that receives the maximum votes is declared the winner. The plurality voting technique is also

referred to as majority voting when a candidate requires more than 50% of the votes to be declared the winner.

2. *Approval voting* (Brams and Fishburn, 1983): Each classifier votes for as many candidates as needed. Each candidate can receive at most one vote from one classifier, and the candidate receiving the greatest number of votes wins.
3. *Borda count* (Saari, 1994): When the preference of ordering of each classifier are taken into account, the most preferred candidate receives *n* votes, while the second most preferred one received *n-1* votes, and finally *1* for the least favored candidate. The candidate with the most number of votes is selected as winner.

Applications of voting techniques to combine classifiers for handwritten numerals recognition can be found in Lam and Suen (1997), and Lin *et al.* (2003).

In contrast to utility-based techniques, *evidence-based* approaches use a priori information regarding the previous performance of each classifier to combine decisions. Two main approaches that form the backbone of many evidence-based approaches in the pattern recognition literature are the Bayesian and the Dempster-Shafer methods. Bayesian technique is a popular method in evidence gathering and uncertainty reasoning by calculating the posteriori probability of an event. In Zheng *et al.* (2005), Bayesian-based fusion was used to integrate various image processing models for diagnosing diseases in endoscopic images. Results from different image analysis methods such as color segmentation, texture segmentation, and lumen detection, are combined to improve the accuracy of cancer detection. Rahman and Fairhurst (1999) proposed a decision combination strategy using a priori information sources for machine printed character recognition. They suggested constructing a knowledge base that incorporates information of a dataset required by the regular

Bayesian approach for combining classifiers. In Foggia *et al.* (1999), threshold-based rejection criteria was proposed for Bayesian combination by using information regarding reliability of a classifier. They showed that introduction of rejection options for Bayesian combination reduces the rate of misclassification. In McArthur *et al.* (2004), three FDI methods have been combined based on Bayesian approach to diagnose faults within a transformer. They analyzed faults using *k*-means clustering, backpropagation neural-network, and user written rules, and showed that collaboration of FDI methods improve system performance.

The Dempster-Shafer theory, also referred to as theory of belief functions, is a generalization of the Bayesian theory of subjective probabilities (Shafer, 1990). The theory has been used to merge various types of classifiers. Instead of utilizing probabilities to address each question of interest as with Bayeisan theory, the Dempster-Shafer approach uses degree of belief collected from independent items of evidence (subjective probabilities) to merge two pieces of information. In Basir and Yuan (2005), Dempster-Shafer approach was used to locate faults in induction motors by combining time-domain, frequency-domain and statistical signal processing methods. Similar approaches were used in Yang and Kim (2006), where time-based and frequency-based features were first extracted from a motor by using two classifiers - vibration classifier and current classifier, before the final outputs were combined using Dempster-Shafer theory to generate final predictions of the motor faults. In Benouhiba and Nigro (2006), Dempster-Shafer method was used to resolve uncertainty in a multi agent-based cooperative system. Their proposed multi-agent system was able to extract rules to describe an environment from a rule database. Each rule was associated with a confidence value and the agents interacted with each other until a consensus was reached among all agents regarding the environment. Application of

Dempster-Shafer method to combine speech recognition classifiers is shown in Chan *et al.* (2006). The authors combined information from acoustic speech information with facial myoelectric signals and showed that decision fusion improved the robustness of speech recognizer under noisy conditions. In Chapter 7 of this thesis, three decision fusion methods (Voting, Bayesian-combination, Dempster-Shafer combination) are compared.

Nomenclature

Indices

i	index used for process time representation (row of a given matrix)
j	index used for neurons within a self-organizing map model
k	index used for representing principal components
n	index used for representing process variable (column of a given matrix)
w	index used for representing different fault classes

Parameters

$\mathbf{1}$	an identity matrix
I	number of samples for a given time-series data X
J	number of neurons within a SOM model
K	number of principal components used for process modeling
W	total number of fault classes in a fault database
β	weighting function used to combine *SPE* and T^2 statistic

Variables

b_i	best matching unit (BMU) on SOM for a given sample
c_α	Standard normal deviation for upper $(1-\alpha)$ percentile in computing $Q_{1-\alpha}$
Cov(X)	covariant matrix of a matrix X

e_i	residuals observed for x_i with *PCA* modeling
F	an F-distribution
F^c	fault candidate that matches the signals from online measurement x_i
$h_{bi}(t)$	neighborhood for an identified BMU, b_i, from sample x_i during iteration t
m_j	a neuron on self-organizing map, $m_j = \{m_1,...,m_n,...,m_N\}$
M^{SOM}	a self –organizing map model $M^{SOM} = \{m_1,...,m_j,...,m_J\}$
p_k	k^{th} principal component
$Q_{1-\alpha}$	upper control limit for SPE statistic
r	the radius of the neighborhood function
SPE_i	squared prediction error associated with PCA decomposition
$T^2_{1-\alpha}$	upper control limit for T^2 statistic
t_i	scores for x_i from principal components projection
T_i^2	Hotelling's T^2 statistic at time i^{th}
X	a multivariate time series data
x_i	i^{th} samples from a multivariate time series data X
$\alpha(t)$	learning rate at iteration t during SOM training
λ_i	diagonal matrix containing inverse of the eigenvalues for eigenvectors of cov(X)
$\sigma(t)$	width of the neighborhood function used for SOM training at iteration t
Φ_i	combined discriminant similarity index observed for sample x_i

Chapter 3 Multivariate Temporal Data Analysis using Self-organizing Maps – Visual Exploration of Multi-State Operations

3.1 Introduction

Previous works on SOM have largely focused on exploiting the clustering capability of SOM for grouping multivariate data. In this chapter, the SOM is extended to visualize in real-time the multivariate samples originating from multi-state processes. A SOM-based methodology was developed to depict multi-state process operations. In the proposed representation, data from different process states (steady state and transient) demonstrate different characteristics in the SOM space. Steady states form clusters of adjacent BMUs while transient operation is reflected as a trajectory. Differences between two states can be observed easily based on the location and evolution of the BMUs.

A SOM has to be suitably trained to represent various process states. For robust visualization, the training of the SOM is performed using all available historical operations data – including those from steady state and transient operations during both normal and abnormal operations (Ng and Srinivasan, 2004). Using all process data for training enables the SOM to represent a wide range of operating conditions on the map. During training, the neurons on the SOM will orient themselves and evolve into a process map representing all the operating conditions in the training data while preserving the topology of the measurement space. The trained SOM model can then be used to visualize the process state in real-time.

3.2 Visualization of Process States

For visualizing process operations, data from online sensors, x_i , are first projected on the trained SOM and the BMU of x_i , m_{b_i} , identified. The location of m_{b_i} on the SOM indicates the current state of the process. As described in Srinivasan *et al.* (2004), process operation states can be classified into modes and transitions as manifested through the values and behavior of the measured variables. A mode corresponds to the continuous operation of the plant and corresponds to a fixed flowsheet configuration, i.e. no equipment is brought online or taken offline. During a mode, the process operates in a quasi-steady state and measurements should generally vary within a narrow range, although some variables may oscillate due to noise, instrumentation faults, or improperly tuned controllers. A transition corresponds to discontinuities in the plant operation, such as change of setpoint, opening valve, change of equipment configuration, turning on or idling equipment, etc, usually induced by operator actions. During a transition, at least one of its constituent variables would show a significant change.

Process modes and transitions display different characteristics on the SOM space. When a process is in a mode, all its variables have near-constant values. Therefore, online measurements from such a state should be projected on the same BMU. Noise and minor variations in process operation could result in projection of the online measurement to different BMUs, however these would be neighboring neurons because of the topology preserving feature of SOM training. Process modes can thus be identified when a high frequency of BMUs are found within a small neighborhood in the map. Process variables have significantly different values between different modes. So, different modes can be distinguished on the SOM based on difference in the BMUs.

In contrast, process transitions are characterized by large changes in plant operating conditions. Such evolution of the variable values during the transition would cause the BMUs to traverse over a wide region in the SOM space. The transition can be visualized by connecting the successive BMUs and displaying the trajectory of process evolution. During transition operations, continuous variables and discrete variables have different effects over the trajectory. Continuous variables usually evolve from their original values to new target values over some period of time. For example, a heating operation will lead to increase in temperature from some initial value to a new value over a period of time. Such evolution of continuous variables would cause the BMU to advance through adjacent neurons, resulting in a smooth trajectory on the SOM space. However, discrete variables correspond to abrupt changes in plant operations such as opening or closing of valves, activating or deactivating of equipments, etc. Such changes can cause abrupt jumps to a BMU that is a significant distance apart, and thus would be exhibited as a discontinuous evolution in the trajectory. The transition trajectory on the SOM represents time in an implicit manner as it is produced based only on magnitude of the variables. An increase or decrease in the rate of change between two instances of the same transition would not be evident in the sequence of BMUs in the trajectory. This makes the representation robust to run length variations and is exploited in Section 3.3 for process monitoring.

Different transitions exhibit different trajectories in the SOM since they would start and end at different operating points; also the sequence of operations executed during different transitions would differ, hence the values of the intervening conditions would be different. These differences manifest themselves as differences in the trajectory on the SOM space. As a corollary, abnormal operations including wrong sequence of SOP execution, wrong procedures, wrong timing, and hardware failures

would also manifest themselves differently in the SOM space, leading to BMUs in abnormal neighborhoods. These can then be detected from the SOM visualization.

3.3 Neuronal clusters

Next, consider the relationship between the process states and their depiction on the SOM map. Typically, the number of neurons is selected to be much larger than the number of states that the process would operate in. So the process states would not map to unique neurons. The j^{th} neuron in the SOM corresponds to the process operating conditions whose mean is specified by m_j. The same neuron would continue to be a BMU for all x_i near this operating condition. A larger SOM with more neurons would offer finer resolution of operating conditions and is required to precisely visualize transition conditions and progression. However in large SOMs, even small changes in the operating condition would lead to different neurons (although in the same neighborhood) becoming the BMU, i.e., the "noise" absorbed by each neuron is low. To meet the conflicting requirements of finer resolution and better noise absorbance, a second layer of abstraction can be defined by grouping neurons into neuronal clusters (see Figure 3-1b).

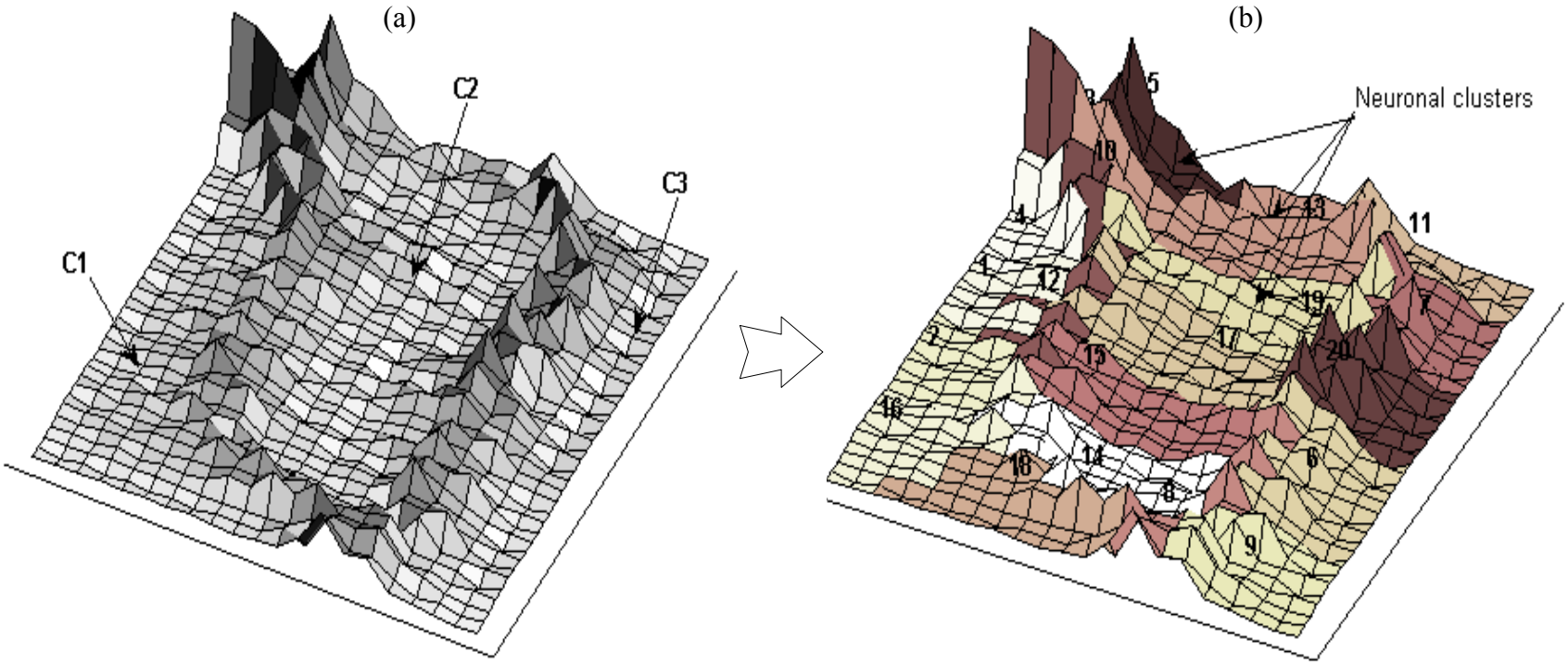


Figure 3-1: Representation of process operational data using (a) single neuron per state, and (b) neuronal clusters

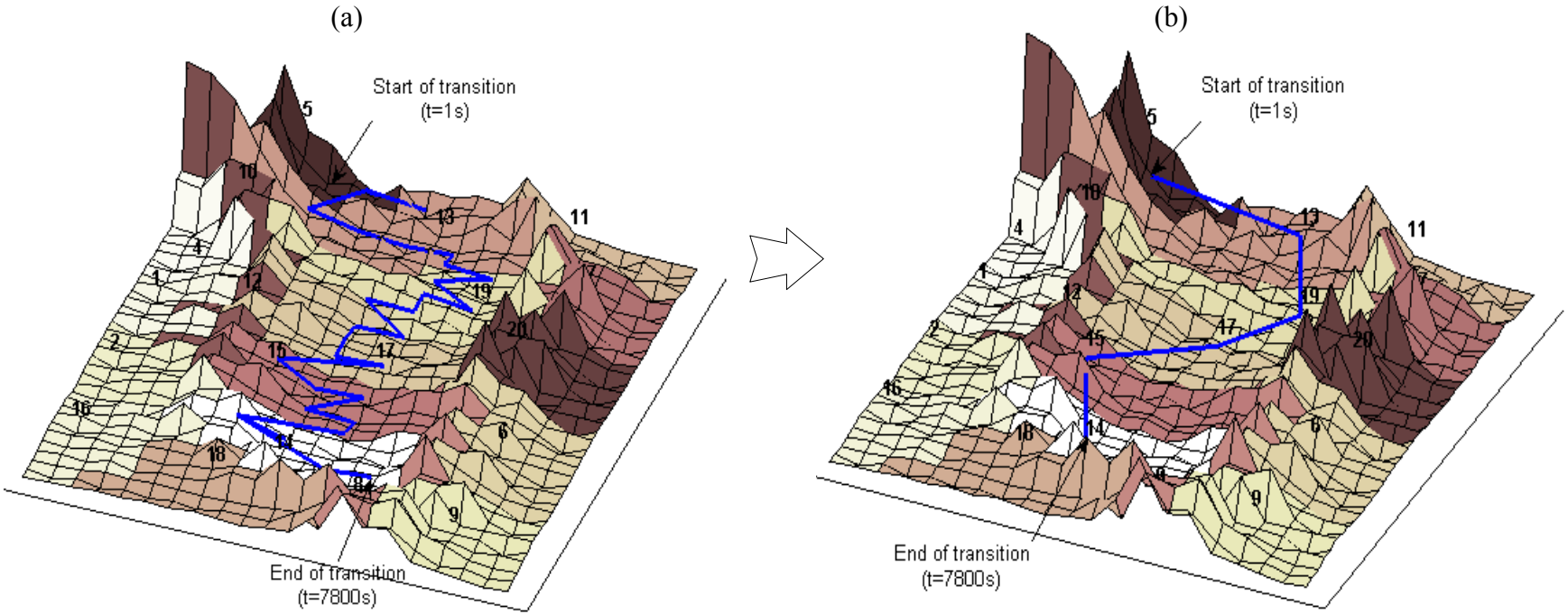


Figure 3-2: Cluster-based representation of process transition

A neuronal cluster is defined as a set of contiguous neurons in the SOM map with high similarity in m_j. The neuronal cluster exploits the topology preserving feature of SOM training to provide a coarser representation of operating conditions and process states. Through this abstraction, as shown in Sections 3.3 and 3.4, different modes would map to unique neuronal clusters. The state can then be identified in real-time based on the hit. A neuronal cluster is said to have a hit if any of its constituent neurons has a hit. This can be used to abstract the trajectory as shown in Figure 3-2, where the hits are visualized on the cluster centroid instead of the neurons, thus depicting the transition at a lower level of resolution.

Neuronal clusters are defined by clustering the neurons in the trained SOM based on their reference vectors. Let all the neurons in M^{SOM} be grouped into K neuronal clusters $\{S_1, S_2, ...S_k, ...S_K\}$. The assignment of neuron j to cluster k is specified by a membership function u_{jk}:

$$u_{jk} = \begin{cases} 1 & \text{if neuron } j \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Eq 3-1)}$$

Any clustering technique can be used to specify u_{jk}. The k-means clustering algorithm has been used here, which identifies the K clusters so as to minimize the total squared distance, ε_p (Seber, 2004):

$$\varepsilon_p = \sum_{k=1}^{K} \sum_{j=1}^{J} \| m_j \cdot u_{jk} - c_k \|, \qquad \text{(Eq 3-2)}$$

where, c_k is the centroid of the k^{th} neuronal cluster and is given by:

$$c_k = \sum_{j=1}^{J} m_j \cdot u_{jk} \qquad \text{(Eq 3-3)}$$

The assignment u_{jk} is usually found through a two-step iterative procedure, starting from a random initialization. In the first step, the j^{th} neuron is assigned to the cluster with the nearest centroid, c_k :

$$\hat{k} = \arg\min_k (r_j - c_k)$$
$$u_{j\hat{k}} = 1; \quad u_{jk} = 0 \quad \text{if } k \neq \hat{k} \qquad \text{(Eq 3-4)}$$

In the second step, the positions of all *K* centroids is updated by Eq 3-3 which may necessitate changes in the assignment. The two steps are repeated until there are no further changes to u_{jk} and the centroids become stable. Since the above procedure could terminate at a local minima, the procedure has to be repeated multiple (*P*) times, with different initial assignments. The subscript *p* in ε_p signifies the total distance in the p^{th} replicate of the procedure. The assignment from the replicate with the minimum ε_p is selected.

In the following sections, these characteristics of SOM are exploited for representation and visualization of high-dimension process operational data.

3.4 Case Study 1: Visualization of distillation column startup operations

Process description

The flowsheet of the pilot-scale distillation unit is shown in Figure 3-3. The column is of 2 meters height and 20 cm width and has 10 trays, where the feed enters at tray 4. The system is well integrated with a control console and data acquisition system. 19 variables comprising of all tray temperatures, reboiler and condenser temperatures, reflux ratio, top and bottom column temperatures, feed pump power, reboiler heat duty, and cooling water inlet and outlet temperatures, are measured at 10-

second interval. Cold startup of the distillation column with ethanol-water at 30% v/v mixture is performed following the standard operating procedure (SOP) shown in Table 3-1. The feed passes through a heat exchanger before being fed to the column. The startup normally takes two hours and different faults such as sensor fault, pump failure, too high a reflux ratio, etc., can be introduced at different states of operation. The disturbances considered here are summarized in Table 3-2. Experiments for each disturbance were conducted and the operating data used to construct the plant historian. The 18 variables shown in Table 3-3 are measured at 10-second intervals.

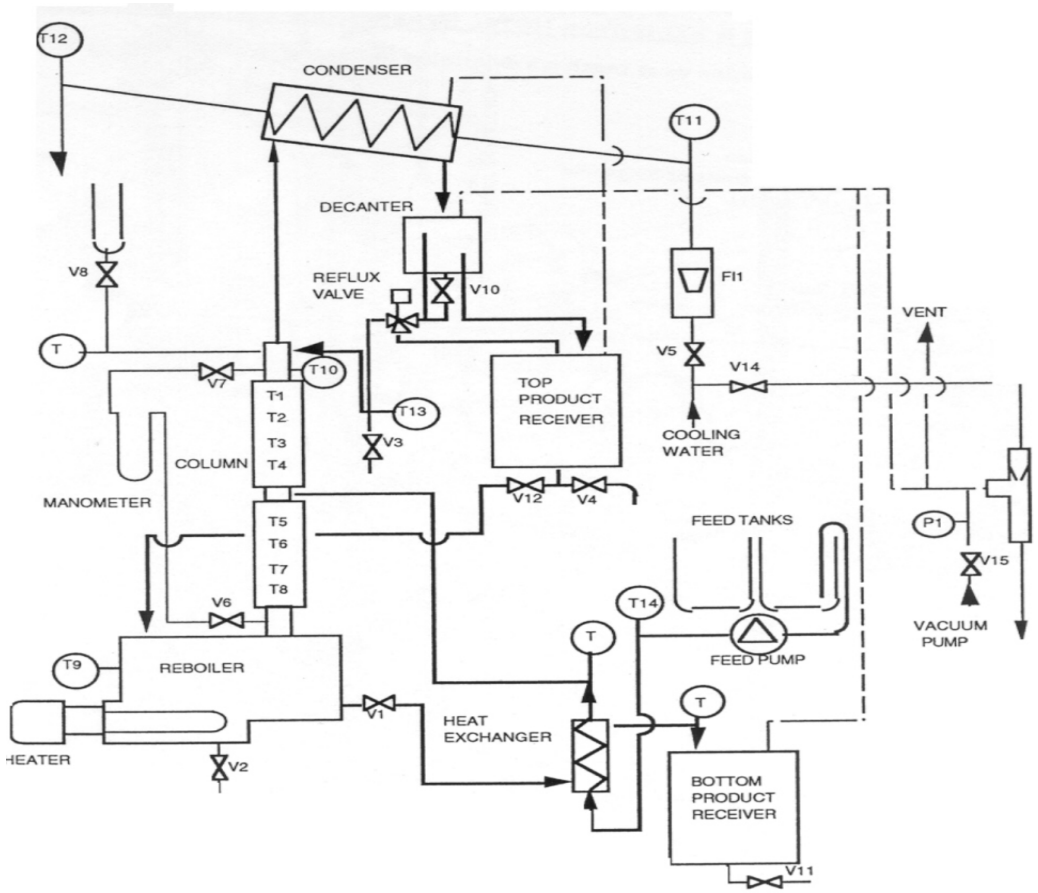


Figure 3-3: Schematic of the distillation unit set up

Table 3-1: Standard operating procedures (SOP) for distillation-unit startup

Distillation column startup SOP	
1.	Set all controllers to manual
2.	Fill reboiler with bottom product
3.	Open reflux valve and operate the column on full reflux
4.	Establish cooling water flow to condenser
5.	Start the power of reboiler heating coil
6.	Wait for all of the temperatures to stabilize
7.	Start feed pump
8.	Activate reflux control and set reflux ratio
9.	Open bottom valve to collect product
10.	Wait for all the temperatures to stabilize

Table 3-2: Process disturbances analyzed for distillation unit startup case study

Case	**Disturbance**	**Type**
DST01	Reboiler power low	Step
DST02	Reboiler power high	Step
DST03	Feed pump high	Step
DST04	Feed pump low	Step
DST05	Tray Temperature Sensor T6 fault	Random variation
DST06	Reflux ratio high	Step
DST07	Reflux ratio low	Step
DST08	Bottom valve	Sticking
DST09	Cooling water	Slow drift
DST10	Low cooling water flow and feed pump malfunction	Slow drift & Step

Table 3-3: Variables used for monitoring of the lab-scale distillation unit startups

Var	**Description**	**Unit**	**Range**
1.	Tray 1 temperature	$^{\circ}C$	20.5 - 89.5
2.	Tray 2 temperature	$^{\circ}C$	20.7 - 90.1
3.	Tray 3 temperature	$^{\circ}C$	20.8 - 90.2
4.	Tray 4 temperature	$^{\circ}C$	20.6 - 89.8
5.	Tray 5 temperature	$^{\circ}C$	20.4 - 91.3
6.	Tray 6 temperature	$^{\circ}C$	20.5 - 91.6
7.	Tray 7 temperature	$^{\circ}C$	20.7 - 91.1
8.	Tray 8 temperature	$^{\circ}C$	20.5 - 91.4
9.	Reboiler temperature	$^{\circ}C$	21.4 - 90.9
10.	Top column temperature	$^{\circ}C$	20.5 - 88.8
11.	Cooling water inlet temperature	$^{\circ}C$	21.1 - 27.2
12.	Cooling water outlet temperature	$^{\circ}C$	21.2 - 33.3
13.	Condenser inlet temperature	$^{\circ}C$	21.1 - 77.9
14.	Feed temperature	$^{\circ}C$	23.6 - 28.4
15.	Reboiler power	kW	0 - 2.0
16.	Feed pump speed	RPM	0 - 199.6
17.	Reflux cycle time	s	0 - 4
18.	Reflux ratio	-	0 - 4

Visualization of Distillation-Unit Startup

The SOM-based visualization methodology described above is illustrated on the startup of a lab-scale distillation unit. The startup normally takes about two hours. The evolution of some of the key variables in one run is shown in Figure 3-4.

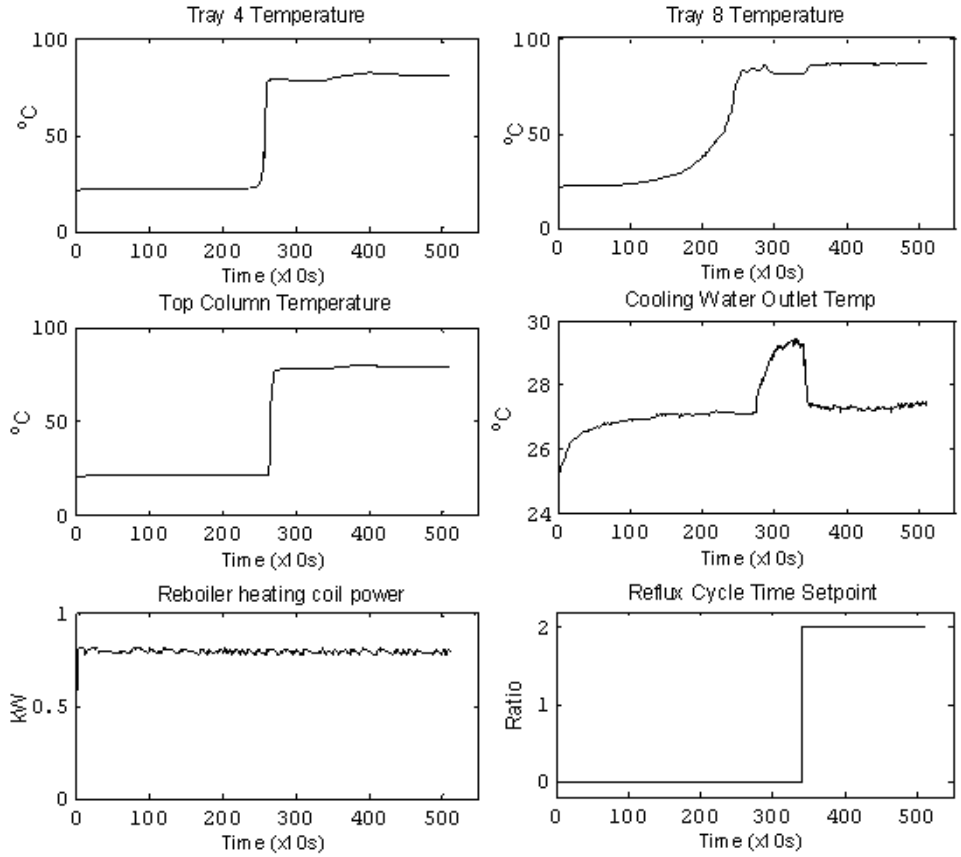


Figure 3-4: Process state variables observed during a normal startup of the distillation unit

To obtain the training data, eleven runs– one normal and ten with the disturbances listed in Table 3-2, were conducted resulting in a total of 5615 training samples. These data were first normalized by auto-scaling each of the 18 variables. Next, a SOM was designed. From Eq 2-5, J was selected to be 600. PCA was performed on the training data and the ratio of the square root of the first two eigen-values was found to be 3.07. This was used as the aspect ratio for the SOM leading to a map configuration of 43x14 neurons. A SOM with this configuration was trained with

all the training data and the average quantization error was found to be 0.303. Then *k*-means clustering with K=60 and P=1000 replicates was performed to identify neuronal clusters. The replicate with the smallest ε_p is shown in Figure 3-6, where all nodes belonging to the same cluster have been similarly shaded and labeled with the cluster number.

The available training data were then projected on the trained SOM to annotate it. Neuronal cluster 3 was found to correspond to the cold state and cluster 56 to the final steady state. This can be understood from Table 3-4 which lists the values of the 18 measured variables for some of the cluster centroid. The centroid of cluster 3 has all its tray temperatures around room temperature. Similarly, at cluster 56, the tray temperatures are around 80–85 °C, the feed temperature is also higher (from preheating), and reflux ratio is about 1. For visualizing the startup transition, the online measurement from the normal startup run were projected on the trained SOM and the evolution of the hits tracked. There are three major phases during the startup – reboiler heating, evaporation, and column stabilization – before steady-state is established. The time evolution of the neuronal-clusters hits is labeled in Figure 3-6 and summarized in Figure 3-5. The startup operation begins at cluster 3 in this run with the column at the cold start state. When the reboiler power is turned on, its temperature (var 9) increases to the boiling point ($\sim 90\,^{\circ}C$). The lower tray temperatures also increase slowly and the hits evolve through adjacent clusters from 6→12→16→17→19→20→22→25. The increasing values of the temperatures are reflected in the centroids of the clusters as shown in Table 3-4. When the reboiler contents start boiling (around 80 °C), the temperatures throughout the column rise significantly. This leads to a smooth evolution of hits through clusters 26→27→29→32→33→37→43. Finally the column is stabilized by establishing the feed flow (var 16) and the hits move to clusters

47→52→54→55. Starting at *t*=3890s, the hits remain at cluster 56, indicating that the process has reached the final steady state. Data from the abnormal runs were also used to annotate the SOM. Figure 3-7 shows the various clusters that correspond to the various faults.

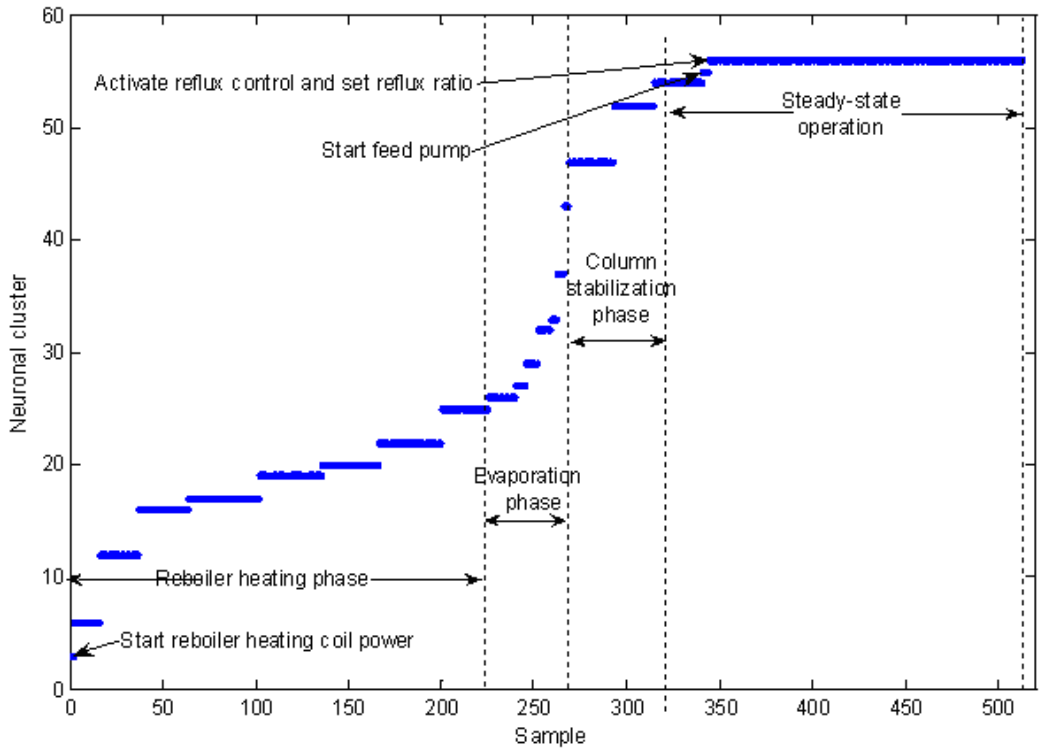


Figure 3-5: Operating state identification based on SOM

The startup transition demonstrates that continuous operations are exhibited as smooth trajectories on the SOM space wherein successive hits are in close proximity. Discrete events on the other hand are represented as abrupt jumps on the SOM. For example, when the feed pump is started at *t*=3410s, the hit moves across eight intervening neurons from a neuron in cluster 55 to another in cluster 56. Similarly, when the reflux ratio is changed from 0 to 1 at *t*=3430, the hits jump over three intermediate neurons.

Table 3-4: Cluster centroids corresponding to various states of the startup transition

Cluster id	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	State
3	21	21	21	21	21	21	22	22	24	21	25	25	22	24	0.3	0	0	0	Cold Start
6	21	21	21	21	21	21	22	22	27	21	26	26	22	24	0.7	0	0	0	Reboiler Heating starts
12	21	21	22	21	21	21	22	22	33	22	26	26	22	24	0.9	0	0	0	
16	22	22	22	22	21	22	22	22	40	22	27	27	22	25	0.8	0	0	0	
17	22	22	22	22	21	22	22	23	49	22	27	27	22	25	0.8	0	0	0	
19	22	22	22	22	22	22	23	25	58	22	27	27	22	26	0.8	0	0	0	
20	22	22	22	22	22	22	24	28	67	22	27	27	22	26	0.9	0	0	0	
22	22	22	22	22	22	23	26	32	74	22	27	27	22	27	0.8	0	0	0	
25	22	22	22	22	23	25	31	41	80	22	27	27	22	27	0.8	0	0	0	
26	22	22	22	22	24	31	39	51	84	22	27	27	22	27	0.8	0	0	0	Evaporation starts
27	22	22	22	22	27	40	52	64	87	22	27	27	22	27	0.8	0	0	0	
29	22	22	22	23	36	58	66	73	88	22	27	27	22	27	0.8	0	0	0	
32	22	22	25	34	60	75	77	80	89	22	27	27	22	27	0.7	0	0	0	
33	26	30	41	58	74	79	79	82	89	25	27	27	22	27	0.8	0	0	0	
37	45	57	68	74	78	79	80	83	89	37	27	27	22	27	0.8	0	0	0	
43	65	72	75	76	79	80	80	83	89	58	27	27	22	27	0.7	0	0	0	
47	76	78	78	78	79	80	80	84	90	76	27	28	22	27	0.8	0.6	0	0	Column Stabilization starts
52	78	78	78	78	79	80	79	82	90	78	27	29	25	27	0.8	4.3	0	0	
54	78	78	78	78	79	80	79	82	90	79	27	30	31	27	0.8	6.2	0	0	
55	78	78	79	79	82	82	81	83	90	79	27	29	30	28	0.8	113.5	0.1	0	
56	79	80	80	81	86	86	85	87	90	80	27	28	27	28	0.8	114	2	1	Final Steady state

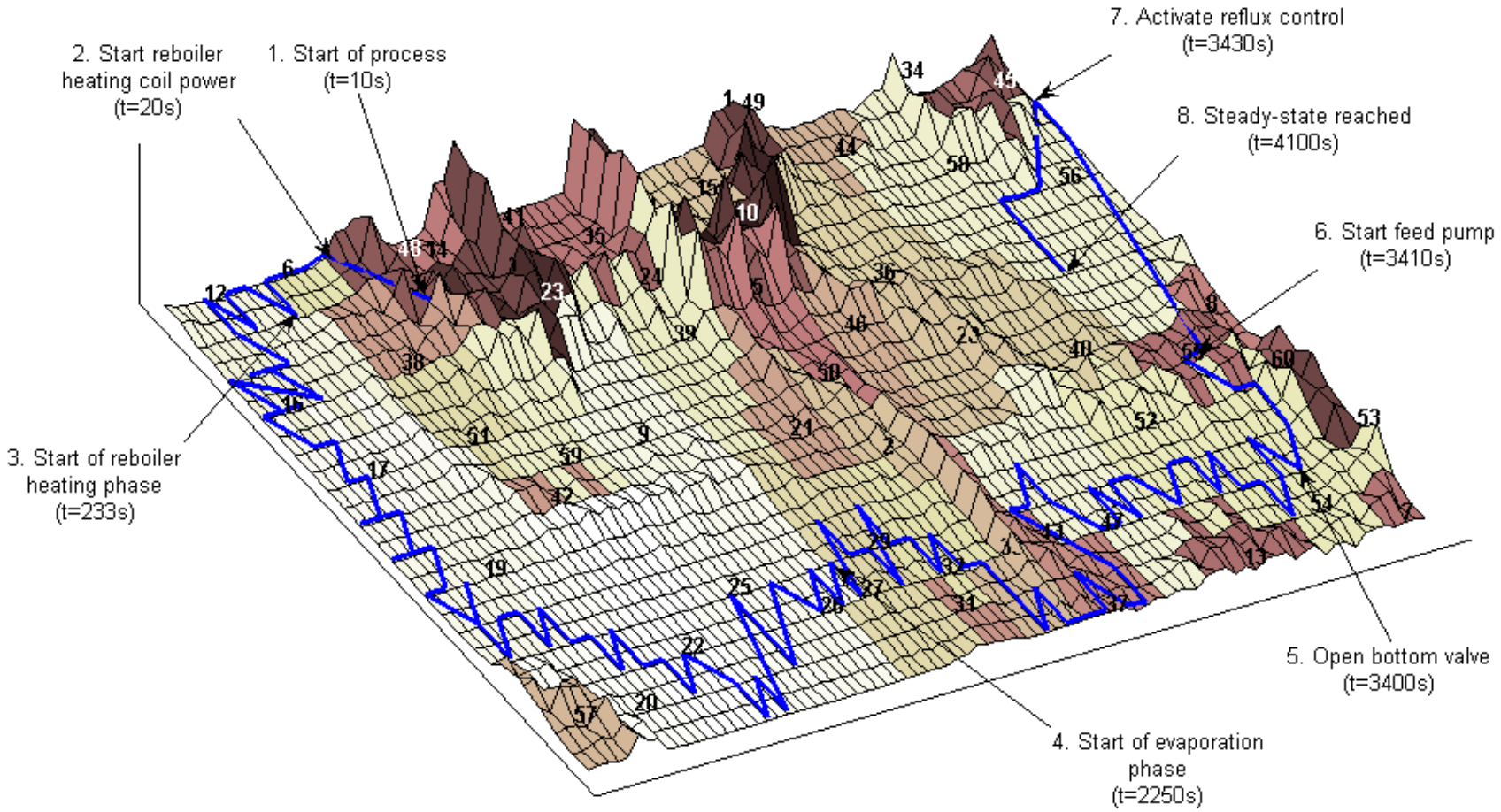


Figure 3-6: Trajectory of normal startup of distillation unit as projected on SOM

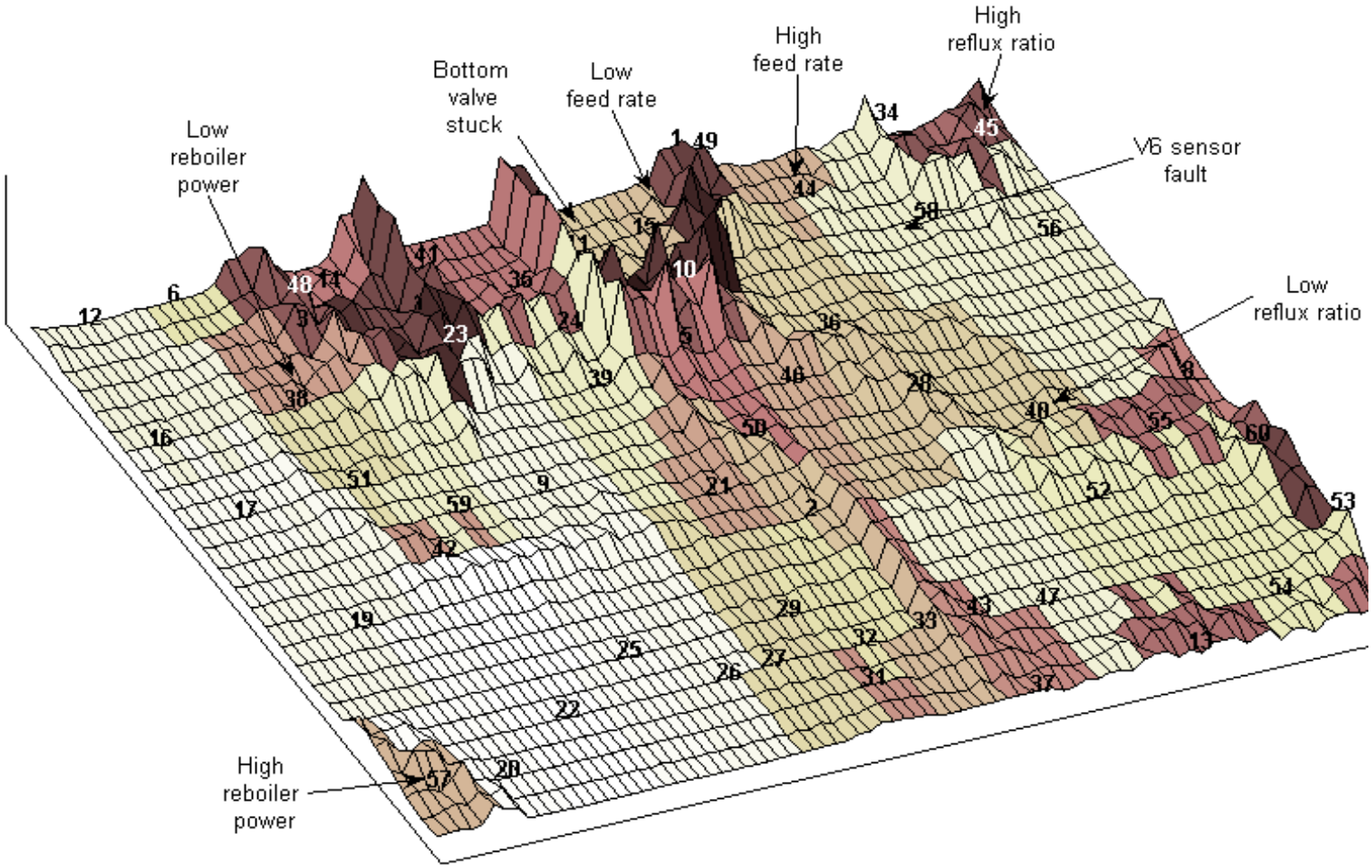


Figure 3-7: Visualization of various process faults on SOM

Visualization of abnormal runs

Next, the use of the trained SOM for fault detection is described. The process signals for DST01 are shown in Figure 3-8 with the solid lines representing the signals for the abnormal operation while the dotted lines represent a normal run. The fault was introduced at *t*=10s. The online samples are projected on the SOM and its trajectory visualized as shown in Figure 3-9. The process can be seen to follow an abnormal trajectory from the beginning, with the hits evolving from cluster 3 to 38 instead of cluster 6. When the feed pump is activated (at t=5160s), the startup operation becomes unsuccessful as there is not enough heat supply to supplement the heat of evaporation. The temperature in most trays start to fall and the process trajectory is seen to move back toward the cold state (cluster 3).

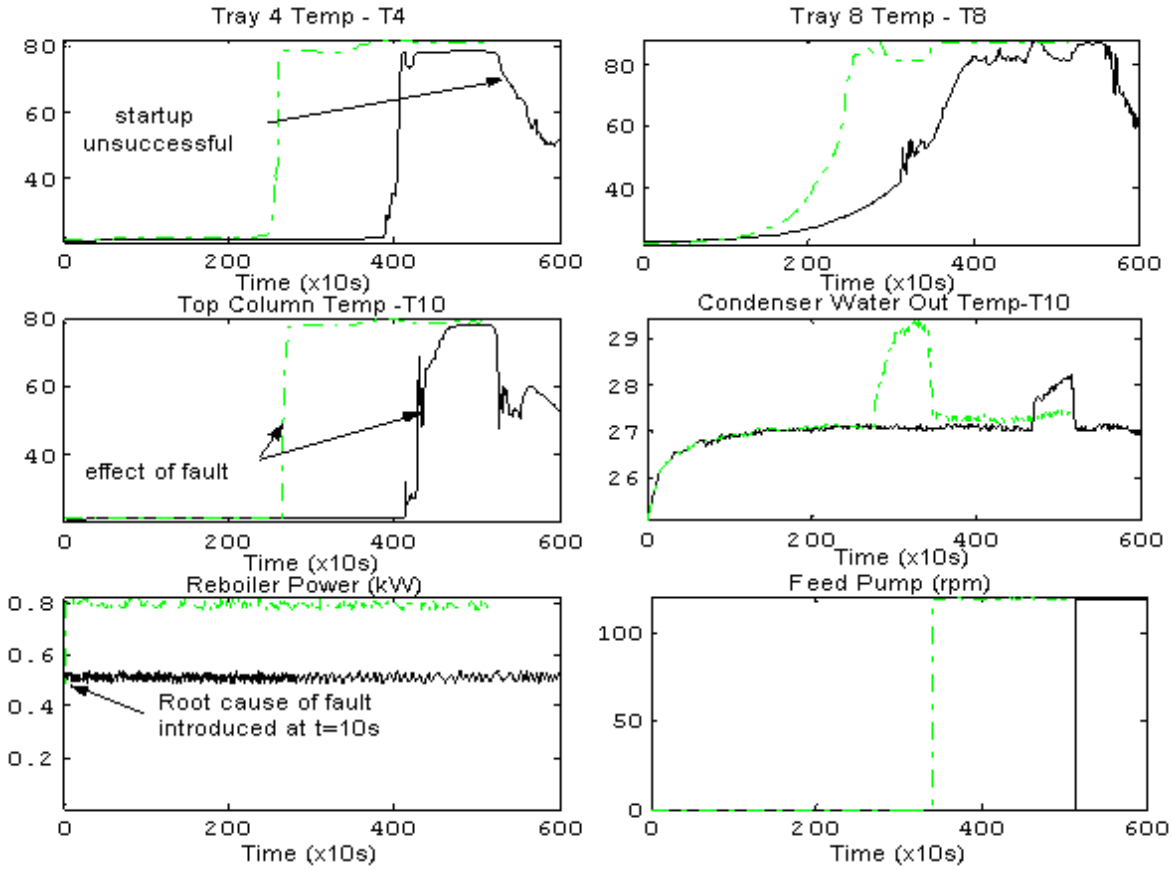


Figure 3-8: Comparison of process state variables during DST01 and normal startup

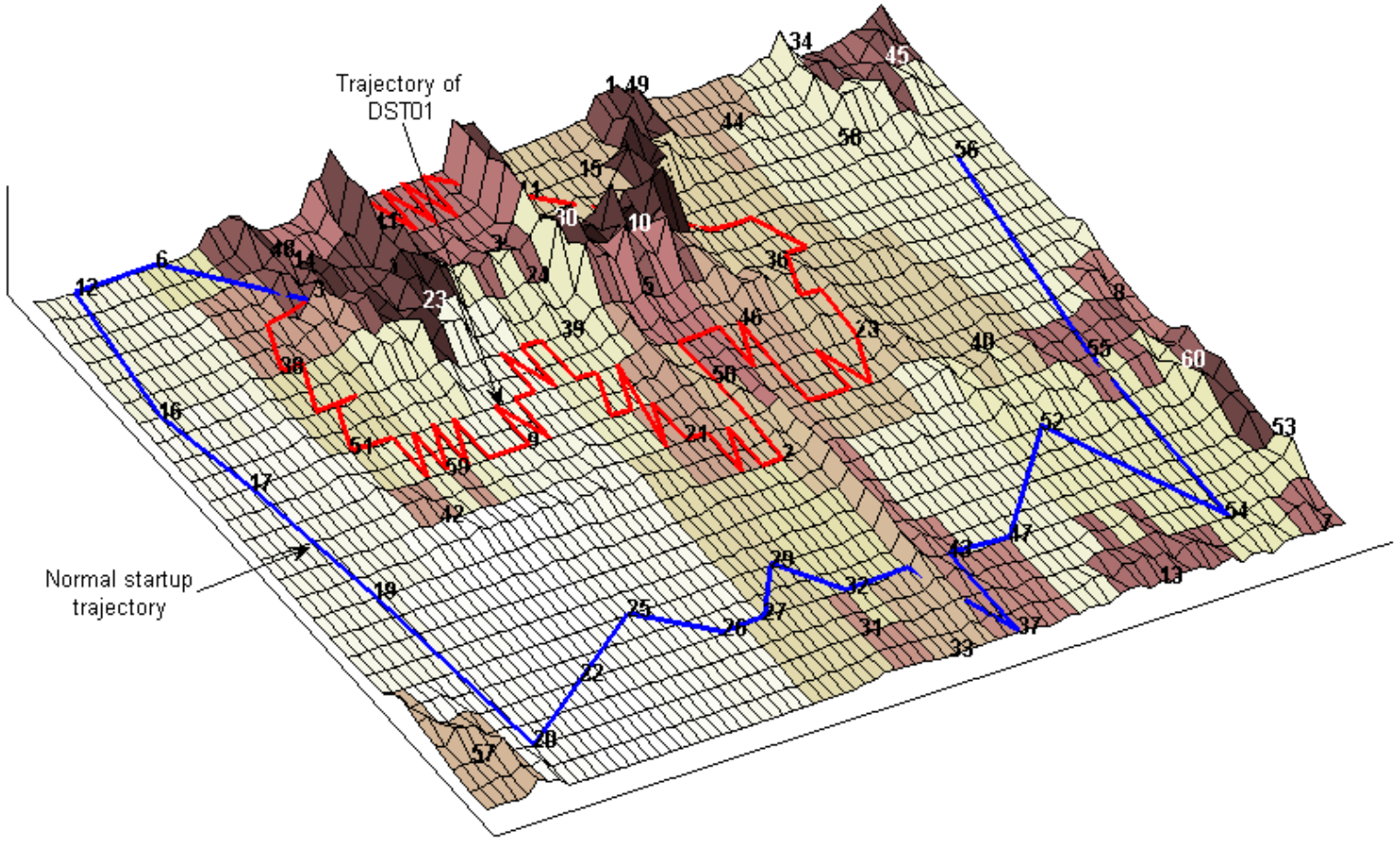


Figure 3-9: Visualization of process operation during run DST01

During another run, a fault (human error) was induced at t=3530s when the reflux ratio was set to 2.5 times its nominal value. This leads to a reduction in product throughput and increases the load of the reboiler. In this run, samples prior to t=3540s broadly correspond to normal operation although there are minor run-to-run deviations (for example from t=2500s to t=3500s). The trajectory on the SOM absorbs these minor differences and the sequence of cluster hits remains the same. At t=3540s, when the fault is introduced, the SOM shows an abnormal hit on cluster 45 which corresponds to DST06. Although only two of the measured variables (var 17 and var 18) reflect this fault, the variation can be clearly observed on the SOM space. This illustrates two key benefits of SOM visualization methodology, (i) dimensionality reduction – the operator does not need to monitor the different variables individually, rather the reduced 2-dimensional SOM map can reveal variations effectively, and (ii) diagnosis support – the abnormal hits serve as a signature of the fault, so the operator can quickly perform fault diagnosis based on the location of the hit on the annotated SOM. The latter can also be used for automated fault diagnosis as described Chapter 4 of this thesis.

One key issue while forming the neuronal cluster is to determine the number of clusters. The number of clusters, K, affects the selectivity and sensitivity of SOM when it is used to track process operations. A larger K results in more neuronal clusters on SOM, and hence improves the system's ability to represent different states (including disturbance classes). However, the SOM also becomes more sensitive with increasing K, and could result in false alarms. To evaluate the effect of K, the same SOM was clustered with various K values. For all K values in the range $K \in [40, 70]$ all the faults could be differentiated from the SOM visualization. The next section describes the application of the proposed visualization method to an industrial case study.

3.5 Case Study 2: Transition Identification & Visualization in an Industrial Hydrocracking Unit

The process analyzed in this section is the boiler of a Hydro-cracking Unit (HCU) in a major refinery in Singapore (Ng *et al.,* 2006). Hydro-cracking is a versatile process for converting heavy petroleum fractions into lighter, more valuable products. The objective of HCU is to convert heavy vacuum gas oil (HVGO) to kerosene and diesel with minimum naphtha production. The simplified process flow diagram of the HCU is shown in Figure 3-10. The operations of the HCU considered are complex, and involves catalytic hydro-cracking reactions in a hydrogen-rich atmosphere at elevated temperatures and pressures. The HCU includes two sections, a reactor section and a fractionation section. Integrated to both these sections is a waste heat boiler (WHB) unit for heat recovery. This section illustrates the application of SOM for visualizing different operating states in the WHB unit.

3.5.1 Analysis of operating data from Waste Heat Boiler

In this study, one month of operating data consisting of 21 measured variables from the WHB unit sampled at five-minute interval is considered. The data was auto-scaled and used to train a SOM with 468 neurons and dimension of 39×12. The trained SOM was then clustered with $K = 70$. The clustered SOM was annotated with the typical regions of operation. As can be seen from Figure 3-11, the WHB unit operates in 5 different modes, shown as M_1 to M_5. Analysis of the SOM showed that the unit underwent 7 different transitions during the period under consideration. Figure 3-12 shows instances of four of these transitions. Next, the SOM is used to visualize transitions in the WHB. Mode M_2 corresponds to the production of steam at 22T/hr, while mode M_3 corresponds a throughput of 14T/hr.

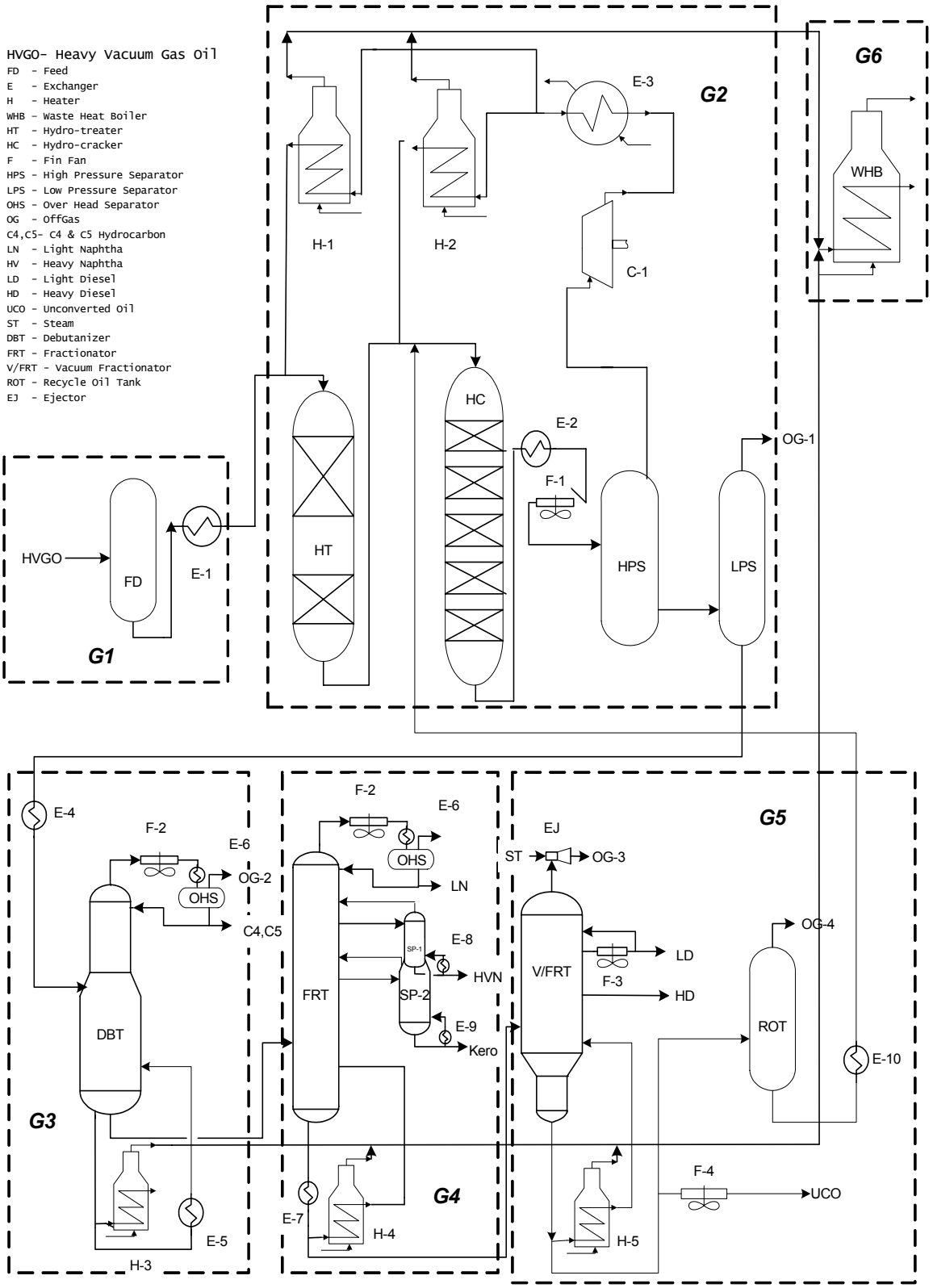


Figure 3-10: Process flow diagram of the refinery hydro-cracking unit

One instance of this transition is depicted as T_{23} in Figure 3-12 and requires 330 mins for completion.

The trained SOM was also used to visualize the operation during another 15-day period. Data from this period was not used during the training and therefore demonstrates the SOMs generalization-ability. The mean quantization error for this period was 0.906, indicating that SOM provides a good representation for these as well. During this period, the plant was observed to operate in mode M_3 for 83% of time, about 3% in M_2 and 4% in M_4. The process underwent transitions for a total of 32.5 hours (~10%) during this period All the transitions could be easily visualized with the previously trained SOM.

For the purpose of comparison, the same data was visualized using PCA. The first three PCs captured 85.47% of the variance as shown in Figure 3-13. Data from the five modes identified from the SOM are shown in the biplot. In contrast to the SOM, the different modes of operation are not as clearly delineated by PCA.

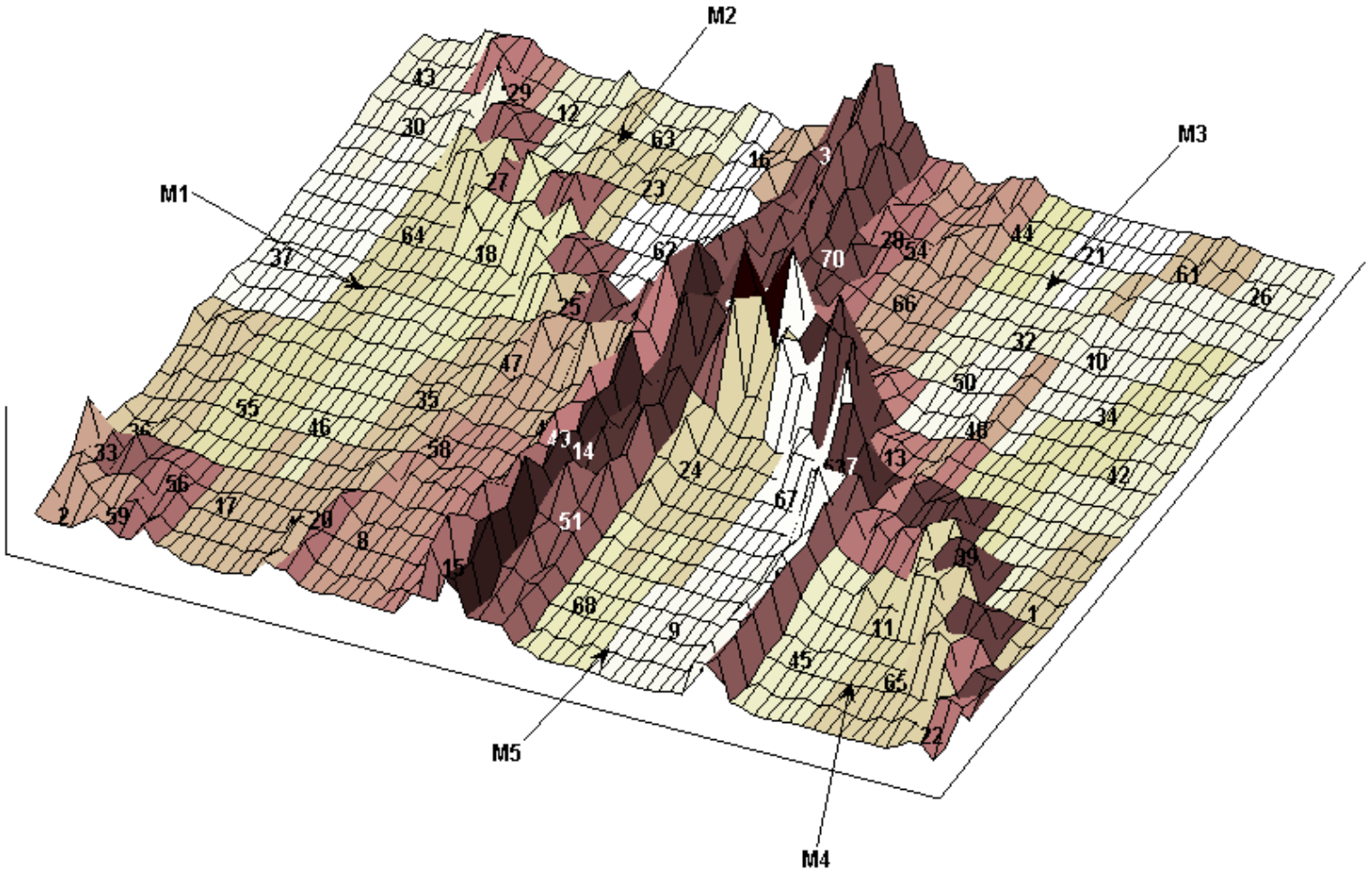


Figure 3-11 : Operation map constructed for unit Waste Heat Boiler

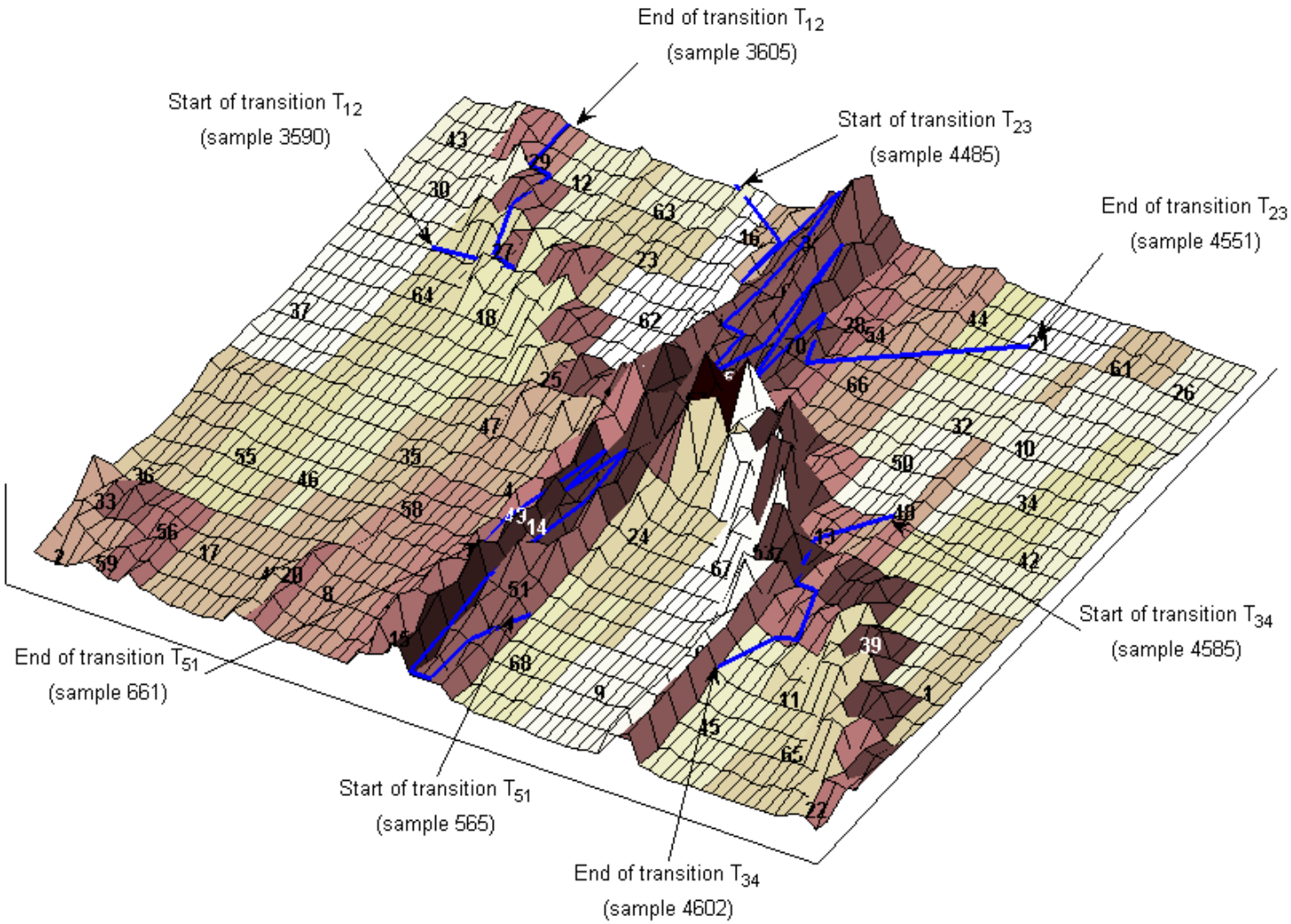


Figure 3-12: Visualization of transition trajectories for the refinery WHB unit

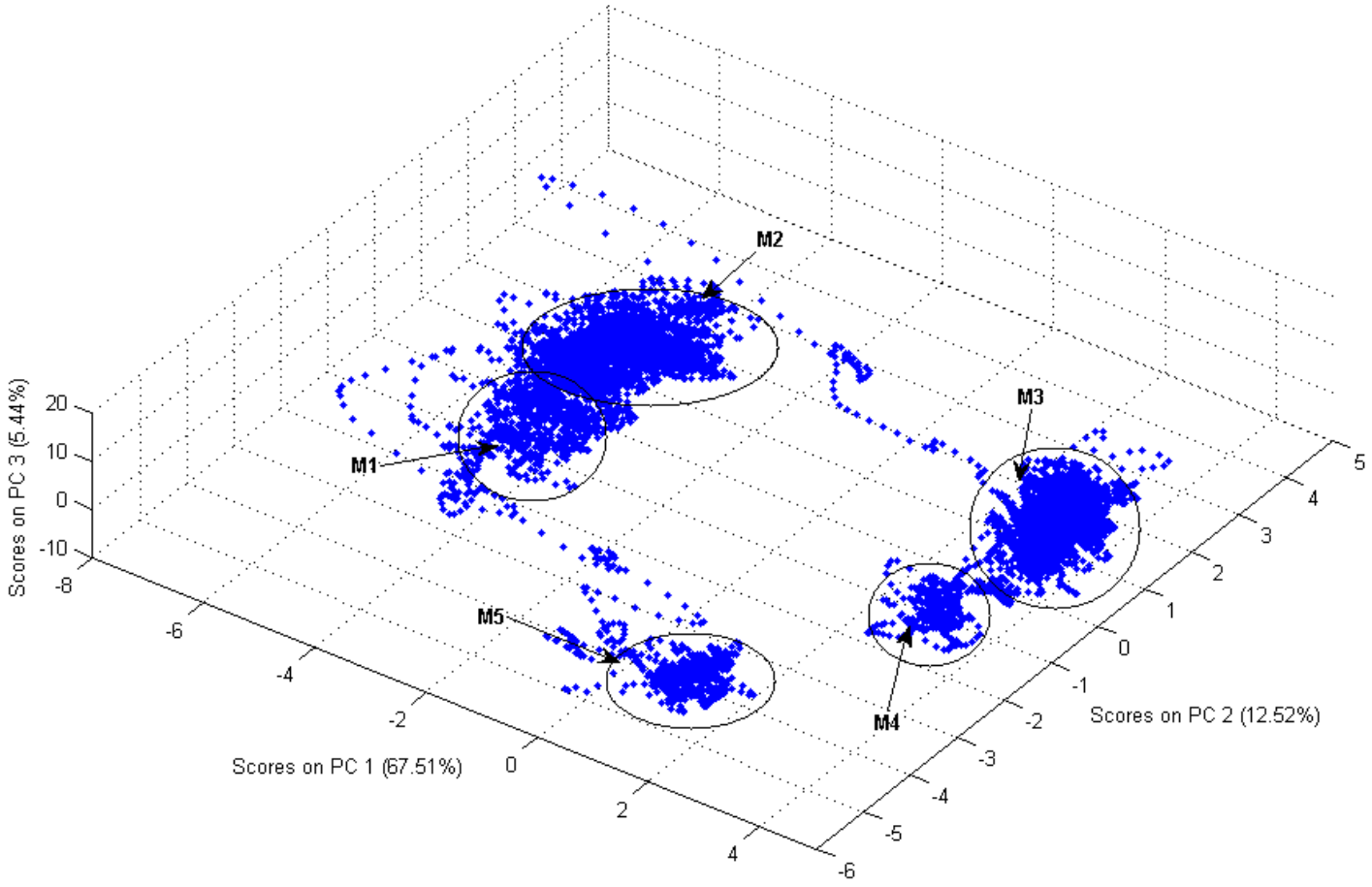


Figure 3-13: Visualization of the WHB unit operation using first three scores

3.6 Conclusions

Methods that enable effective visual exploration are crucial for extracting knowledge from complex, high-dimensional, temporal, multi-state data. In this work, the self organizing map has been shown to be capable of providing a method to reduce dimensionality and visually depict high dimensional process data in an intuitive graphic. Process operation can be represented in the SOM with each state – mode or transition – having a distinct representation. Process modes and transitions are mapped differently onto the SOM. Process modes, with its process variables exhibiting near-constant values, are projected to a small neighborhood on SOM. In contrast, process transitions exhibit large changes in their variables, and are visualized as trajectory on the SOM by connecting the hits identified from SOM with lines. In cases where the underlying process has high noise levels, an additional layer of abstraction is desirable. In such cases, neighboring neurons are combined into a neuronal cluster which maps to a broad range of process operation.

Application of the proposed approach to two case studies - startup of a distillation unit, and operations of an industrial boiler within a hydro-cracker in a major refinery illustrate the benefits of visualization in extracting process knowledge even from complex, multi-state operations. Visualization of multi-state operations using the SOM results in a map of the process that has numerous uses. As demonstrated in the two case studies, values insights about the process operations can be obtained by analyzing historical operations data. The different states that the process operates in can be segregated. If multiple instances of the same state are present in the data, they can be compared. The trained map can also be used for real-time state identification by identifying the location of the latest BMU on the annotated SOM.

The SOM represents multi-state data as neurons on the SOM space that preserves the topological relationships of the measurement space. Differences between a node and its neighboring neurons are visualized through U-matrix. So dissimilar neighbors that would be distant in the original space are separated by nearby neurons separated by large U values, and depicted as dark mountains on the map. In contrast to traditional dimensionality reduction approaches such as PCA, which preserve global distances, the SOM dedicates neurons to an operating region only if it is present in the training data. It thus offers a more compact and rich representation of the operation which can be exploited for process monitoring as well, as proposed in the next chapter.

Nomenclature

Indices

i sample

j,j' neuron in SOM model

n variable

$k, \hat{k}$ cluster

p replicate in k-means clustering

Parameters

I total number of samples (rows) in X

J total number of neurons in SOM

K total number of neuronal clusters

N total number of variables (columns) in X

P total number of replicates for clustering

Variables

b_i BMU for sample x_i

c_k centroids for k^{th} cluster

$D_{jj'}$ distance between neurons j and j'

E_i^q quantization error for sample x_i

$h_{b_i j}$ neighborhood function of b_i

M^{SOM} set of neurons $= \{m_1, ..., m_j, ..., m_J\}$

m_j reference vector of neuron $j = \{m_{j1}, ..., m_{jn}, ..., m_{jN}\}$

$\mathcal{N}_j$ set of neurons that are topological neighbors of neuron j

r_j location of neuron j in two-dimensional grid

S_k	k^{th} neuronal cluster
X	training dataset $=\{x_1,...x_i,...,x_I\}$
x_i	i^{th} sample in $X=\{x_{i1},...,x_{in},...,x_{iN}\}$
u_{jk}	1 if the class membership of neuron j equal to k
α	learning rate factor
ε_p	total cluster assignment distance in replicate p
σ	neighborhood width

Chapter 4 A Self-organizing Map based Methodology for Process Monitoring

4.1 Introduction

This chapter describes a SOM-based methodology to detect and diagnose process disturbances during process operations. In the proposed approach, process operations (modes and transitions) are first represented in the SOM space. The SOM can serve as a high-fidelity model for any process operation. Steady states tend to form clusters of BMUs while transient operations are reflected as trajectories. Differences between two states can then be quantified based on the location and evolution of their corresponding BMUs. A method for comparing two trajectories and different means of performing disturbance identification are also presented.

4.2 SOM for Process Operations

The SOM can be used to represent various process operations (Ng and Srinivasan, 2004a). The training of SOM is normally performed using all available data collected from previous operations, which includes data from modes and transition operations. The incorporation of all data enables SOM to represent a wide range of operating conditions on its topology map, and thus giving SOM better classification ability. During training, the neurons on the SOM will orient themselves and evolve into a process map representing all the operating conditions in the training data. Such a trained SOM can then be used to identify the process state in real-time. Data from online sensors, x_i, can be projected on the SOM and the BMU, m_{b_i}, are identified. The location of m_{b_i} on the SOM indicates the current state of the process.

Process modes and transitions display different characteristics on the SOM space. When a process is in a mode, all its variables have near-constant values. Therefore, online measurements from such a state should be projected on the same BMU. Noise and minor variations in process operation could result in projection of the online measurement to different BMUs, however these would be neighboring neurons because of the topology preserving feature of SOM training. Process modes can thus be identified when a high frequency of BMUs are found within a small neighborhood in the map. Process variables have significantly different values between different modes. So, different modes can be distinguished on the SOM based on difference in the BMUs.

4.3 Representing Process Operations using State-signatures

Consider the signal segment $X = \{x_1,..,x_i,..,x_I\}^T$ observed from a process as it operates in a specific state. Here, the i^{th} element $x_i = \{x_{i1},...,x_{in},...,x_{iN}\}$ is the *N*-dimensional measurement from the process at time *i*. As illustrated in Figure 4-1, each x_i can be projected on a suitably trained SOM $M^{SOM} = \{m_1,...,m_j,...,m_J\}^T$ resulting in a hit on BMU b_i.

$$b_i = \arg\min_j \{x_i - m_j\} \qquad \text{(Eq 4-1)}$$

If the neurons in the SOM have been previously grouped as described in Chapter 3 into neuronal-clusters $S = \{S_1,...S_k,...S_K\}$, where $S_k \in [1,K]$, a mapping u_{jk} between the neurons and the neuronal-cluster would have been established. Let $\hat{S}_i$ be the cluster-hit of x_i, $\hat{S}_i$ can be identified from the clustering space of the BMU of x_i as:

$$\hat{S}_i = \arg\max_k (u_{b_i k}) \quad \text{(Eq 4-2)}$$

As the process evolves with time, a sequence of cluster-hits $\hat{S}^X = \left\{\hat{S}_1, ..., \hat{S}_i, ..., \hat{S}_I\right\}$ would be observed, corresponding to each x_i in X. However, several x_i in a small range would have the same $\hat{S}_i$, so the sequence of cluster-hits would have repetitions. For efficient representation and comparison without consideration of timing differences, the repetition-free cluster sequence information is extracted from this sequence. The *state-signature* Σ^X is the ordered set of hits *across* neuronal-clusters arising during X, ignoring consecutive hits on the same neuronal-cluster. $\Sigma^X = \{\Sigma_1, ..., \Sigma_r, ..., \Sigma_R\}$, where $\Sigma_r \in \left\{S_1, ..., S_k, ..., S_K\right\}$, $R \le I$. Typically, $R << I$. The *dwell-time* t_r in a neuronal-cluster Σ_r denotes the number of samples (hits) that map to the cluster. The *dwell-time signature* for X is the set of dwell-times ordered as per the neuronal-clusters in the state-signature. $T^X = \{t_1^X, ..., t_r^X, ..., t_R^X\}$. The derivation of Σ^X and T^X from the sequence of cluster-hits has to consider two cases:

Case 1: Cluster-hit continuation – when the cluster-hit for x_i is the same as that for x_{i-1}, i.e., $\hat{S}_i = \hat{S}_{i-1}$: $\hat{S}_i$ is not included in Σ^X, only the dwell-time of $\hat{S}_{i-1}$ is updated. $t_r = t_r + 1$.

Case 2: Cluster-hit change – when the cluster-hit for x_i is different from that for x_{i-1}, i.e., $\hat{S}_i \neq \hat{S}_{i-1}$: $\hat{S}_i$ is inserted as a new element in Σ^X, i.e., $\Sigma^X \leftarrow \{\Sigma^X, \hat{S}_i\}$. A corresponding new entry is also included in $T^X \leftarrow \{T^X, 1\}$

Two instances of the same transition with no variation in the temporal profiles of the measurements (except process noise) would have the same state- and dwell-time-signatures. If the magnitude profile is the same between the two instances, but

different states persist for different durations (i.e., the two instances are not time synchronized) due to run-to-run or operator-to-operator variations, then their dwell-time signatures would differ, but the state-signature would be the same. The state-signature of two completely dissimilar transitions would be different since the sequence of cluster-hits would be distinct. This makes the proposed neuronal-cluster representation ideal for efficient transition monitoring and fault diagnosis.

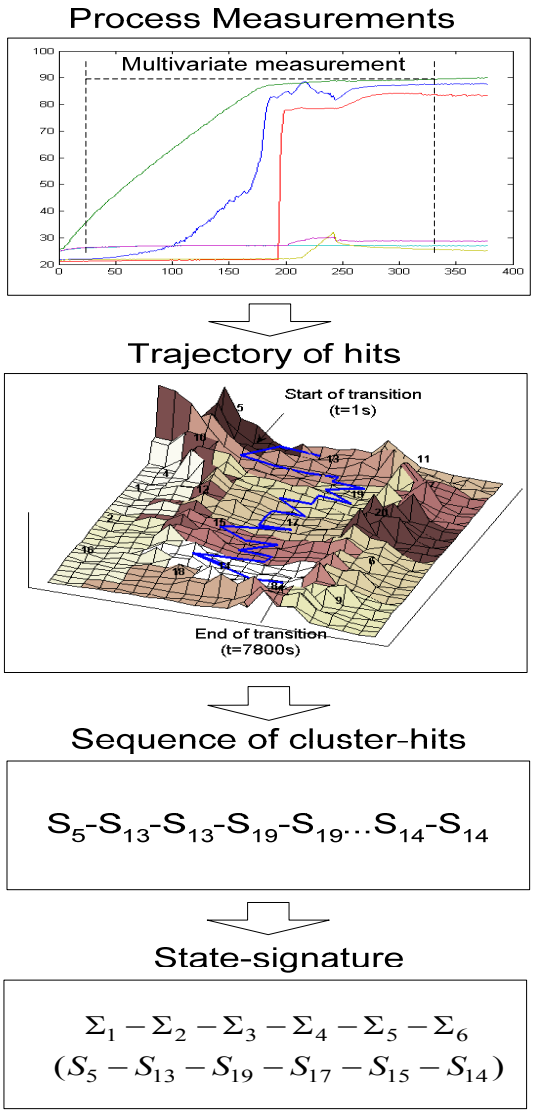


Figure 4-1: Abstraction of multivariate process data into state-signature

4.4 State-signature Comparison

The state-signatures can be used for online state identification. Consider the signal segment $X = \{x_1,..,x_i,..,x_I\}^T$ which is the last *I* samples from the multivariate process. This signal has to be compared with a collection of *W* reference signals in database Ψ to identify the one that best matches *X*. Let $Y = \{y_1,...,y_p,...,y_P\}^T$ be one of the reference signals. Let $M^{SOM} = \{m_1,...,m_j,...,m_J\}$ be a SOM that has been trained with all the signals in Ψ, and subsequently clustered using *k*-means. Let the state-signature of *X* be $\Sigma^X = \{\Sigma_1^X,...,\Sigma_r^X,...,\Sigma_R^X\}$ and that of *Y*, $\Sigma^Y = \{\Sigma_1^Y,...,\Sigma_q^Y,...,\Sigma_Q^Y\}$. Since both *X* and *Y* are projected onto the same trained SOM as $\Sigma_r^X \in [1,K]$ and $\Sigma_q^Y \in [1,K]$, the two can be compared.

A distance between Σ^X and Σ^Y can be defined as:

$$\tilde{D}\left(\Sigma^X,\Sigma^Y\right) = \sum_{r=1}^{R}\left|d\left(\Sigma_r^X,\Sigma_r^Y\right)\right| \qquad \text{(Eq 4-3)}$$

$$d(\Sigma_r^X,\Sigma_q^Y) = \| c_{\Sigma_r^X} - c_{\Sigma_q^Y} \|, \quad \Sigma_r^X,\Sigma_q^Y \in [1,K] \qquad \text{(Eq 4-4)}$$

If a more precise comparison including timing is necessary, T^X and T^Y can also be compared to account for dwell-time in the different states.

$$\tilde{E}\left(T^X,T^Y\right) = \sum_{r=1}^{R}\left|\frac{t_r^X - t_r^Y}{t_r^Y}\right| \qquad \text{(Eq 4-5)}$$

However, this does not account for small local variations between *X* and *Y*. Therefore, the direct comparison described above is of limited use for online comparisons. During online monitoring, since the real-time signal *X* is generated dynamically, it is incomplete and the corresponding start time and states vis-à-vis the reference signal is not known a priori. Also, noise and run-to-run variations could lead

to some differences in the state-signatures even from two instances of the same transition. Consequently, brute-force direct comparison of Σ^X and Σ^Y is not suitable. Here, an efficient approach is devised using a concept similar to the local sequence alignment approach of Smith and Waterman (1981). The Smith-Waterman algorithm has to be extended to handle a characteristic specific to process operations – oscillations in measurements. Oscillations occur commonly in chemical processes due to a variety of reasons including disturbances, poorly tuned controllers, control valve problems, etc. These oscillations would be partly abstracted away by the state-signature since a range of x_i map to the same BMU and a set of BMUs map to the same neuronal-cluster. However, large oscillations would correspond to hits across two (or more) different neuronal-clusters – so there would be multi-element repeats in the state-signature, and the number of repeats could vary between two instances of the same transition. The Smith-Waterman algorithm considers repeats of single elements, but it does not adequately handle multi-element repeats. The algorithm is therefore extended to handle such cases.

The proposed state-signature comparison algorithm is based on a dissimilarity matrix derived from the principle of optimality and dynamic programming. The main steps in aligning the state-signatures for performing similarity analysis between two state-signatures Σ^X and Σ^Y are:

Step 1: Dissimilarity matrix initialization. The first column and row of the dissimilarity matrix H, whose size is $Q \times R$, are initialized at this stage:

$$H_{q1} = d(\Sigma_1^X, \Sigma_q^Y),\ q \in [1, Q] \qquad \text{(Eq 4-6)}$$

$$H_{1r} = H_{1(r-1)} + d(\Sigma_{r-1}^X, \Sigma_r^X),\ r \in [2, R] \qquad \text{(Eq 4-7)}$$

The reader would note that Eq (4-6) is similar to Needleman-Wunsch and Eq (4-7) to Smith-Waterman. This initialization ensures that a complete match is obtained for X based on a sub-sequence in Y.

Step 2: Matrix propagation. Other elements of H are constructed recursively using distance between the centroids. As per the classical Smith-Waterman algorithm, three alternatives are considered for each $H_{(q+1)(r+1)}$.

a) extending the alignment of Σ_r^X and Σ_q^Y with a match between Σ_{r+1}^X and Σ_{q+1}^Y;

b) allowing a missing element in Y, i.e., extend the alignment to Σ_{r+1}^X and Σ_q^Y.

c) allowing a missing element in X, i.e., extend the alignment to Σ_r^X and Σ_{q+1}^Y; and

In this chapter, the additional case of oscillation in the signals that is manifested as repeats in the online state-signature is considered. Repeats are eliminated while constructing the reference state-signatures Σ^Y during offline model development. Without loss of generality, only two-element repeats is considered in Σ^X, i.e.,

d) allowing the recurrence of the previous element in Y, i.e., match Σ_{r+1}^X with Σ_{q-1}^Y

The option that yields the lowest distance is selected as the value of $H_{(q+1)(r+1)}$

$$H_{(q+1)(r+1)} = \min \begin{bmatrix} H_{qr} + d(\Sigma_{r+1}^X, \Sigma_{q+1}^Y) \\ H_{q(r+1)} + d(\Sigma_q^Y, \Sigma_{q+1}^Y) \\ H_{(q+1)r} + d(\Sigma_r^X, \Sigma_{r+1}^X) \\ H_{qr} + d(\Sigma_{r+1}^X, \Sigma_{q-1}^Y) \end{bmatrix} \quad \text{(Eq 4-8)}$$

Since the segment in Σ^Y that matches the complete Σ^X was sought, the smallest distance between them is obtained as the smallest value in the last column, i.e., H_{q^*R} where

$$q^* = \arg\min_q \left\{ H_{qR} \right\}, \quad q \in [1, Q]$$

$$D\left(\Sigma^X, \Sigma^Y\right) = H_{q^*R} \tag{Eq 4-9}$$

In contrast to $\tilde{D}$ above, D discounts minor mismatches and oscillations; hence $D \le \tilde{D}$.

Step 3: Back-tracing. Once the alignment matrix H is fully computed, back-tracing can be used, if necessary, to recover the two aligned state-signatures, Σ^{X*} and Σ^{Y*} from H. Here, the optimal predecessor is recursively identified starting from H_{q^*R}.

4.5 Transition Monitoring and Diagnosis

The state-signature comparison algorithm proposed above can be used for transition monitoring and diagnosis as follows. When a transition is monitored in real time, the state-signature $\Sigma^X = \{\Sigma_1^X, ..., \Sigma_r^X, ..., \Sigma_R^X\}$ emanating from the process is compared with the reference signature for the normal transition, say Σ^Y. The process is deemed to have entered an abnormal state if there is a large difference in the sequence of neuronal-clusters or the dwell-times:

$$State = \begin{cases} Normal & D\left(\Sigma^X, \Sigma^Y\right) \le D^{\max} \\ Abnormal & Otherwise \end{cases} \tag{Eq 4-10}$$

where, $D^{\max}$ is a user-defined threshold. Comparison is performed with the W reference signals in the historical database Ψ when an abnormality is detected. The state-signatures for each of the fault candidates contain only the portion corresponding to the abnormal operating region. Therefore, Σ^X is also truncated, retaining only the last one or more elements that do not match the previously known reference signature (Y). The truncated Σ^X is compared with the signatures in Ψ. The fault is conclusively identified when exactly one of the reference signals, say Σ^Z in the database significantly matches the real-time signature. The specificity of the match is measured

using inseparability α, which is defined as the ratio of the distance of the best matching reference state-signature and that of the second-best one. A small value of α (≈ 0) implies that the real-time signal clearly matches a specific reference pattern while $\alpha \approx 1$ implies that the real-time state-signature cannot be differentiated from two or more reference patterns. The fault can be identified when α decreases below a user-specified threshold $\alpha_{\min}$.

$$State = \begin{cases} \text{Fault } Z & \left(D(\Sigma^X, \Sigma^Z) \le D^{\max}\right) \& \left(\alpha \le \alpha_{\min}\right) \\ \text{Unknown} & \text{Otherwise} \end{cases} \qquad \text{(Eq 4-11)}$$

An index called the maturity degree is also defined to measure the extent of progression of the fault at the time of detection. The maturity degree, Π, is the ratio between the number of clusters in Σ^{Z*} (the signature of the fault as synchronized with Σ^X) and that in Σ^Z (the complete fault signature), at the time of fault isolation.

If $D(\Sigma^X, \Sigma^Z) > D^{\max}$, $\forall Z \in \Psi$, the process can be said to be in a novel state. Some diagnosis support can be provided even in such cases by comparing the variable values between the online signal and the reference trajectory. The variable-residual, δ_{in}, measures the deviation between the current measurement x_i and the closest operating condition $m^{\exp}$ in the reference trajectory, Y, based on the set of neurons M^Y in the state-signature Σ^Y:

$$M^Y = \left\{ m_j \right\} \quad \text{s.t.} \quad u_{j,\Sigma_q^Y} = 1 \quad \forall j, q \qquad \text{(Eq 4-12)}$$

The reference operating condition $m^{\exp}$ that best corresponds to x_i is located from M^Y:

$$b_* = \arg\min_j \| x_i - m_j \|, \ j \in M^Y$$

$$m^{\exp} = m_{b_*} \qquad \text{(Eq 4-13)}$$

The variable-residual, δ_{in}, is then computed as the range normalized deviation between the current process measurement x_i and $m^{\exp}$:

$$\delta_{in} = (x_{in} - m_n^{\exp}),\ \forall n = [1,\ N] \qquad \text{(Eq 4-14)}$$

The variable-residuals are thus similar to the contribution plot in PCA and can serve as an aid for fault isolation as illustrated in the case studies. A fault rectification strategy oriented towards minimizing the residuals can then be synthesized.

4.6 Case Study 1: Disturbance Identification for Tennessee Eastman Process

Process Description:

The Tennessee Eastman (TE) plant (Downs and Vogel, 1993) is a popular test bed for process systems applications such as plant-wide control, optimization, predictive control, fault diagnosis and signal comparison. The TE process produces two products (G and H) and a byproduct (F) from reactants A, C, D, and E based on the following reactions:

$$A(g) + C(g) + D(g) \rightarrow G(liq)$$

$$A(g) + C(g) + E(g) \rightarrow H(liq)$$

$$A(g) + E(g) \rightarrow F(liq)$$

$$3D(g) \rightarrow 2F(liq)$$

The flow diagram of the process based on the control structure of McAvoy and Ye (1994) is shown in Figure 4-2. The process has five unit operations: a two-phase reactor, a product condenser, a flash separator, a recycle compressor, and a product stripper. There are altogether 22 continuous process measurements (Table 4-1), 12 manipulated variables and 19 composition measurements sampled less frequently.

Table 4-1: TE process measurements and their base value

Variable name	**Variable number**	**Base case value**	**Units**
A feed (stream 1)	XMEAS (1)	0.2505	kscmh
D feed (stream 2)	XMEAS (2)	3664.0	kgh^{-1}
E feed (stream 3)	XMEAS (3)	4509.3	kgh^{-1}
A and C feed (stream 4)	XMEAS (4)	9.3477	kscmh
Recycle flow (stream 8)	XMEAS (5)	26.902	kscmh
Reactor feed rate (stream 6)	XMEAS (6)	42.339	kscmh
Reactor pressure	XMEAS (7)	2705.0	kPa gauge
Reactor level	XMEAS (8)	75.0	%
Reactor temperature	XMEAS (9)	120.40	°C
Purge rate (stream 9)	XMEAS (10)	0.3371	kscmh
Product separator temperature	XMEAS (11)	80.109	°C
Product separator level	XMEAS (12)	50.000	%
Product separator pressure	XMEAS (13)	2633.7	kPa gauge
Product separator underflow (stream 10)	XMEAS (14)	25.160	m^3h^{-1}
Stripper level	XMEAS (15)	50.000	%
Stripper pressure	XMEAS (16)	3102.2	kPa gauge
Stripper underflow (Stream 11)	XMEAS (17)	22.949	m^3h^{-1}
Stripper temperature	XMEAS (18)	65.731	°C
Stripper steam flow	XMEAS (19)	230.31	kgh^{-1}
Compressor work	XMEAS (20)	341.43	kW
Reactor cooling water outlet temperature	XMEAS (21)	94.599	°C
Condenser cooling water outlet temperature	XMEAS (22)	77.297	°C

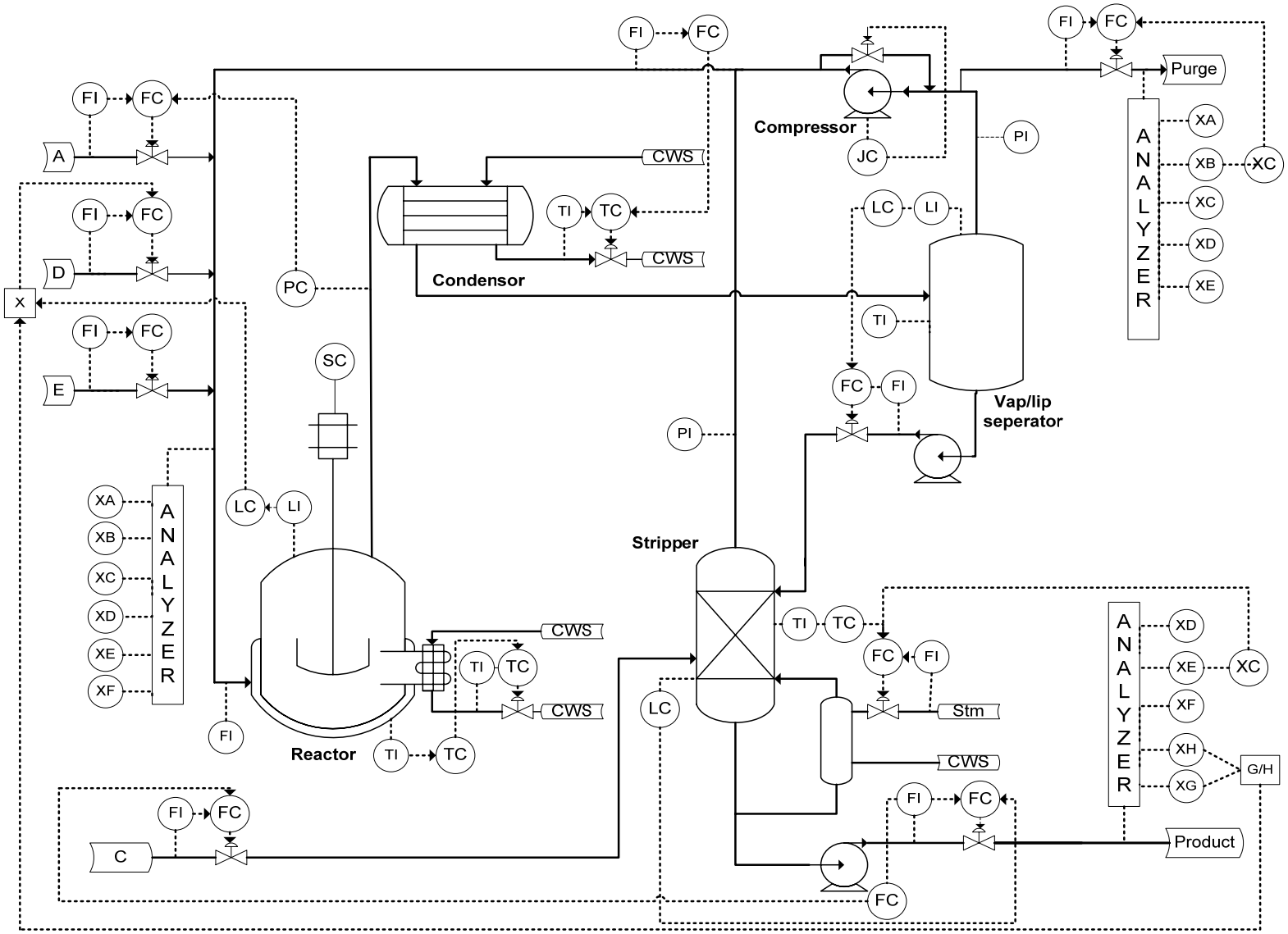


Figure 4-2: Flowsheet of Tennessee Eastman process

Table 4-2: Disturbance profile for TE process resulting from changes in base values of : (a) A feed during XD1; (b) reactor pressure during XD2; (c) reactor level during XD3; (d) reactor temperature during XD4; (e) compressor work during XD5

	Target	**Time (min)**	**Target**	**Time (min)**	**Target**	**Time (min)**	**Target**	**Time (min)**
(a)								
XD1-A	1.20*Base value	180	1.40*Base value	190	1.60*Base value	200	1.0*Base value	780
XD1-B	1.15*Base value	240	1.35*Base value	254	1.55*Base value	268	1.0*Base value	900
XD1-C	1.10*Base value	300	1.30*Base value	318	1.50*Base value	336	1.0*Base value	1020
(b)								
XD2-A	1.03*Base value	180	1.05*Base value	190	1.07*Base value	200	1.0*Base value	1020
XD2-B	1.025*Base value	240	1.045*Base value	254	1.065*Base value	268	1.0*Base value	1080
XD2-C	1.02*Base value	300	1.04*Base value	318	1.06*Base value	336	1.0*Base value	1200
(c)								
XD3-A	1.05*Base value	180	1.10*Base value	190	1.15*Base value	200	1.0*Base value	780
XD3-B	1.045*Base value	240	1.09*Base value	254	1.135*Base value	268	1.0*Base value	900
XD3-C	1.04*Base value	300	1.08*Base value	318	1.12*Base value	336	1.0*Base value	1020
(d)								
XD3-A	1.05*Base value	180	1.10*Base value	190	1.15*Base value	200	1.0*Base value	780
XD3-B	1.045*Base value	240	1.09*Base value	254	1.135*Base value	268	1.0*Base value	900
XD3-C	1.04*Base value	300	1.08*Base value	318	1.12*Base value	336	1.0*Base value	1020
(e)								
XD5-A	0.95*Base value	180	0.90*Base value	190	0.85*Base value	200	1.0*Base value	780
XD5-B	0.955*Base value	240	0.91*Base value	254	0.865*Base value	268	1.0*Base value	900
XD5-C	0.96*Base value	300	0.92*Base value	318	0.88*Base value	336	1.0*Base value	1020

In this section, the proposed SOM-based FDI method is tested for online disturbance identification on the Tennessee Eastman (TE) industrial challenge problem (Downs and Vogel, 1993). The 22 continuous process measurements (shown in Table 4-1) are used to identify unknown process disturbances online. Five disturbances, henceforth referred to as XD1 to XD5, that affect the *A* feed flowrate, reactor pressure, reactor level, temperature, and compressor work are considered. These are representative of setpoint changes or servo control problem in industrial process operations. Three different instances are created for each disturbance, i.e., for fault class XD2, XD2-A to XD2-C are created with different start times, duration, and magnitude. The profiles for each of the disturbance are listed in Table 4-2. In general, the five disturbance classes have very similar effect and are difficult to distinguish (Srinivasan and Qian, 2005). They therefore serve as good illustration for the proposed method.

In this case study, XD1-B, XD2-B, XD3-B, XD4-B, and XD5-B are used as the references for the five disturbances. The reference disturbances are first autoscaled and used to train a SOM. The fully trained SOM contains 756 map units. The quantization error, E^q, during training is observed to be 1.072, or about 0.8% indicating that the SOM provides a good representation of the training data. *k*-means (K=50) is then used to cluster the SOM reference vectors. Data from all five reference runs are projected to construct the reference state-signatures. Three neuronal-clusters, namely, S_{19}, S_{21}, and S_{44} correspond to nominal steady state operation (XD0-B). All other clusters correspond to various disturbance states. State-signatures were then generated for each disturbance by projecting them to the SOM. The fully trained SOM and the state-signatures are then used for fault detection and diagnosis for the testing data.

Scenario 1: Change in Reactor Pressure

During Run-4 the reactor pressure is increased from the base case value of 2705 to 2867kPa gauge in three steps, starting from 300 min, as shown in Table 4-2. The signals of Run-4 are shown in Figure 4-3 (labeled as XD2-C). After the process settles at the new steady-state, the inverse change, decreasing the reactor pressure is introduced at t=1200 min and the process is allowed to return to a steady-state. Initially (t<300min) the process remains in a normal state, and the BMUs are located in the three nominal neuronal-clusters – S_{19}, S_{21}, and S_{44}. After the fault occurs, the signals of Run-4 start to exhibit abnormality and are projected to neuronal-cluster S_{25} at t=312 min (see Figure 4-4). Since S_{25} is not in the nominal neuronal-cluster set, a fault is flagged. This neuronal-cluster and all subsequent ones form the real-time state-signature and are used for disturbance identification.

The state-signatures of XD1, XD2, and XD5 have element S_{25} in their state-signature, so the distance to them is small and all of them are initially identified as fault candidates. At t=354 min, a new neuronal cluster-hit occurs on S_{27}. The online state-signature at this time becomes $\Sigma^{Run4} = \left\{ S_{25}^{Run4}, S_{27}^{Run4} \right\}$ which matches the reference for XD2 exactly but has large distances with XD1 and XD5. XD2 is clearly distinguishable from the others and α drops to 0 ($\alpha^{\min} = 0.75$). The fault is hence isolated as XD2 at t=354min. A low maturity value of $\Pi_{354\,\min}$=0.077 at the time of fault identification indicates that the fault has been identified at an early stage of its development. Table 4-3 shows a part of the dissimilarity matrix between XD2 and this run. As can be observed there, there are minor run-to-run variations in the online state-signature (neuronal-cluster

42 inserted before cluster 40) as well as oscillatory behavior (neuronal-clusters 8 and 16), but the proposed method efficiently discounts them.

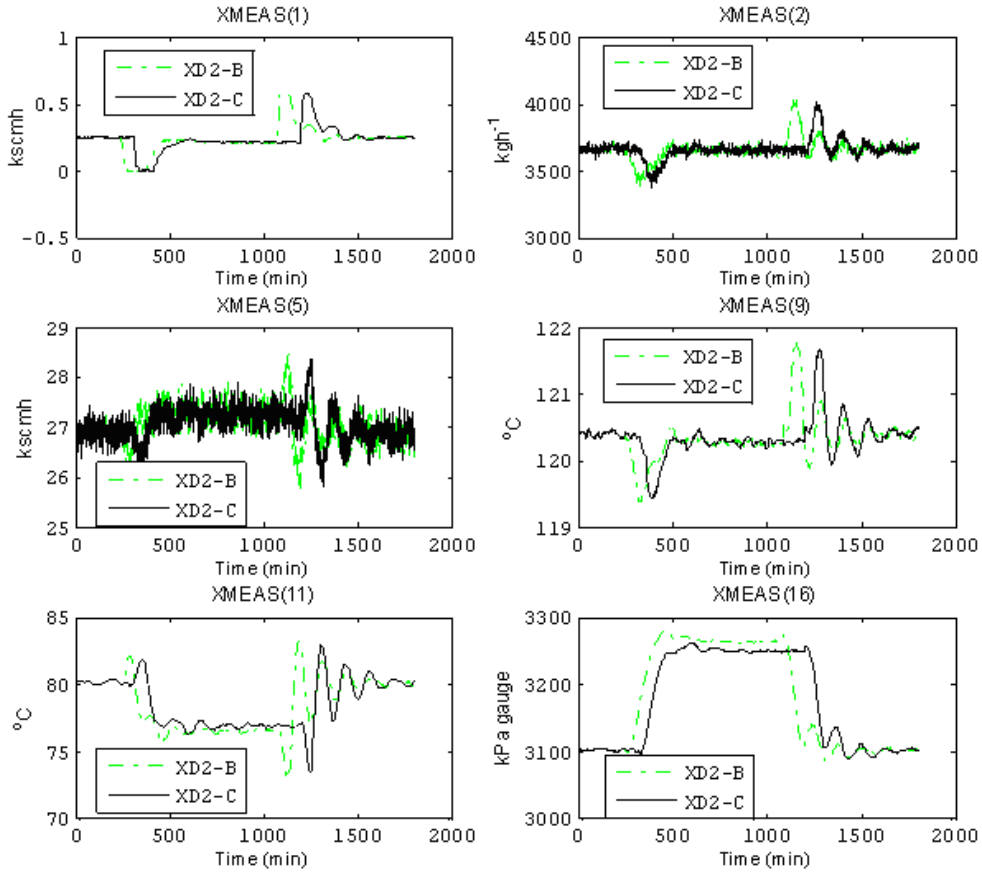


Figure 4-3: Process signals during runs XD2-B and Run-4 (XD2-C) in TE case study

Table 4-3: Dissimilarity matrix between state-signatures of Run-4 and XD2-B

$\Sigma_q^{XD4} \setminus \Sigma_r^{Run4}$	**25**	**27**	**…**	**28**	**2**	**8**	**16**	**8**	**16**	**8**	**16**	**22**	**36**	**11**	**1**	**42**	**40**	**7**	**2**	**8**
25	**0.000**	0.267	**…**	2.325	2.530	2.744	2.943	3.160	3.360	3.577	3.777	4.109	4.261	4.480	4.749	4.924	5.084	5.234	5.368	5.582
27	0.267	**0.000**	**…**	2.058	2.263	2.477	2.676	2.893	3.093	3.310	3.510	3.842	3.994	4.213	4.482	4.657	4.817	4.967	5.101	5.315
…	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**	**…**
28	0.324	0.591	**…**	**0.000**	**0.205**	0.419	0.618	0.419	0.618	0.419	0.618	0.951	1.103	1.322	1.591	1.766	1.925	2.075	2.210	2.424
8	0.157	0.424	**…**	0.283	0.214	**0.205**	0.405	0.205	0.405	0.205	0.405	0.737	0.889	1.108	1.377	1.552	1.711	1.861	1.996	2.210
16	0.317	0.566	**…**	0.482	0.414	0.405	**0.205**	**0.405**	**0.205**	**0.405**	**0.205**	0.537	0.689	0.908	1.177	1.352	1.512	1.662	1.796	2.010
22	0.154	0.421	**…**	0.815	0.746	0.609	0.537	0.400	0.537	0.400	0.537	**0.205**	0.357	0.576	0.845	1.020	1.180	1.330	1.464	1.678
36	0.275	0.329	**…**	0.967	0.898	0.761	0.689	0.552	0.689	0.552	0.689	0.357	**0.205**	0.424	0.693	0.868	1.028	1.178	1.312	1.526
11	0.262	0.529	**…**	1.186	1.117	0.980	0.908	0.771	0.908	0.771	0.908	0.576	0.424	**0.205**	0.474	0.649	0.808	0.958	1.093	1.307
1	0.416	0.683	**…**	1.455	1.386	1.249	1.177	1.040	1.177	1.040	1.177	0.845	0.693	0.474	**0.205**	0.380	0.540	0.689	0.824	1.038
40	0.329	0.596	**…**	1.736	1.633	1.531	1.459	1.322	1.459	1.322	1.459	1.127	0.975	0.756	0.487	**0.365**	0.380	0.530	0.664	0.878
10	0.449	0.716	**…**	1.943	1.840	1.737	1.666	1.529	1.666	1.529	1.666	1.334	1.182	0.962	0.694	**0.572**	0.572	0.674	0.809	1.023
40	0.329	0.596	**…**	2.192	2.088	1.986	1.914	1.777	1.914	1.777	1.914	1.582	1.430	1.211	0.942	0.820	**0.572**	0.705	0.839	1.053
7	0.238	0.505	**…**	2.341	2.238	2.135	2.064	1.927	2.064	1.927	2.064	1.732	1.580	1.360	1.092	0.970	0.722	**0.572**	0.706	0.920
2	0.235	0.502	**…**	2.361	2.341	2.270	2.198	2.061	2.198	2.061	2.198	1.866	1.714	1.495	1.226	1.104	0.856	0.706	**0.572**	0.786
8	0.157	0.424	**…**	2.437	2.555	2.341	2.412	2.198	2.279	2.198	2.279	2.080	1.928	1.709	1.440	1.318	1.070	0.920	0.786	**0.572**
22	0.154	0.406	**…**	2.464	2.669	2.536	2.607	2.393	2.474	2.393	2.474	2.275	2.123	1.904	1.635	1.513	1.265	1.115	0.981	0.767
34	0.218	0.485	**…**	2.442	2.647	2.722	2.793	2.579	2.660	2.579	2.660	2.461	2.309	2.090	1.821	1.699	1.451	1.301	1.167	0.953
42	0.317	0.584	**…**	2.504	2.675	2.887	2.958	2.744	2.825	2.744	2.825	2.626	2.474	2.255	1.986	1.821	1.616	1.466	1.332	1.118
7	0.238	0.505	**…**	2.453	2.632	2.846	3.046	2.846	2.995	2.846	2.995	2.796	2.644	2.425	2.156	1.991	1.786	1.616	1.502	1.288
34	0.218	0.485	**…**	2.541	2.652	2.866	3.066	2.866	3.066	2.866	3.066	2.971	2.819	2.600	2.331	2.166	1.961	1.791	1.677	1.463
44	0.175	0.442	**…**	2.444	2.648	2.842	3.041	2.842	3.041	2.842	3.041	3.082	2.930	2.711	2.442	2.277	2.072	1.902	1.788	1.574

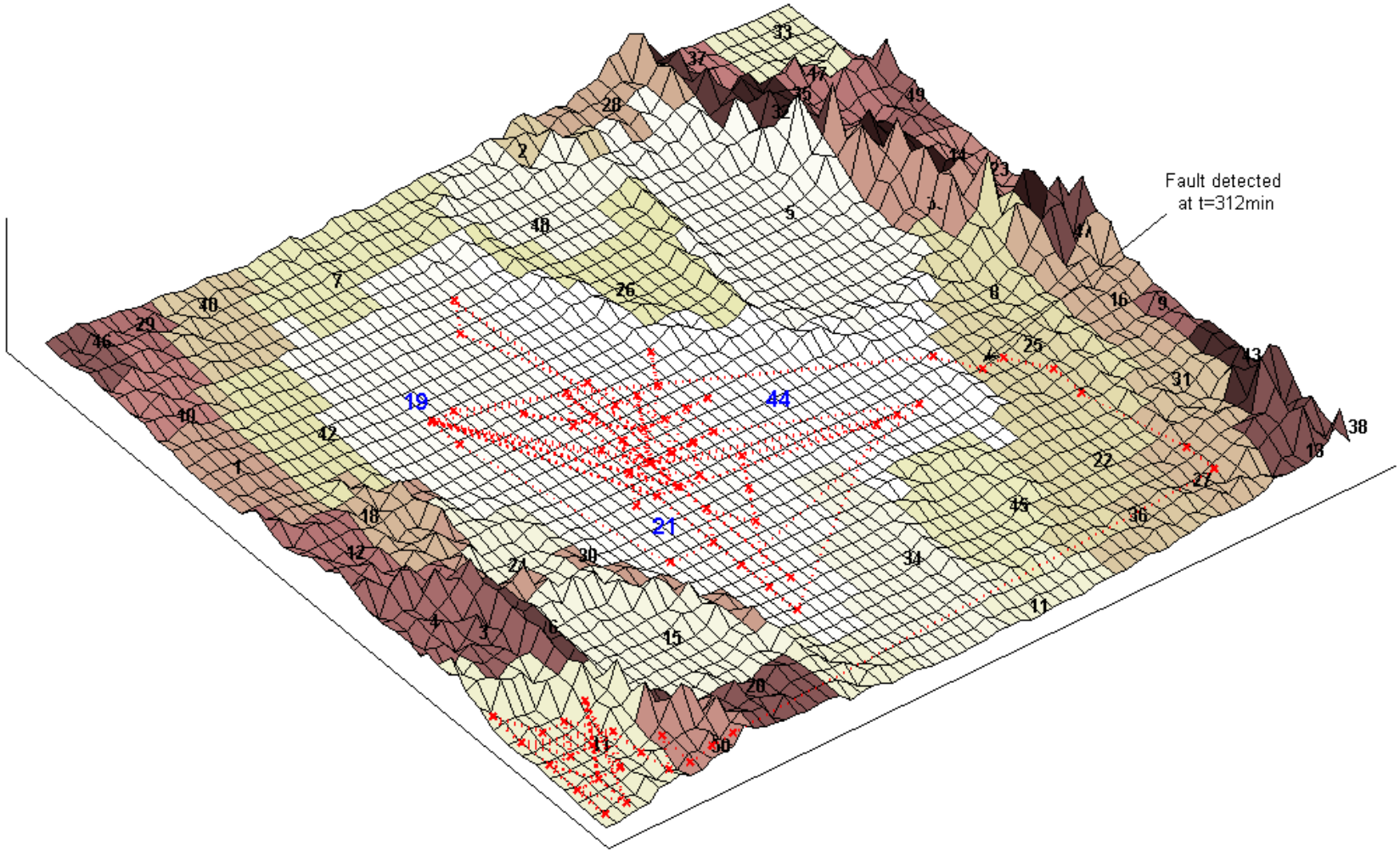


Figure 4-4: Operating profile of Run-4 in TE case study from *t*=1min to *t*=1200min

Scenario 2: Change in Reactor temperature

During Run 7, the reactor temperature was increased from its base value of 120.4 ºC to 134.85ºC starting at t=180 min. At t=185 min, the hit on the SOM deviates from the nominal neuronal-cluster S_{21} to S_{32} indicating an abnormality. Since neuronal-cluster S_{32} is unique to disturbance XD4 (in its state-signature), this disturbance is identified immediately.

Similar analysis was also performed for the rest of the runs as summarized in Table 4-4. In all runs, the average quantization error observed is 0.9313 or an average of 0.71% per variable, and the maximum is 6.4714 or 4.9% indicating that the SOM was adequately representative for all cases. All faults are successfully detected and diagnosed with an average detection delay of 14.6 min and a diagnostic delay of 32.3 min. The distance observed at the end of the runs for all cases are shown in Table 4-5. The proposed method is able to identify the actual fault despite significant run-to-run variations including oscillations. The results from the classical Smith-Waterman algorithm are presented for comparison in Table 4-6. As can be seen, Runs 1 and 2 are indistinguishable based on the classical approach but are clearly separated by the proposed method ($\alpha << \alpha^{min}$). In all cases, the distance calculated by the proposed method is smaller or equal to that by the classical approach. Another major advantage of the proposed method is its computational speed. The computational time for the complete analysis of each sample is less than 0.1s (on a Pentium® Xeon™ 3.0 GHz CPU), making it suitable even for large industrial-scale case studies.

Table 4-4: Online fault diagnosis results for Tennessee Eastman Process

	Fault Detection				**Fault Identification**			
	Fault Introduced (min)	**Fault detected (min)**	**Detection delay (min)**	**Fault candidate**	**Time fault Diagnosed (min)**	**Diagnosis Delay (min)**	**Fault Maturity**	**Best-matching reference**
Run-1	180	209	29	1,2,3,5	272	63	0.300	XD1
Run-2	300	342	42	1,2,3,5	402	60	0.300	XD1
Run-3	180	184	4	1,2,5	224	40	0.077	XD2
Run-4	300	312	12	1,2,5	354	42	0.077	XD2
Run-5	180	183	3	1,2,3	192	9	0.043	XD3
Run-6	300	304	4	1,2,3	322	18	0.043	XD3
Run-7	180	185	5	4	185	0	0.034	XD4
Run-8	300	305	5	4	305	0	0.034	XD4
Run-9	180	197	17	1,2,3,5	242	45	0.042	XD5
Run-10	300	325	25	1,2,3,5	371	46	0.042	XD5
		Avg delay	**14.6**		**Avg delay**	**32.3**		

Table 4-5: Distance across all runs for TE process

	XD1	XD2	XD3	XD4	XD5	α
Run-1	**0.2589**	3.8514	3.4275	4.7354	3.6363	0.0755
Run-2	**0.2005**	5.3449	4.9934	6.5299	5.0009	0.0401
Run-3	6.8863	**2.2394**	5.5613	9.8257	5.3012	0.4224
Run-4	6.2306	**1.0292**	4.5460	6.3150	4.7808	0.2264
Run-5	11.6967	10.4014	**1.9361**	15.3684	9.6389	0.2008
Run-6	11.1057	9.7793	**2.1498**	10.7938	9.2747	0.2318
Run-7	7.6900	6.2753	6.3665	**0.2376**	7.2001	0.0379
Run-8	8.7610	7.0106	6.9907	**0.7839**	7.9564	0.1121
Run-9	10.1245	10.1992	9.9775	17.5436	**3.1151**	0.3122
Run-10	10.6604	10.1364	9.8865	14.6375	**4.2110**	0.4259

Table 4-6: Distance across all runs based on the classical Smith-Waterman algorithm

	XD1	XD2	XD3	XD4	XD5	α
Run-1	**2.9746**	3.8807	3.5087	4.7354	3.6505	0.8477
Run-2	**4.8179**	5.8806	5.4901	6.5299	5.5651	0.8776
Run-3	8.4095	**4.9950**	8.5673	9.8257	7.3123	0.6831
Run-4	6.5026	**2.2223**	5.6519	6.7388	5.4776	0.4057
Run-5	14.3717	14.1112	**5.6357**	15.3684	12.3101	0.4578
Run-6	11.8885	10.7510	**3.7666**	11.6408	10.2432	0.3677
Run-7	7.6900	6.2753	6.3665	**0.5682**	7.2001	0.0905
Run-8	8.7610	7.0106	6.9907	**1.4451**	7.9564	0.2067
Run-9	14.7669	15.0777	15.3282	17.7383	**9.0327**	0.6117
Run-10	12.5298	11.8390	12.6871	15.1359	**8.7695**	0.7407

4.7 Case Study 2: Fault diagnosis during startup of a distillation unit

In this section, the proposed methodology is tested with a lab-scale distillation unit. This case study is the same as the one described in Chapter 3.3 of this dissertation. The SOM created for this case-study contains 60 neuronal-clusters. A reference training data for normal startup, *R*, is projected to SOM to construct the dictionary state-signature, Σ^Y, and the reference dwell-time recorded. The state-signature for the normal start-up was constructed and used for subsequent monitoring of the transition. Fault signatures were also constructed using data for the ten failures. These reference state-signatures were then used for process supervision. Three scenarios are discussed in detail.

Scenario 1: Reboiler power fault

In one run, a reboiler power failure was induced at t=10s, resulting in long heating time. The startup becomes unsuccessful at step 7 when the feed pump is activated as there is not enough heat to evaporate the feed. The online samples from this run are projected onto the SOM and the cluster-hits identified. The fault is successfully detected at t=130s when the state-signature deviates from neuronal-cluster S_3 to S_{38}. The variable-residuals at time of fault detection are shown in . It is apparent that the reboiler power has significantly deviated from its profile during normal start-up. A direct time-based signal comparison would lead to erroneous conclusions (significant residuals for Tray 4 temperature) since a different state (boiling phase) of the reference trajectory would then be used as the basis instead of the reboiler heating phase of the current run. The proposed state-signature comparison method also identifies the right fault (DST01) based on Eq (4-11).

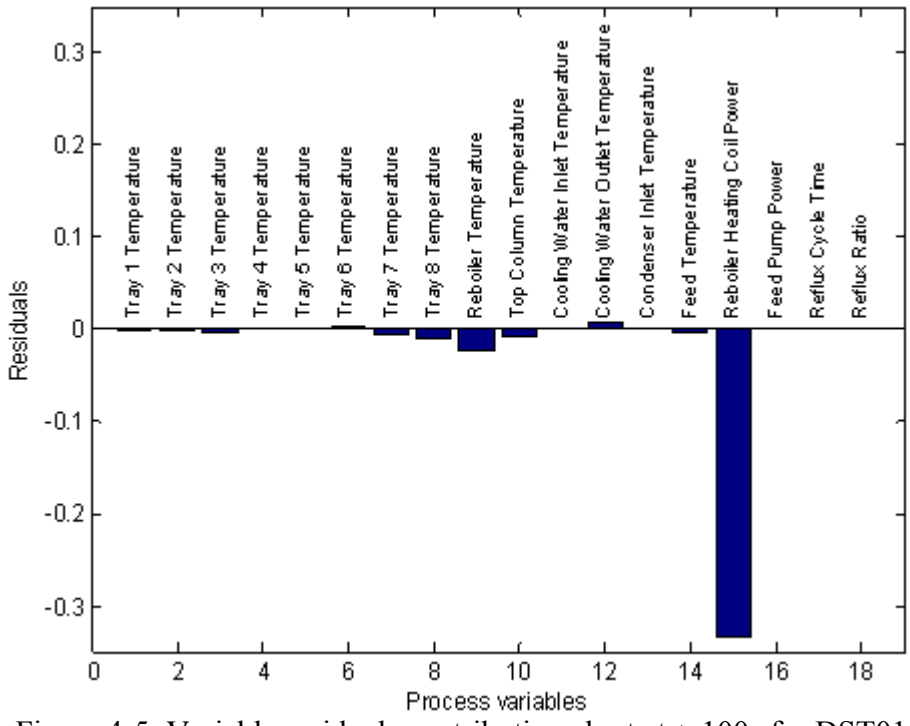


Figure 4-5: Variable residuals contribution chart at t=100s for DST01

Scenario 2: Low cooling water flow and feed pump malfunction

In this run, a complex fault arising from a low flow of condenser cooling water and feed pump malfunction is introduced. The process signals from this run are shown in Figure 4-6. The low flow of cooling water is first introduced at t=10s, but cannot be directly observed since the heat exchanger is not equipped with a flow sensor. The symptoms of the fault start around t=2990s when the outlet temperature of the condenser cooling water – T12 rises above its nominal value. The first alarm is flagged at t=3040s when abnormal state-sequence evolution (hit in neuronal-cluster 7) is observed (see Figure 4-7). The variable-residuals at t=3040s, shown in Figure 4-8a, readily depict the symptoms of the abnormality. The state-signature comparison also reveals a high similarity with DST10. A second fault was introduced at t=4000s when feed loss occurs due to pump failure. This is also immediately discernible from the variable-residuals at t=4020s (see Figure 4-8b) which shows both the differences.

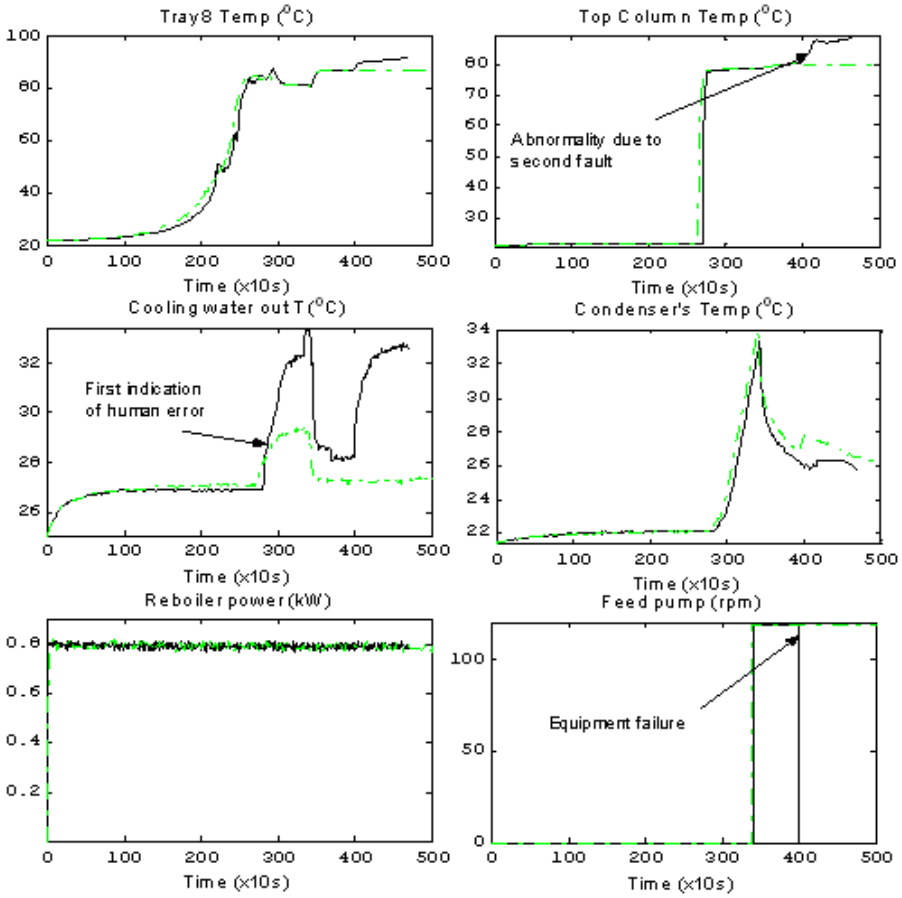


Figure 4-6: Process signals for Run-10 in distillation unit case study

Scenario 3: High reflux ratio

In this run, a high reflux ratio occurs due to human error at t=3540s. This leads to a reduction in product yield and increases the load on the reboiler. The fault is immediately detected at t=3540s while other run-to-run differences are ignored as can be seen in Figure 4-9.

Similar studies were conducted for the other faults as well. All faults were successfully detected and diagnosed as summarized in Table 4-7. Most faults are isolated quickly with an average diagnosis delay of t=137s (from time of fault introduction). DST04, and DST08 have similar responses initially and require a longer duration to be distinguished.

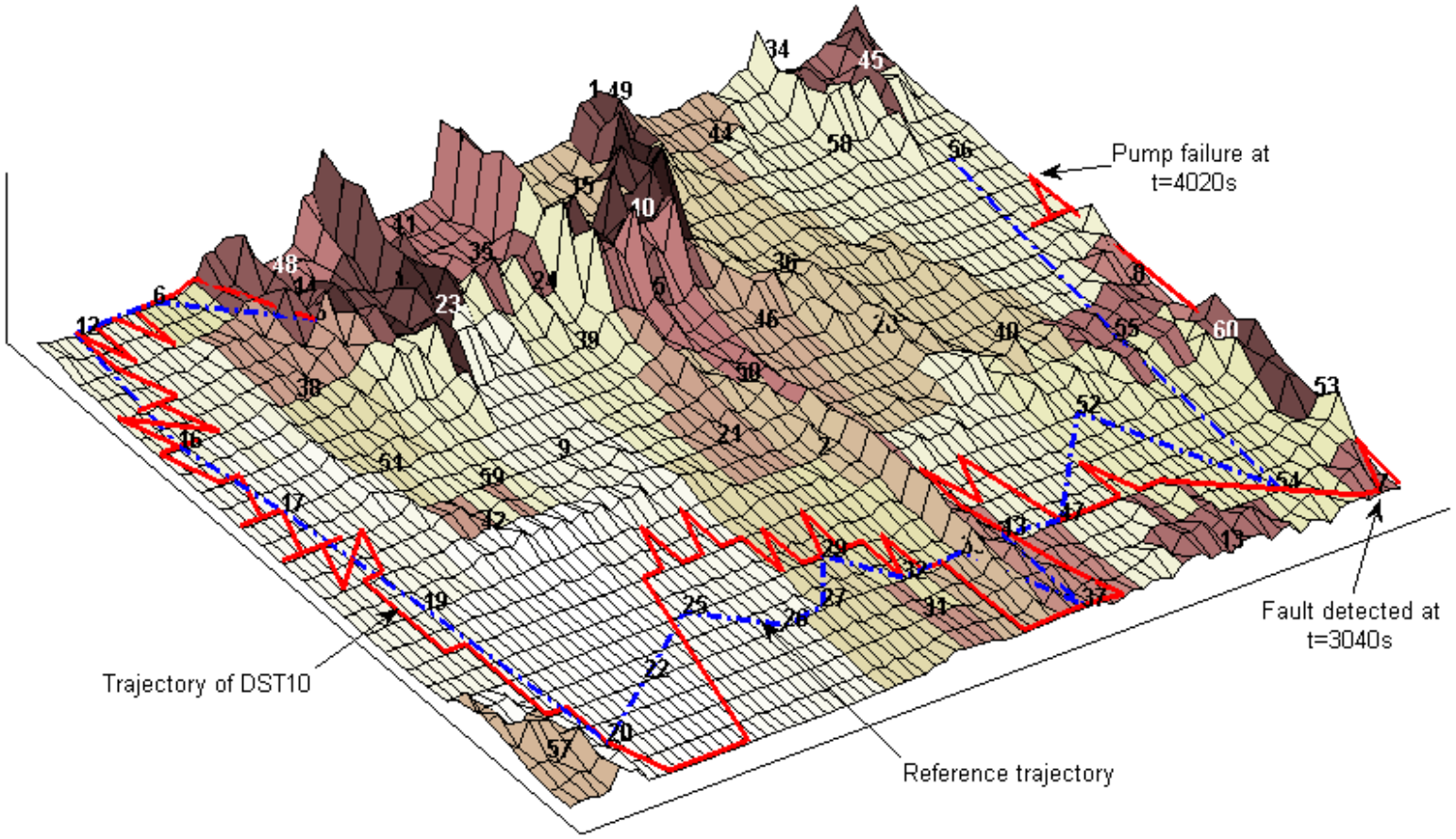
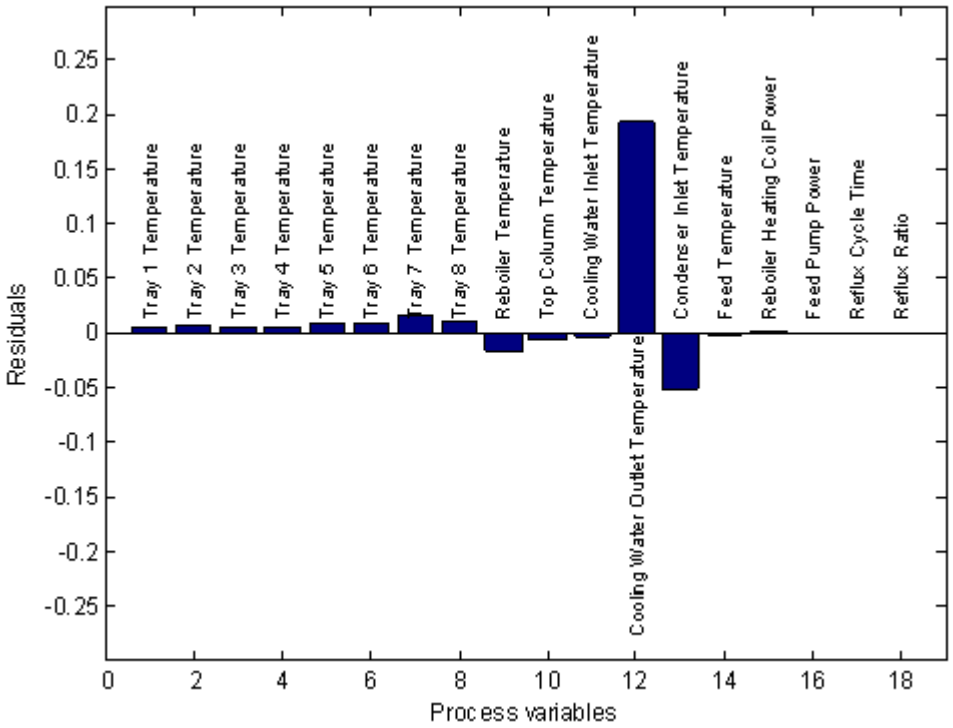


Figure 4-7: Operating trajectory of Run-10 in distillation unit case study



(a) Variable residuals contribution chart at t=3040s for DST10

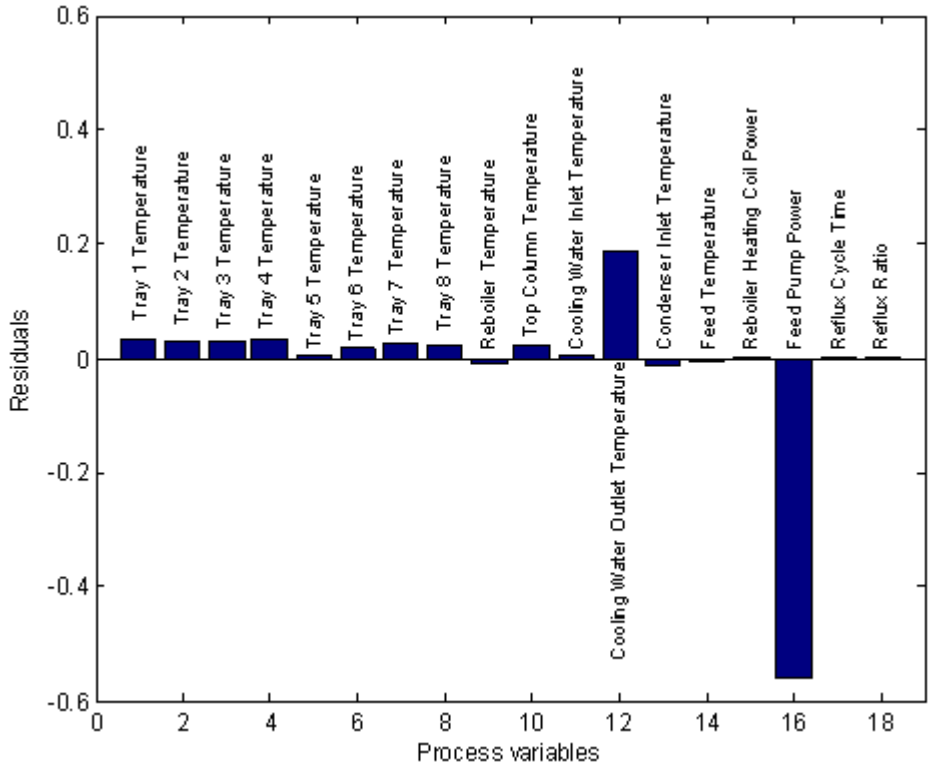


Figure 4-8: (b) Variable residuals contribution chart at t=4020s for DST10

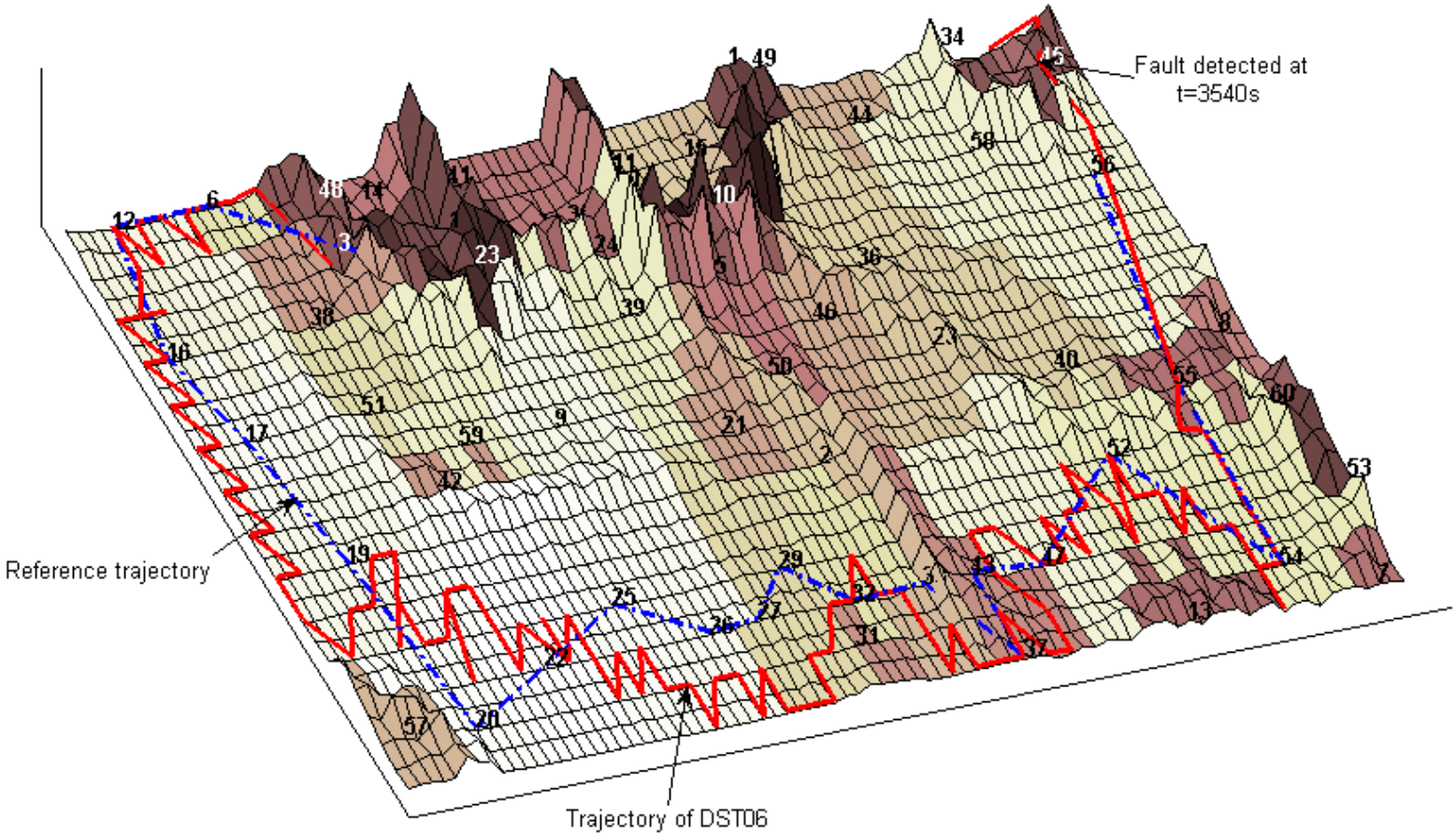


Figure 4-9: Operating trajectory of Run-6 in distillation unit case study

Table 4-7: Fault diagnosis results for distillation unit startup case study

Scenario #	**Disturbance**	**Time fault introduced (x10s)**	**Time fault detected (x10s)**	**Detection Delay (x10s)**	**Fault Candidate (DST)**	**Time fault Diagnosed (x10s)**	**Diagnosis Delay (x10s)**	**Fault maturity (%)**
DST01	Reboiler power low	1	13	12	1	13	0	0.07
DST02	Reboiler power high	1	18	17	2	18	0	0.16
DST03	Feed pump high	359	372	13	3	372	0	0.33
DST04	Feed pump low	430	459	29	4,8	462	3	0.60
DST05	Tray Sensor T6 fault	425	426	1	5	426	0	1.00
DST06	Reflux ratio high	353	354	1	6	354	0	0.50
DST07	Reflux ratio low	345	347	2	7	347	0	1.00
DST08	Bottom valve	420	470	50	4,8	473	3	0.60
DST09	Cooling water	1	1	0	9	1	0	0.04
DST10	Low cooling water flow and feed pump malfunction	299	304	6	10	304	0	0.33
			Avg Delay	**13.1**		**Avg Delay**	**0.6**	

4.8 Robustness analysis

In the proposed approach, the number of clusters, K, is the main tuning parameter that determines performance. The effect of K on selectivity – the speed of fault identification and sensitivity – Type-I errors – was studied. A large K results in more neuronal-clusters and hence improves the ability to represent the process operation at a finer level of resolution. However, the SOM would also becomes more sensitive to minor deviations from normal operations, and could result in false alarms. Using too small a K could render some faults undetectable as the neurons correspond to abnormal operating regions would be amalgamated with the clusters for normal operation, leading to false negatives. To evaluate the effect of K on the results, the previously trained SOM described above, was partitioned with various K values. Table 4-8 hows the results of the study. For a wide range of values, $K \in [40, 70]$, all the faults can be well differentiated although the average detection delay decreases with increasing K.

Table 4-8: Sensitivity studies for distillation unit startup case study based on various K

K	DST Differentiable	Avg Detection Delay	Misclassification Rate
10	7	1261s	0.02%
25	8	185s	0.00%
40	10	132s	0.00%
55	10	127s	0.00%
70	10	111s	0.02%
90	10	104s	2.80%

4.9 Conclusions and Discussion

Online tracking of the process operating state is essential to detect and identify the occurrence of faults – known or novel. For transient states, the temporal evolution of the process has to be considered. The complexity stems from the need to compare

large-scale, multivariate, temporal signals in real-time. In this chapter, the SOM is used to reduce the dimensionality of the measurement space. Each SOM neuron and neuronal-cluster is a landmark in the multivariate measurement space – the latter provides a higher granularity and is hence more robust to run-to-run and operator-to-operator variations. The one-dimensional string of neuronal-clusters can adequately serve as a signature of the transition state. With this representation, the transition monitoring and diagnosis problem becomes one of sequence comparison. A method to compare two state-signatures using dynamic programming has also been developed. Deviations during the execution of a known transition are detected when the distance between the reference signal and the online observations exceeds a pre-specified threshold. Two approaches have also been proposed for identifying the disturbance. (1) similarity with known faults is determined by comparing the deviant portion of the online state-signature with the library of reference signals. Known faults can be identified through this approach when the online state-signature matches unequivocally to one reference signal. (2) For novel faults where no similar sequence is available in the library, residuals can be calculated at the variable-level, so that the differences are localized and the fault can be diagnosed by the operator. Application of the proposed approach to two case studies – the Tennessee Eastman challenge problem and the startup of a distillation unit – illustrate its benefits method in diagnosing faults during multi-state, temporal process operations.

The proposed methodology shares some similarities with other data-driven approaches for online state identification. Similar to the SOM-based methodology, qualitative trend analysis also abstracts the signal into a small set of alphabets and can naturally discount time variations between the reference and the online signals. In the case of the latter the alphabets are based on the first (and second) derivatives of the

signal. The string of alphabets that represent the transition therefore dictates the gross profile (sets of shapes) for each variable. The proposed approach however decomposes the temporal signal based on magnitude ranges. The key advantages of the proposed approach are that (1) the granularity of the representation, i.e., the alphabet set (the set of neuronal-clusters) is adaptable to the process through the user-defined *K*. The method is robust for a wide-range of *K* values as shown in Section 4.8, yet fault detection with lesser delay can be obtained using a larger *K*. (2) the trend-analysis approach requires a separate logic for specifying the dictionary pointer which governs the correspondence between the reference and the online trends. Here, the dynamic programming approach obviates this step.

Signal comparison based strategies such as singular point augmented time warping and dynamic locus analysis evaluate the signals directly, without abstraction. The main advantage of the direct signal comparison is the speed of detection of deviations. There are three main shortcomings of the direct comparison – (1) the inherent uni-variate nature – this is partially addressed by the dynamic locus analysis of Srinivasan and Qian (2006), (2) the computational complexity of time warping in real-time, which is also partially addressed by the Singular point based simplification strategy of Srinivasan and Qian (2005;2007), and (3) the inability to address oscillatory signals where a one-to-one mapping between the reference and the real-time signals cannot be expected. The singular points use uni-variate landmarks to simplify the warping in contrast to the proposed approach which uses multivariate landmarks to abstract the signals and simplify the comparison. While this leads to an increase in the detection and diagnosis delay, the major advantage of the proposed method is computational efficiency and the inherent ability to deal with oscillatory signals.

Also, since each neuron on SOM corresponds to a certain process conditions, the state-signature is analogous to a multi-model representation of the process with the evolution of the neuronal cluster-hit during the transition imparting the model switching, *i.e.*, as the process evolves, a new neuron or neuronal-cluster, closer to the new operating regime, serves as the reference model. The proposed SOM-based approach is computationally efficient and suitable for real-time application even in large industrial case studies.

Nomenclature

Indices

i,p	sample
j	neuron
k	cluster
n	variable
q,r	state-signature

Parameters

I	total number of samples (rows) in X
K	total number of neuronal-clusters
N	total number of variables (columns) in a multivariate data
$Q,,R$	total number of state-signatures in a sequence
W	total number of fault classes available in database Ψ

Variables

b_i	best matching unit on SOM for x_i
c_k	centroid of k^{th} cluster
$D(\Sigma^X, \Sigma^Y)$	distance between Σ^X and Σ^Y
$\tilde{D}(\Sigma^X, \Sigma^Y)$	distance between Σ^X and Σ^Y
$D^{\max}$	user defined threshold to detect abnormal process state
E_i^q	quantization error for sample x_i
H	dissimilarity matrix used for synchronizing two sequences
m_j	the j^{th} neuron on SOM
m_{b_i}	reference vector of the BMU observed for sample x_i
$m^{\exp}$	reference operating condition for a sample

M^R	neurons identified for a reference transition
M^{SOM}	a self-organizing map model $M^{SOM} = \{m_1, ..., m_j, ..., m_J\}$
S_k	k^{th} neuronal-cluster
$\hat{S}_i$	state-cluster observed from x_i $\hat{S} = \{\hat{S}_1, ..., \hat{S}_i, ..., \hat{S}_I\}$
t_r	the dwell-time observed for the r^{th} element of a fault-signature
u_{jk}	1 if the class membership of neuron j equal to k
x_i	i^{th} sample in $X = \{x_{i1}, ..., x_{in}, ..., x_{iN}\}$
X	multivariate data $X = \{x_1, ..., x_i, ..., x_I\}^T$
Σ_r^X	the rth element of a fault-signature generated from data X
Σ_q^Y	the q^{th} element of a fault-signature generated from data Y
α_{min}	threshold for inseparability between two faults
δ_{in}	variable-residual for n^{th} variable in x_i
Π	maturity degree of a fault
Ψ	fault database containing data from various faults

Chapter 5 Adjoined Dynamic Principal Components Analysis for Transition Monitoring

5.1 Introduction

Much of the statistical process control (SPC) literature has focused heavily on methods for handling data sampled from normal distributions. Current approaches for process monitoring based on the popular Principal Components Analysis techniques also assume that the process data follows a normal distribution. This assumption is not valid for most batch/transient operations. As an illustrative example, consider the univariate signal *S* shown in Figure 5-1a that arises from a transient operation. *S* can be classified into modes (*M*) and transitions (*T*). A mode corresponds to the continuous operation of the unit and a fixed flowsheet configuration, while a transition corresponds to discontinuities in the plant operation (Srinivasan, *et al.,* 2004). Three modes $M \in \{M1,\ M2,\ M3\}$ and two transitions $T \in \{T1\ ,T2\}$ can be identified in *S*. Figure 5-1b shows the result of a distribution test conducted on *S*. The data distribution of *S* fits neither a bell curve normal distribution, nor a more complex nonparametric distribution. There are two implications when the normal distribution assumption is not satisfied. First, the monitoring limits constructed using *SPE* and T^2 are prone to Type-II errors (false negatives) as the limits for the monitoring statistics cover a possibly abnormal operating region, e.g., portion of *S* at $t \in [40\ ,70]$. Second, and perhaps the more common scenario, is that a number of normal samples are not covered with in the monitoring limits on both sides of *S*, leading to Type-I errors (false positives), for

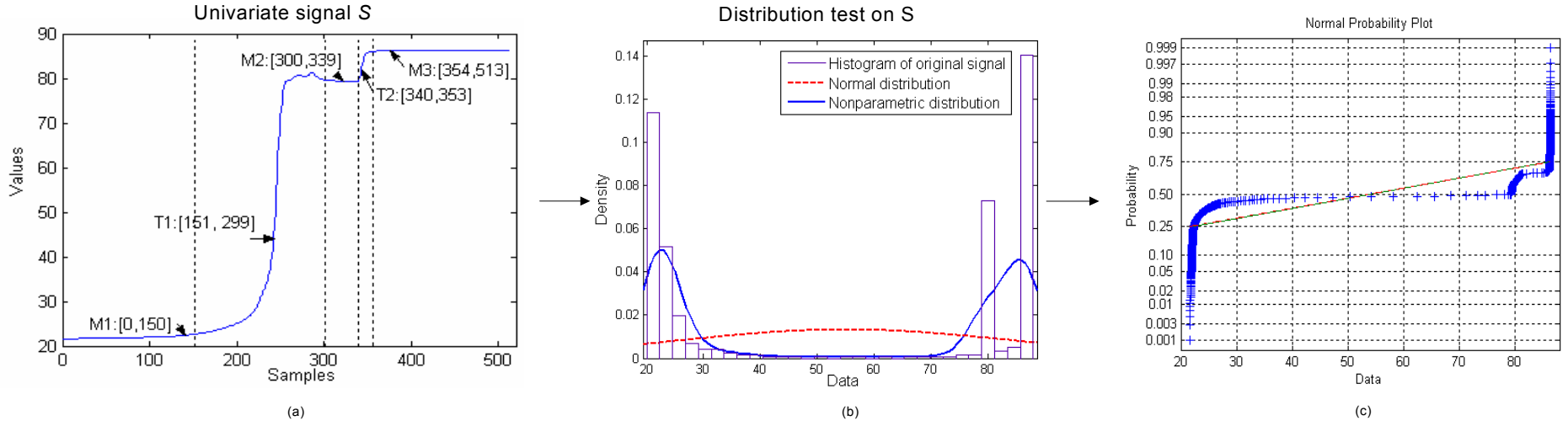


Figure 5-1: (a) A typical univariate signal, S, from a transient operation. (b) Probability density test on S. (c) Normal probability plot for S

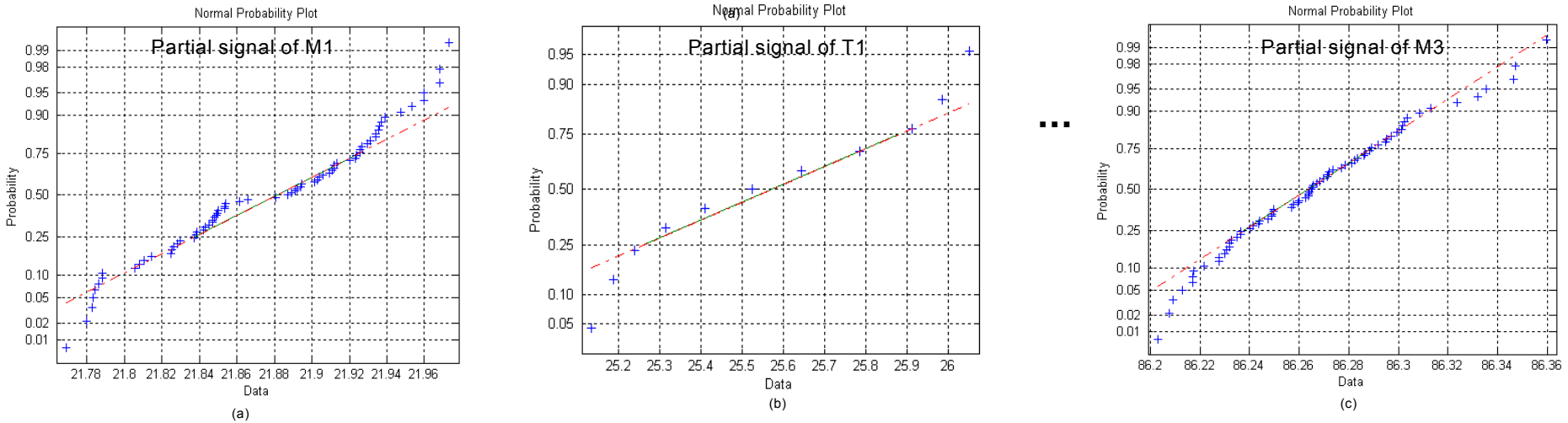


Figure 5-2: Normal probability plot for different segment of S

example, see S in $t \in [20\,,\,28]$, and $t \in [75\,,\,100]$. Even for data sampled from a normal distribution, the occurrence of Type-I errors are generally close to α, i.e., ~50 false positives for every 1000 samples analyzed with 95% confidence limits (Nomikos and MacGregor, 1995; Martin and Morris, 1996). The number of Type-I errors for a non-Gaussian process, is much higher and included additionally the errors induced by the data modeling. As a result, the reliability of the supervision system is greatly reduced. This motivates alternate approaches for monitoring transient operations.

5.1.1 Need for Multiple Adjoined Models

For an intuitive appreciation of the multi-model approach, consider the previous example. The normal probability plot (Theil *et al.*, 1982) for each state is separately shown in Figure 5-2 and shows a much better fit, i.e., the underlying data from each state is locally linear, compared to the entire signal (Figure 5-1c). However, when multiple models are used in such cases, as illustrated in Sections 5.2, numerous false positives are encountered in the *border* between the models (see Figure 5-3). This high level of errors during model switching is due to discontinuity in modeling a continuous transient operation that follows a smoothly varying – but nonlinear – trajectory as versus one with abrupt changes from one state to another. This chapter addresses this critical problem in monitoring transients by using overlapping models as described in the following section.

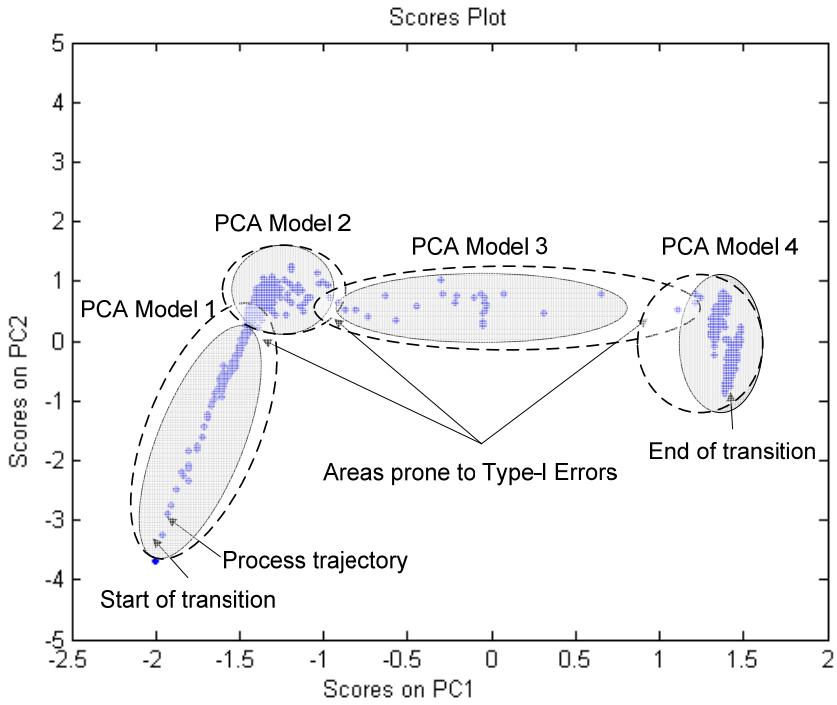


Figure 5-3: Illustration of areas within disjoint models (solid lines) that are prone to false positives and their incorporation into adjoined models (dotted lines)

5.2 Adjoined Multi Model-based Approach for Monitoring Transitions

In this section, an adjoined multi model-based methodology is proposed for online monitoring and supervision of transient operations. As noted earlier, single model based approaches are inadequate for transient operations; existing multi-model approaches rely on disjoint models, which makes them prone to Type-I errors due to discontinuity in modeling transient, multiphase operations. A multi-model approach that allows the different models to overlap is therefore proposed.

Adjoined models allow the representation of the smooth evolution of the transient operation in addition to any abrupt changes that occur. As a result, the monitoring statistics constructed for each constituent model comprehensively covers the relevant portion of the normal operating region (NOR), so false positives do not

occur during model-switching. Also, compared to monolithic models, the control limits are tighter and hence more sensitive to process faults. Such properties are ideally suited for monitoring transient operations which often characterized with non-linearity, non-stationary, and different batch length.

Though the proposed approach can be applied to most model-based approaches, it is illustrated here using multivariate statistical models, specifically PCA. Similar to other multi-model approaches, different PCA models are developed for different operating regimes. Traditionally, the state space is segmented into regimes by partitioning the training data into different non-overlapping clusters, i.e., each training sample contributes to exactly one model. Here, such concept is generalized and each training sample is allowed to contribute to possibly more than one model. The underlying precept is that for transient operations, crisp clustering draws artificial boundaries in a smooth distribution. Therefore adjoining points at the boundary would be assigned different clusters, which would lead to abrupt transitions between models. Such abrupt transitions lead to false positives. The proposed approach overcomes this by allowing a training sample to be used for multiple (one or more) models. While this has no affect on points that are at the interior of the cluster, points in the periphery would be assigned to multiple (typically 2) clusters and therefore contribute to the corresponding models. The resulting areas of influence of the constituent models would therefore intersect. During online monitoring, this intersection of the models can be exploited to ensure a smooth transition between the models.

The proposed methodology is based on fuzzy clustering as a preprocessing step to PCA model development. Fuzzy clustering of historical data is used to differentiate multiple modes of operations in temporal signals. The membership information obtained through fuzzy clustering provides a means to construct overlapping PCA

models since the suitability of assigning a sample to multiple clusters can be assessed; such membership information is not available in crisp clustering algorithm such as *k*-means. The training data is then split into multiple overlapping groups and a PCA model is constructed for each data group. During online monitoring, at every instant the PCA model that best describes the current state is selected. Monitoring statistics from this best-fit PCA model is then used for monitoring. Each of the above steps is described in detail next.

5.3 Sample Assignment for Training Multiple Models using Fuzzy c-means

Adjoined models are created based on clustering the normal operating data. Let a normalized 2-dimensional training data be $X = \{x_1,...,x_i,...,x_I\}^T$, where x_i is the i^{th} sample of *X*. *X* can be partitioned into different clusters through *k*-means clustering which assigns each sample x_i to a cluster $k \in [1, K]$. The membership function u_{ik} of sample x_i is given as:

$$u_{ik} = \begin{cases} 1 & \text{if sample } x_i \text{ is assigned to cluster } k \\ 0 & \text{otherwise} \end{cases} \qquad \text{(Eq 5-1)}$$

The cluster assignment seeks to minimize the total squared distance, ε_p (Seber, 2004):

$$\varepsilon_p = \sum_{k=1}^{K} \sum_{i=1}^{I} \| x_i \cdot u_{ik} - c_k \|, \qquad \text{(Eq 5-2)}$$

where, c_k is the centroid of the k^{th} cluster and $\|...\|$ notates the Euclidean distance.

$$c_k = \frac{\sum_{i=1}^{I} x_i \cdot u_{ik}}{\sum_{i=1}^{I} u_{ik}} \qquad \text{(Eq 5-3)}$$

The *k*-means classifies each sample so that it is assigned to exactly one cluster. The fuzzy *c*-means clustering (Bezdek, 1974) is a generalization of the *k*-means, where each sample can be assigned to one or more clusters as per a membership grade. Unlike in *k*-means, the membership function in fuzzy *c*-means u_{ik} is continuous with $u_{ik} \geq 0$ and indicates the extent to which a sample matches a cluster. The membership function is related to the distance from the centroid (Hathaway and Bezdeck, 1988):

$$u_{ik} \propto \frac{1}{\|x_i - c_k\|} \qquad \text{(Eq 5-4)}$$

When u_{ik} are normalized across all clusters and fuzzified with parameter ν:

$$u_{ik} = \frac{1}{\sum_{\substack{r \in [1,K] \\ r \neq k}} \left(\frac{\|x_i - c_k\|}{\|x_i - c_r\|} \right)^{2/(\nu-1)}}, \ \nu \geq 1 \qquad \text{(Eq 5-5)}$$

A ν value of 2 is normally used which is equivalent to normalizing the u_{ik} linearly to make their sum 1.

$$\sum_{k=1}^{K} u_{ik} = 1 \qquad \text{(Eq 5-6)}$$

When $\nu \to 1$, the cluster center closest to the point is given much more weight than the others, and the algorithm reduces to the *k*-means.

The centroid of the cluster is calculated as the mean of all samples weighted by their membership to the cluster.

$$c_k = \frac{\sum_{i=1}^{I} x_i \cdot {u_{ik}}^{\nu}}{\sum_{i=1}^{I} {u_{ik}}^{\nu}} \qquad \text{(Eq 5-7)}$$

The fuzzy *c*-means algorithm computes the memberships by iteratively minimizing (Hathaway and Bezdeck, 1988):

$$\varepsilon_p = \sum_{k=1}^{K}\sum_{i=1}^{I} ||x_i \cdot u_{ik}^v - c_k|| \qquad \text{(Eq 5-8)}$$

The procedure is terminated when the change in u_{ik} between two iterations is small. Since this could terminate at a local minimum, the fuzzy *c*-means algorithm is usally repeated multiple (P) times, starting from different initial assignments. The subscript p in ε_p signifies the p^{th} such replicate. Of the P replicates, the one that yields the minimum ε_p is selected and the samples assignment from it used for model construction.

5.4 Constructing Adjoined Models

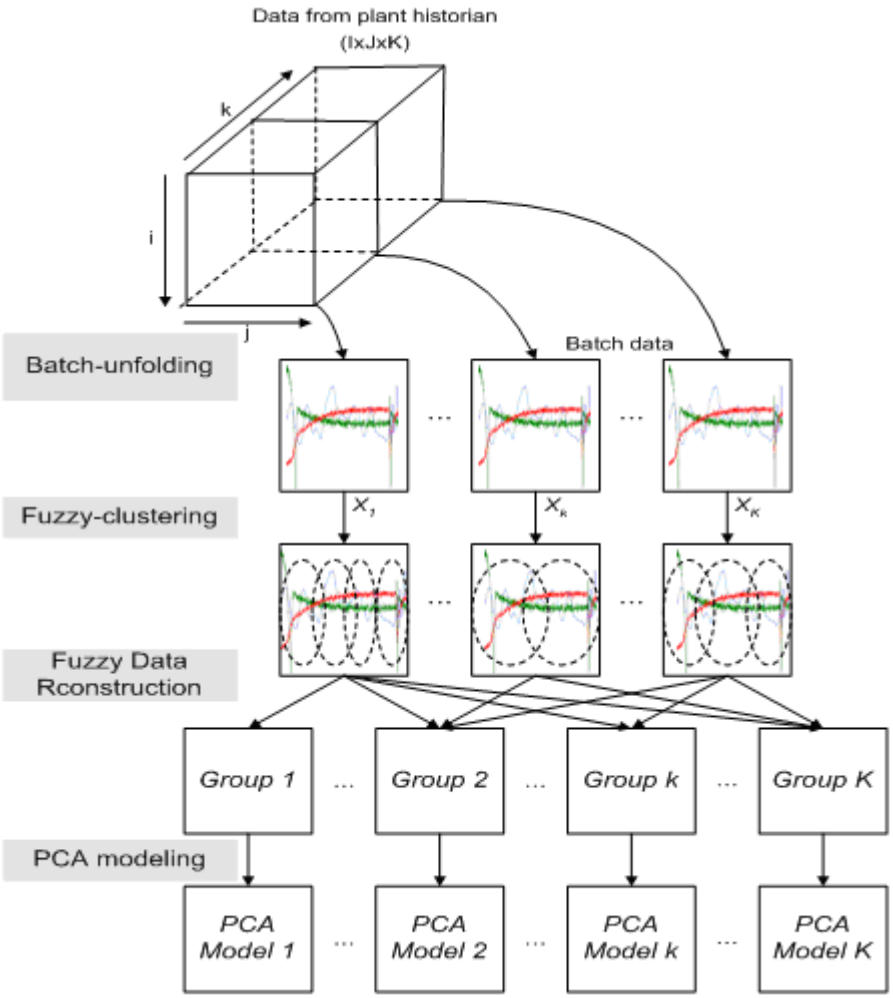


Figure 5-4: Offline training methodology for the proposed adjoined-PCA method

Based on the clustering results, K different training data groups are prepared from X (see Figure 5-4) which are then used to train the K PCA models. A sample x_i can be concurrently present in one or more data groups G_k, $k \in [1, K]$. This assignment of a x_i to data groups is based on the fuzzy cluster membership u_{ik}. A simple method is to assign the sample to a data group based on a threshold participation:

$$x_i \in G_k \quad \text{if } u_{ik} > \theta$$

However, the selection of a robust θ is difficult. An alternate criterion based on cluster separability is therefore proposed.

Each x_i of X is analyzed based on its membership in G_k. Consider the training data X along with the fuzzy cluster membership, u_{ik}. The highest value of u_{ik} thus gives the cluster which is closest to x_i, designated here as the best membership of x_i, B_i^1:

$$B_i^1 = \arg\max_k (u_{ik}), \; k \in [1, K],$$

The k^{th} best cluster can be identified through:

$$B_i^k = \arg\max_k (u_{ik}), \; k \in [1, K], \qquad \text{(Eq 5-9)}$$

subject to constraints:

$$B_i^k \notin \{B_i^{(k-1)}, B_i^{(k-2)}, ..., B_i^1\} \;\; \&$$

$$B_i^k - B_i^{(k-1)} < \delta. \qquad \text{(Eq 5-10)}$$

The *adjoining threshold*, δ, defines the allowable degree of overlap between the data groups G_k, where $0 \le \delta \le 1$. A large value of δ allows complete overlap between the regions identified while $\delta = 0$ prevents any overlap and reduces to the traditional non-adjoint multiple models.

Each G_k is used to train a PCA model M_k, $k \in [1, K]$. Model M_k would share some similarity with its immediate neighbors since the two have common training data especially in the boundaries. The similarity between the models can be measured based on the angles between the spaces of their PCs (Krzanowski, 1979). Let G_k and $G_{k'}$ be two data groups. The similarity between the two groups can be quantified by comparing their principal component subspaces L and M, which are the eigenvector matrices corresponding to the first C PCs (Krzanowski, 1979):

$$S^{PCA}(A,B) = \frac{1}{C}\sum_{c=1}^{C}\sum_{c'=1}^{C}\cos^2\theta_{cc'} = \frac{trace(L'MM'L)}{C}, \qquad \text{(Eq 5-11)}$$

where $\theta_{cc'}$ is the angle between the c^{th} PC of G_k and the c' PC of $G_{k'}$. A small values of S^{PCA} indicates low similarity between the models while a large value signifies high similarity. A shortcoming of S^{PCA} is that it is not normalized. A modified form was therefore proposed by Singhal and Seborg, (2002) by normalizing the similarity factor with eigenvalues:

$$S_{\lambda}^{PCA}(A,B) = \frac{\sum_{c=1}^{C}\sum_{c'=1}^{C}\lambda_c^A\lambda_{c'}^B\cos^2\theta_{cc'}}{\sum_{c}^{C}\lambda_c^A\lambda_c^B}. \qquad \text{(Eq 5-12)}$$

S_{λ}^{PCA} is in the range of 0 to 1.

The S_{λ}^{PCA} is used here to evaluate the number of models that are needed to represent the transient operation. The optimal K is selected by starting from a large value of K and measuring the similarity of all pairs of the resulting models. If there exist any model pair whose S_{λ}^{PCA} is larger than a selected threshold, γ, that K is considered high. The procedure is therefore repeated with a smaller K. Once the

optimal number of models have been identified and trained, they can be used for monitoring.

5.5 Choosing Current Active Model for Online Monitoring

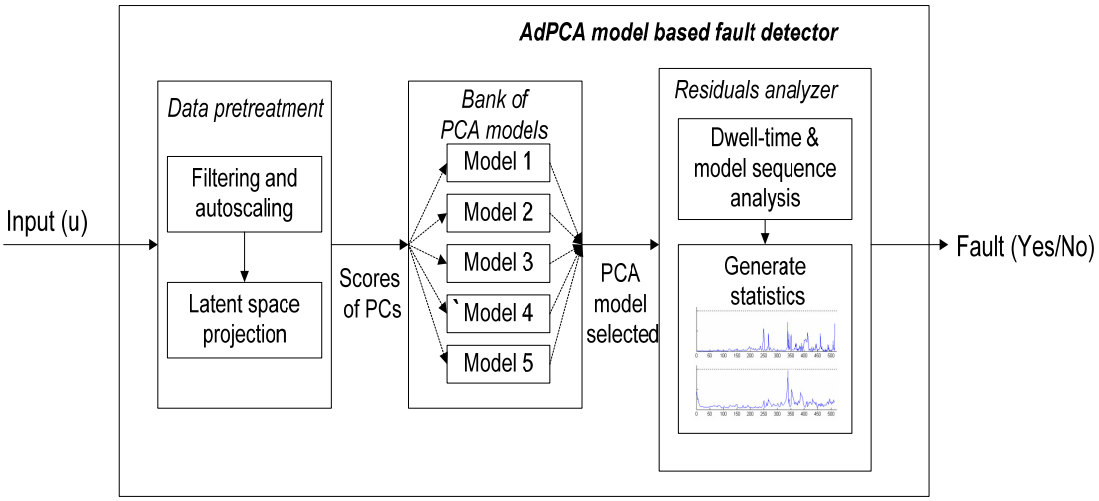


Figure 5-5 : Architecture of adjoined-PCA

In the proposed approach, as shown in Figure 5-5, only one model is used for online monitoring. This requires that the model that best describes the operation, M^{opt}, at time *t* has to be selected. The SPC statistics used for online monitoring is based on the identified M^{opt}. Let the samples collected online be designated as x_i. The distance between each $M_k, \forall k \in [1, K]$ and x_i first needs to be evaluated. For evaluating the distance between x_i and M_k, the combined discriminant similarity factor, Φ is used. At every instant, the distance of x_i the k^{th} PCA model , $M_k, k \in [1, K]$, is evaluated as:

$$\Phi_{ik} = \beta \bullet \frac{SPE_{ik}}{Q_k(\alpha)} + (1-\beta)\frac{T_{ik}^2}{T_k^2(\alpha)} \qquad \text{(Eq 5-13)}$$

Here, SPE_{ik} and T_{ik}^2 are the *SPE* and T^2 for x_i generated from the k^{th} model, and $Q_k(\alpha)$, $T_k^2(\alpha)$ the control limits from the k^{th} model. The nearest PCA model to x_i is selected as:

$$M_i^{opt} = \arg\min_{k\in[1\ ,\ K]} \Phi_{ik} \qquad \text{(Eq 5-14)}$$

The monitoring statistics, SPE and T^2 from the M^{opt} model are then compared with their corresponding limits $Q_{opt}(\alpha)$ and $T^2_{opt}(\alpha)$ for fault detection. The proposed model development and monitoring approach is described next using two case studies.

5.6 AdPCA Method for Fault Detection

The proposed AdPCA method can be summarized as follows:

Offline Model Development

Step 1: *Data unfolding & normalization*: Let $\tilde{X}^{3d}$ be a 3-dimentional raw dataset collected from a plant historian. Time-wise unfolding is first carried out on the 3-dimensional dataset to reduce $\tilde{X}^{3d}$ to a 2-dimensional dataset $\tilde{X}$ for analysis. Each variable of the training data, $\tilde{X}_n$, is then range-normalized to eliminate the varying scales of the variables:

$$X_n = \frac{\tilde{X}_n - \tilde{X}_n^{min}}{\tilde{X}_n^{max} - \tilde{X}_n^{min}},\ n\in[1\ ,\ N]. \qquad \text{(Eq 5-15)}$$

Step 2: *Clustering*: A large initial K is selected. Fuzzy *c*-means clustering is then applied on X to identify membership function of each sample x_i compared to all classes, u_{ik}, $i\in[1,I]$ and $k\in[1,K]$.

Step 3: *Data reconstruction*: Overlapping data groups G_k, $k\in[1\ ,\ K]$ are created based on the data reconstruction method as described in Section 5.4 over all samples of X.

Step 4: *AdPCA model generation*: A PCA/DPCA model is trained for each G_k, $k\in[1\ ,\ K]$ identified. The samples within each group are autoscaled separately and projected to their principal components subspace as $AdPCA\in\{M_1,...,M_K\}$.

Step 5: *Model evaluation*: All PCA models are evaluated by comparing the $S_{k,k'}^{PCA}$ values across all M_k, $M_{k'}$ pair, where $k,k' \in [1, K]$ and $k \neq k'$. If there exist a model pair which gives $S_{k,k'}^{PCA} \geq \gamma$, K is reduced by 1 and the training algorithm repeated from Step 2.

Step 6: The trained AdPCA models $AdPCA \in \{M_1,...,M_K\}$ can be used for monitoring transient operations online.

Online Monitoring

As shown in Figure 5-5, the algorithm for online monitoring involves:

Step 1: *Data preprocessing*: Online process measurements, designated here as $\tilde{x}_{i'}$, are first scaled to $x_{i'}$ based on the mean and variance obtained previously for each model (Step 4 of the model development phase).

Step 2: *Projection to PC subspace*: $x_{i'}$ is projected to the principal components subspace of all available models $M_k, \forall k = [1,K]$, as scores and loadings.

Step 3: *Model selection*: The best PCA model, M^{opt}, is selected from the existing models based on Eq (5-13) and Eq (5-14).

Step 4: *Online monitoring*: Monitoring statistics, e.g.: SPE_i and T_i^2 are computed for x_i based on M^{opt} identified, and their corresponding limits $Q^{opt}(\alpha)$ and $T^{2(opt)}(\alpha)$ of M^{opt} used for monitoring.

5.7 Case Study 1: Monitoring Startup of a Distillation Unit

In this section, the proposed method is tested on a lab-scale distillation unit as described in Section 3.4. The startup normally takes two hours. From an operation point of view, four operating states can be identified during the startup, namely,

reboiler heating phase, boiling phase, reflux heating phase, and final steady-state phase. Different faults, both equipment failure and operator errors are introduced at different states of the operation and monitored using the proposed approach.

To develop the various models, the operating data from five normal transitions are used as reference data. A low-pass FIR filter is used to remove high frequency process noise in the reference data before they are auto-scaled.

In order to construct adjoined models, K training data groups are first constructed. An adjoining threshold, δ, of 0.2 is chosen for constructing the adjoined-PCA models. Process samples whose cluster separability ($B_i^k - B_i^{(k-1)}$) is lesser than δ are placed in multiple data groups (see Figure 5-6). When the PCA models are formed using these data groups, the neighboring models become adjoined. The criterion described in Section 5.5 is used here for selecting the number of data groups, K. The training algorithm is started with an initial K of 15, and γ of 0.85. If any pair of the constituent models, M_k and $M_{k'}$ ($k \neq k'$) shows high similarity ($S_{k,k'}^{PCA} > \gamma$), K is reduced by 1 and the training repeated.

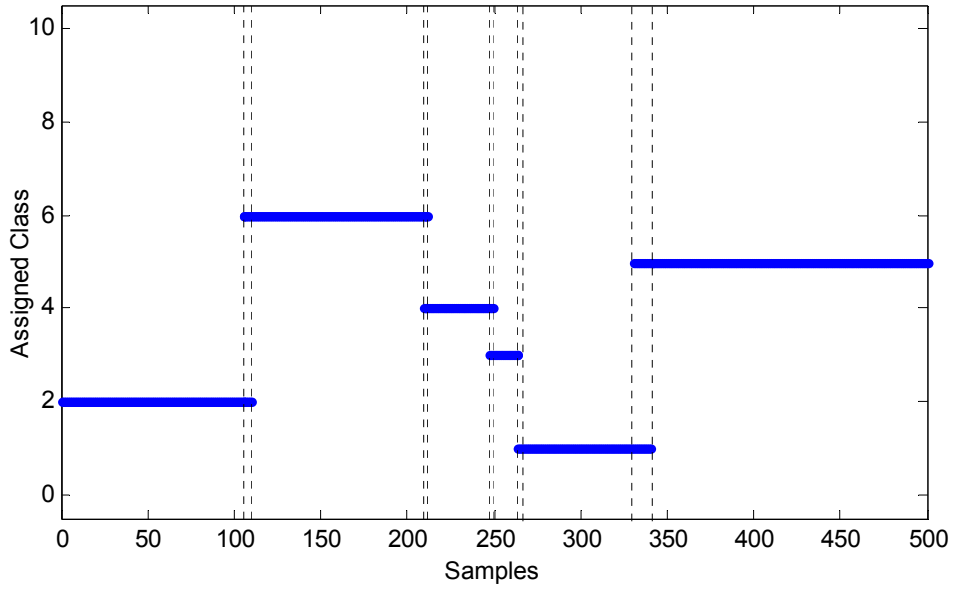


Figure 5-6: Class assignment of samples during one normal run of the distillation unit

The AdPCA model constructed by following this training procedure contains six constituent models. The number of PCs for each model M_k is selected such that cumulative percentage variance captured is $\geq 95\%$. Models M_2, M_4, and M_6 correspond to reboiler heating phase; M_3 and M_1 to boiling and reflux heating phases, and M_5 to the final steady-state phase. Their S^{PCA} model comparison values are shown in Table 5-1. All models show low similarity with one another with the highest inter-model similarity value of 0.7568.

Table 5-1: $S^{PCA}(M_k, M_{k'})$ between PCA models identified for distillation unit startup case study

PCA Model #	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**
M1	1.0000	0.4351	0.6877	0.4940	0.6366	0.5216
M2		1.0000	0.5182	0.5347	0.5837	0.6288
M3			1.0000	0.7240	0.5032	0.7568
M4				1.0000	0.4721	0.7366
M5					1.0000	0.5196
M6						1.0000

When tested with a different control dataset (normal operation), the AdPCA model shows high reliability, as shown in Figure 5-7. During normal operation, all six models are used for process monitoring (see Figure 5-7a which shows the model used at each instant). Initially at *t*=10s (sample 1), the combined statistics (with $\beta = 0.5$) for model M_2, $\Phi_{1,2}$ is the smallest among all models and is chosen as the current active model, M^{opt}. At *t*=1080s, $\Phi_{108,6}$ reduces to 0.064 from 0.113 while $\Phi_{108,2}$ increases to 0.070. Since $\Phi_{108,6}$ is the smallest, the method switches to M_6 as the current active model for monitoring. Similarly, the current active model is switched as the process evolves. The remaining model switches occur at *t*=2100s ($M_6 \rightarrow M_4$), *t*=2450s ($M_4 \rightarrow M_3$), *t*=2640s ($M_3 \rightarrow M_1$), and *t*=3400s ($M_1 \rightarrow M_5$). The reduced *SPE* and T^2

statistics from the current active model are used for monitoring the process. As seen from Figure 5-7b, there are no Type-I errors throughout the run.

For comparison, MPCA and DPCA models are also constructed based on the reference data. Six PCs are retained for the MPCA model while the DPCA model is constructed with l=2 and retaining 13 PCs. When tested with the same control dataset, both MPCA and DPCA are prone to false positives during the reboiler heating ($t \sim [0\ ,\ 30]s$) and column stabilization phases ($t \sim 340s$). DPCA shows 11 false positives (Figure 5-8) while MPCA shows 6 false positives. A classical multi PCA model with *no overlap* (termed disjoint PCA, or DisPCA) is also created to compare the performance improvement arising from adjoining the models. DisPCA reduces the number of Type-I errors during $t \sim [0\ ,\ 30]s$ compared to DPCA and MPCA, since nonlinearity during the initial phase of the startup is better modeled by the multiple PCA models. However, DisPCA performs poorly during model switching, i.e., 5 false positives are observed during model switches around t~2640s and t~3400s. This illustrates the advantage of adjoined models for monitoring transient operations. The overlapping models ensure continuity in the PCA subspace and reduce the occurrence of Type-I errors during switching of models. Next, the efficiency of the proposed AdPCA method during various faults was evaluated. In this study, ten process disturbances as listed in Table 3-2 have been tested. Three disturbances are described in detail next.

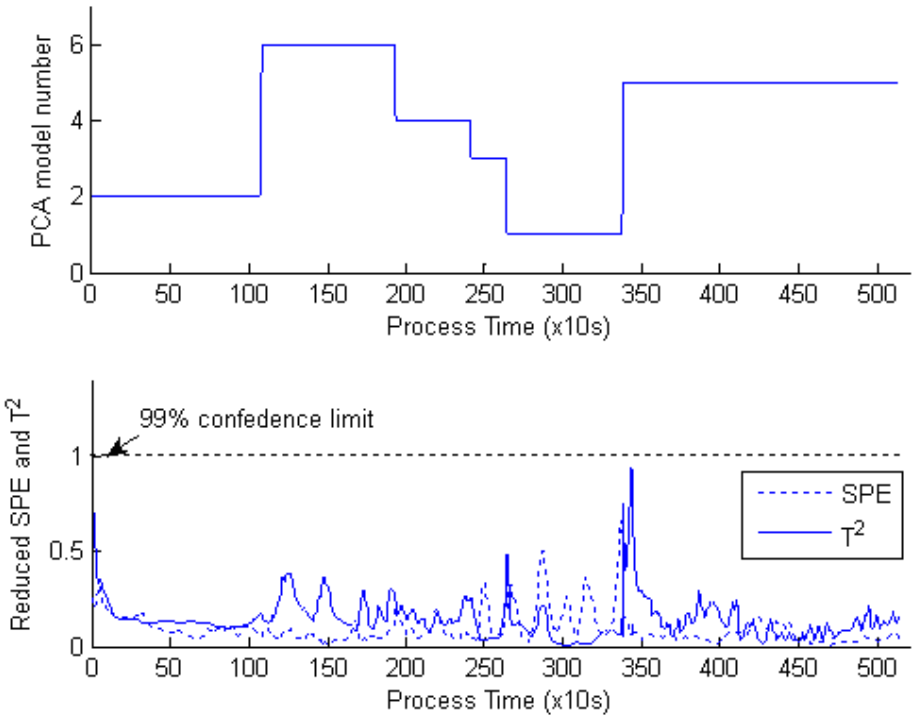


Figure 5-7: Monitoring of a normal startup of the distillation unit using AdPCA

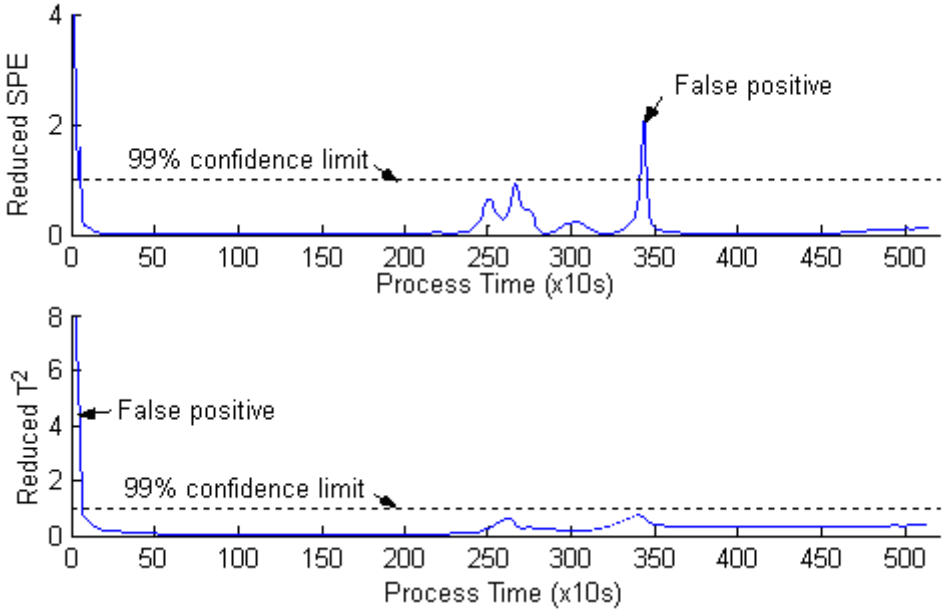


Figure 5-8: Monitoring of a normal startup of the distillation unit using DPCA

Scenario 1: Feed pump fault

DST04 corresponds to a feed pump malfunction with its process signals shown in Figure 5-9. The fault was introduced at *t*=3550s during step 7 of the startup SOP when the operator erroneously sets the pump power at 55% of the nominal value. Such an action would adversely affect the throughput of the ethanol production and lead to continuous reduction in the reboiler level. There are two stages of fault propagation. Stage-1 spans from *t*=3560s to *t*=4600s. Recovery is possible only when the fault is still in an early stage. If the fault remains undetected or unidentified, the reboiler level will drop below the minimum operating limit (Stage-2 starting around *t*=4600s) and shutdown procedures will be automatically triggered by the safety interlock.

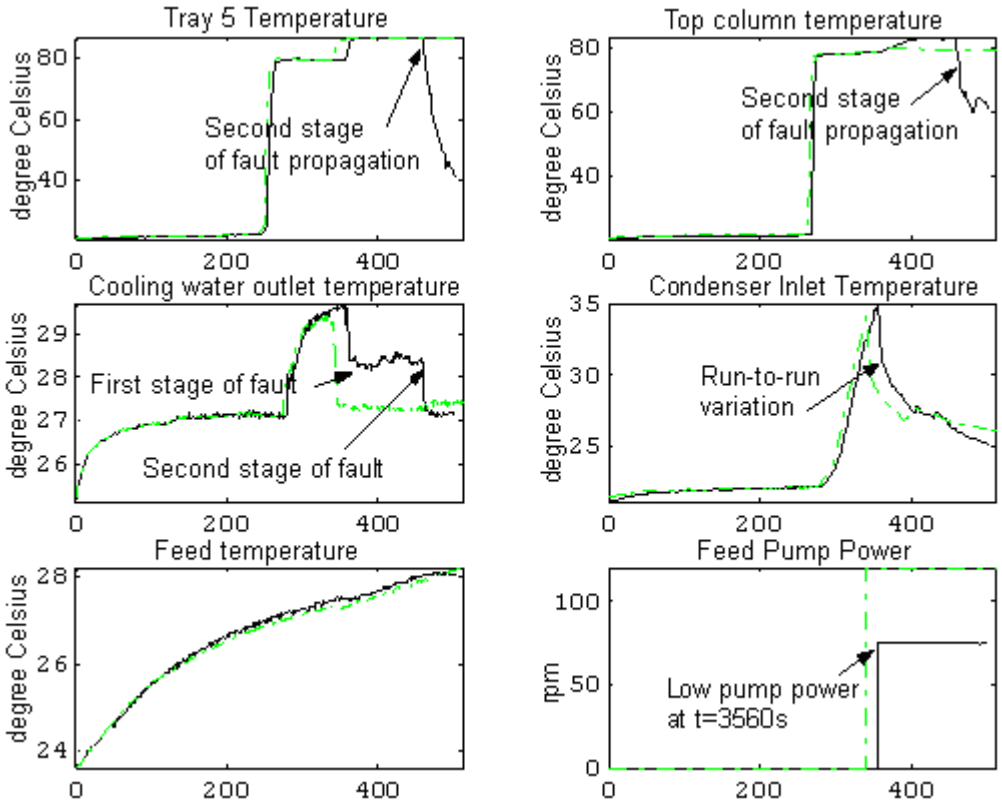


Figure 5-9: Process signals for DST04 (x10s) – the dotted lines indicate the process signals of a normal startup while the dark lines represent the signals of the faulty run

During $t \in [1,\ 2300]s$, the process is in reboiler heating phase and models M_2, M_6, and M_4 are used for monitoring (Figure 5-10). During the boiling and column-stabilization phases, the method switches to models M_3 ($t \in [2300,\ 2670]$s) and M_1 ($t \in [2680,\ 3540]$s) as during the normal startup operation. When the fault occurs at *t*=3550s, the steady-state model M_5 which gives the smallest combined statistics (Eq 5-14) is selected for monitoring since the operation is progressing towards steady-state (SOP step 7 & 8 have been executed). As can be seen from Figure 5-10, at *t*=3560s both *SPE* and T^2 statistics violate the upper control limit of model M_5 and a fault is flagged. Although corrective measures could be performed at this juncture based on this information, for the sake of illustration, in this run, no recovery action is taken; hence the safety interlock is triggered and the distillation unit shut down. The temperatures in the column then begin to drop and model M_1 (*t*~4600s) is chosen as the current active model, indicating the process is progressing towards the cold state.

MPCA, DPCA, and DisPCA were also tested on this scenario. The MPCA method detects the fault only in Stage-2 (*t*=4600s with 6 false positives around *t*~[1, 60]s) and fault recovery is not possible (see Figure 5-11). DPCA method improves sensitivity by detecting the disturbance earlier (*t*=3570s) but with 10 false positives. The performance of DisPCA is comparable to AdPCA in terms of detection speed (detection at *t*=3560s). However, 3 false positives arise during model switching (from M_3 to M_1). The proposed AdPCA method hence gives the best performance and detects the fault promptly without any false positives.

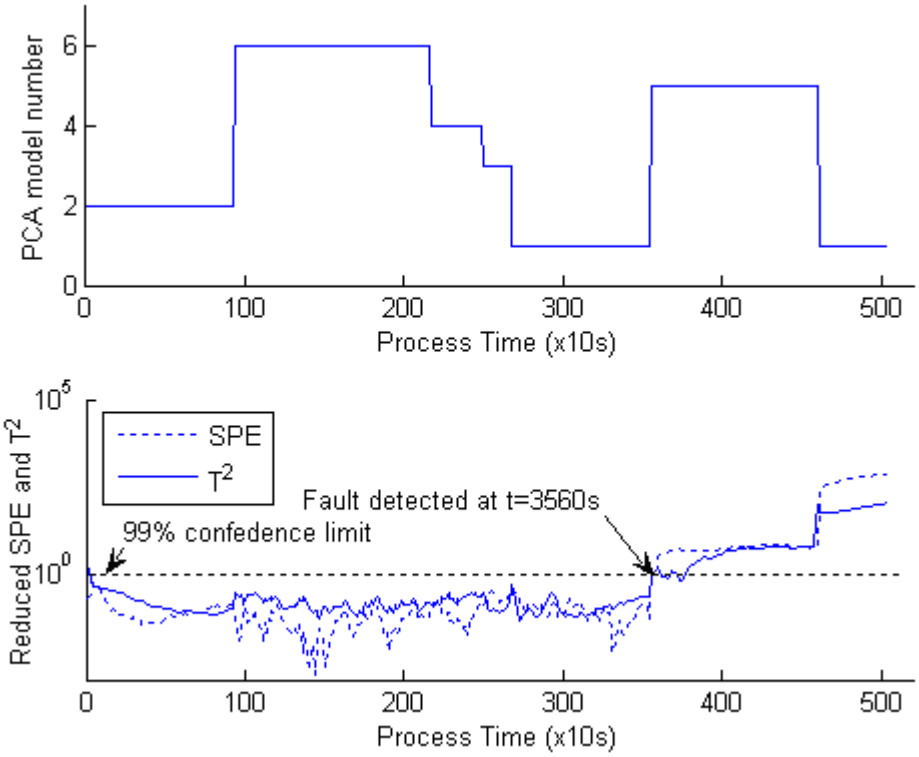


Figure 5-10: Monitoring of DST04 using AdPCA

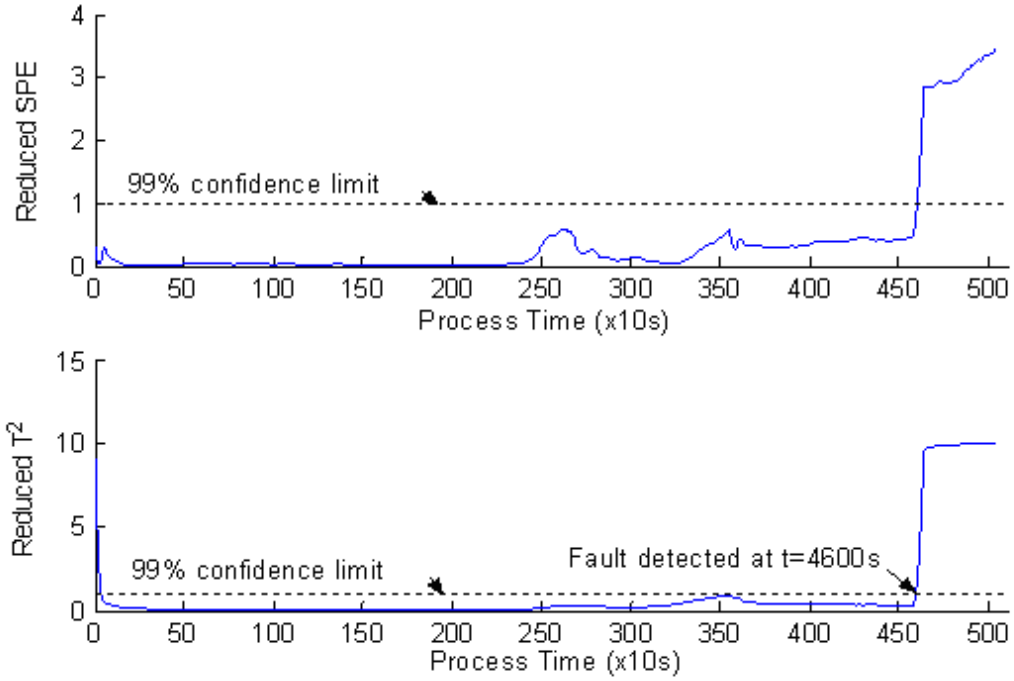


Figure 5-11: Monitoring of DST04 using MPCA

Scenario 2: Sensor fault

DST05 corresponds to a sensor fault. The fault was introduced at t=4250s when the process approaches steady-state. In this scenario, all variables remain normal except for the affected sensor (Tray 6 Temperature Sensor). Both MPCA and DPCA fail to detect this fault. Large number of Type-I errors (12 false positives for MPCA and 8 false positives for DPCA) is observed for both methods around t~[1, 200]s and t~3400s. When disjoined models are used (based on DisPCA), the method becomes more sensitive and the fault is promptly detected at t=4260s. However, five normal samples are rejected when the model switches from M_3 to M_1 and M_1 to M_6 (see Figure 5-12). In contrast, the proposed AdPCA method is able to detect the sensor fault promptly at t=4260s with only 1 Type-I error at t=3750s due to run-to-run deviations (see Figure 5-13).

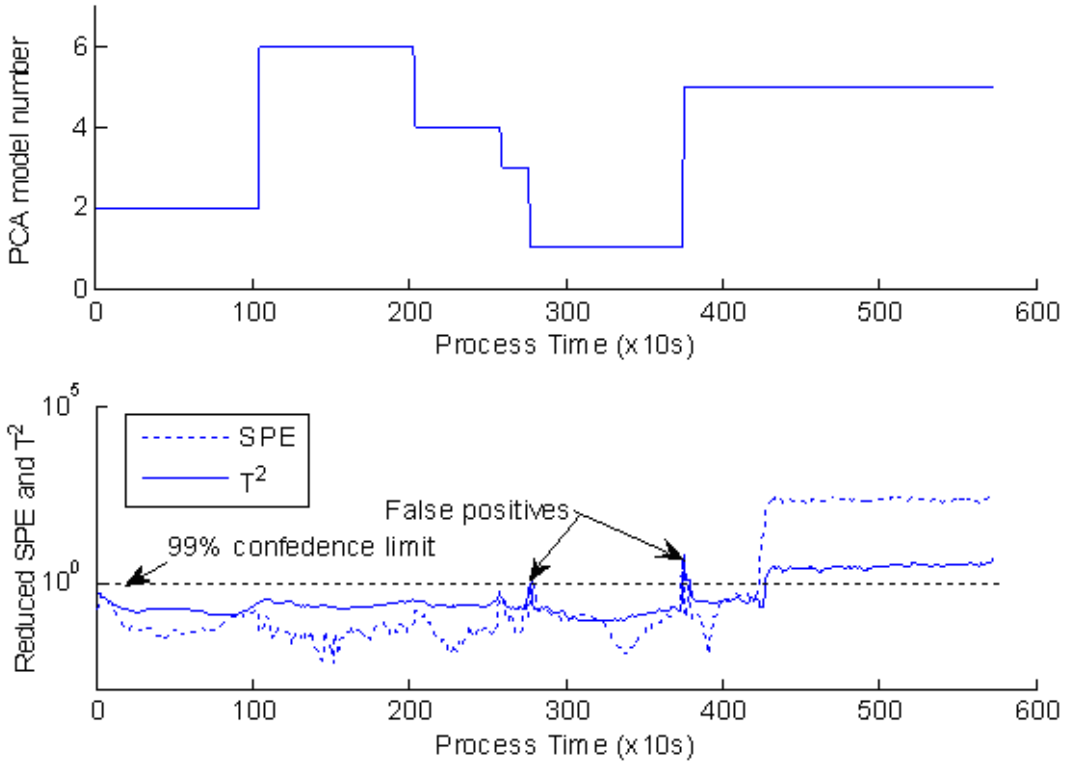


Figure 5-12: Monitoring of DST05 using DisPCA

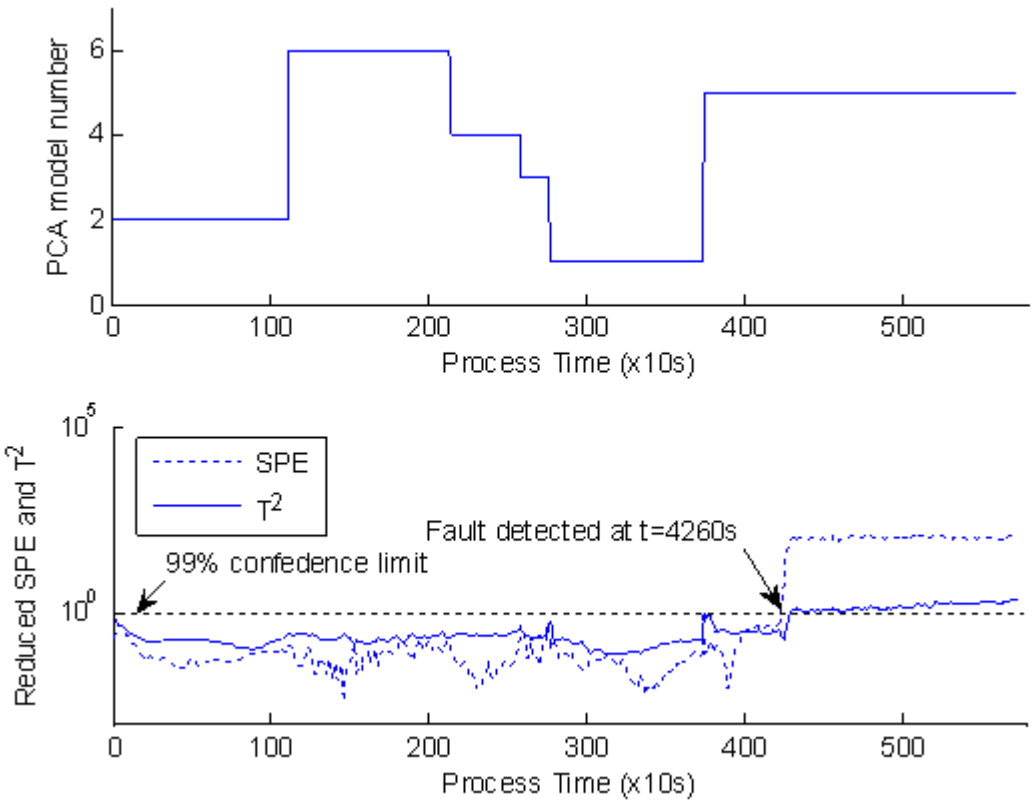


Figure 5-13: Monitoring of DST05 using AdPCA

Scenario 3: Low reflux rate

DST07 corresponds to a fault in reflux ratio. The fault was introduced at step 8 of the SOP (t=3440s) when the operator set the value of reflux ratio to 50% lower than that of the nominal value. MPCA fails to detect the error while DPCA detects the fault at t=3490s with 4 false positives (Table 5-2). DisPCA also suffers from 4 false positives during switching of models. The proposed AdPCA technique yields the best performance by detecting the fault at t=3450s with no false positives.

The summary of results for monitoring other faults is presented in Table 5-2. In general, MPCA and DPCA are prone to Type-I and Type-II errors. MPCA misclassified 37 samples in total while DPCA misclassified 48 samples. MPCA failed to detect two disturbances (DST05 and DST07) and DPCA failed to detect one – DST05. The average detection delay for MPCA and DPCA are 670.0s and 356.7s,

respectively. The proposed AdPCA method gives the best performance in all cases by (i) detecting all faults promptly with an average detection delay of 310.4s, and (ii) reducing Type-I Errors (7 total misclassified samples). The improvement in AdPCA is achieved because of (i) using multiple models to represent transitions more accurately, and (ii) making the models overlap to ensure continuity within the models. Switching of models during online monitoring can hence be performed smoothly without excessive Type-I errors.

Next, the robustness of the proposed AdPCA approach to various δ and K is evaluated, which govern the above two features of AdPCA. To study AdPCA's robustness to different values of K, a selectivity (detection accuracy) versus sensitivity (earliness of detection) analysis is conducted. For this study, instead of the optimal K, the number of models is prescribed without consideration of the inter-model similarity. The method reduces to MPCA when K=1 – the average detection delay is hence 670s and there are 37 total false positives. When a small K value is used, the constructed models fail to represent the nonlinearity and discontinuities in the training data. Increasing K leads to an improvement in the average detection delay. However, the total false positives observable is still large (>20) for $K \leq 3$. When $K \geq 4$, the total Type-I errors from AdPCA decreases to 10 and the detection speed also improves with average detection delay of about 390s (Table 5-3). For a wide range of K values the performance is robust; for instance, even K=20 results in the same detection delay (~390s) and a minor difference in the total Type-I errors (1 additional false positive) is observed. The effect of varying the level of overlap between the models is shown in Figure 5-14. As expected, when δ=0, all models become disjoint (DisPCA) and an increase in Type-I error rate is observed, especially during switching of models (total of 26 false positives for all scenarios analyzed). A sharp decrease occurs when

$\delta \approx 0.05$ which decreases to ~7 for $0.2 \le \delta \le 0.5$. These remaining seven Type-I errors are caused by high variability in the cooling water temperature in some scenarios. These studies therefore indicate that the method is robust over a wide range of K and δ. Next, the proposed AdPCA method is applied to a batch process.

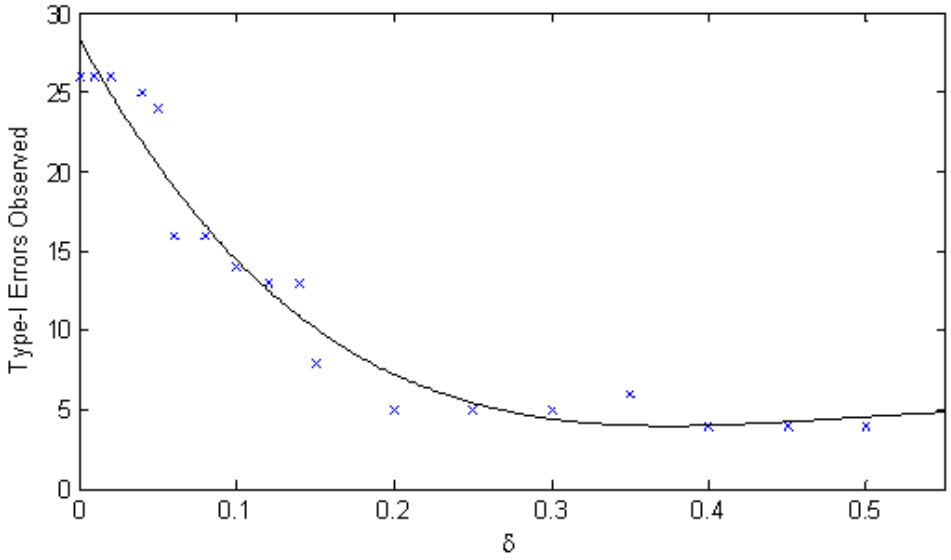


Figure 5-14: Type-I Error observed for different δ in distillation unit case study

Table 5-2: Summary of monitoring results for distillation unit startup case study

Fault ID		Multiway-PCA		Dynamic-PCA		Disjoined-PCA		Adjoined-PCA	
	Time fault introduced	Time Fault Detected	False Alarms	Time Fault Detected	False Alarms	Time Fault Detected	False Alarms	Time Fault Detected	False Alarms
DST01	1	1	0	1	0	1	0	1	0
DST02	1	1	0	1	0	1	0	1	0
DST03	359	365	2	372	5	368	4	368	2
DST04	355	460	6	357	10	356	3	356	0
DST05	425	-	12	-	8	426	5	426	1
DST06	352	353	4	353	7	353	3	353	2
DST07	344	-	2	349	4	345	4	345	0
DST08	346	471	5	347	4	347	4	347	2
DST09	1	1	0	1	5	1	0	1	0
DST10	1,400	300	6	300	5	301	3	301	0

Table 5-3: Selectivity vs. sensitivity analysis in distillation unit startup case study

	Detection delay in x10s (misclassified samples)								
# Model	***K*=1**	***K* =2**	***K*=3**	***K*=4**	***K* =5**	***K* =8**	***K* =10**	***K* =15**	***K* =20**
DST01	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
DST02	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
DST03	6 (2)	10 (0)	9 (2)	9 (1)	9 (0)	9 (0)	9 (0)	10 (0)	9 (0)
DST04	105 (6)	103 (6)	1 (8)	1 (1)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)
DST05	- (12)	147 (2)	1 (5)	1 (2)	1 (2)	1 (2)	1 (2)	1 (2)	1 (2)
DST06	1 (4)	1 (7)	1 (3)	1 (3)	1 (5)	1 (3)	1 (2)	1 (2)	1 (3)
DST07	- (2)	2 (2)	2 (1)	1 (2)	1 (2)	1 (0)	1 (2)	1 (2)	2 (2)
DST08	125 (5)	124 (2)	124 (3)	124 (1)	1 (0)	1 (2)	1 (1)	1 (0)	1 (0)
DST09	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)
DST10	299 (6)	299 (4)	300 (0)	300 (0)	300 (0)	300 (0)	300 (0)	299 (0)	299 (0)
Avg Delay	**67.0**	**67.1**	**54.4**	**54.4**	**39.0**	**39.0**	**39.0**	**39.0**	**38.9**

5.8 Case Study 2: Monitoring of Fed-batch Penicillin Cultivation Process

Case Study Description

The production of antibiotics from penicillin is a multiphase process with nonlinear process dynamics. The penicillin fed-batch simulator PenSim v2.0 (Birol *et al.*, 2002) is used to generate the training and testing data for analysis. The process flowsheet is shown in Figure 5-15 The simulator is developed based on the mathematical model of Bajpai and Reuss (1980), and is able to capture the dynamics of sixteen process variables, namely, flow rates of input streams, temperature, pH, heat generated, aeration rate of fermenter, and concentrations of penicillin, CO_2, and substrate utilization under a variety of operating conditions. Since the product is very much affected by pH and temperature, they are therefore controlled at specified setpoint (pH 5.0 and 25°C through PID controllers). In all runs, an initial batch culture is followed by a fed-batch operation based on the depletion of the carbon source (glucose). The process switches to the fed-batch mode of operation when the level of glucose concentration reaches 0.3g/l. Eight process faults have been generated with the details of each fault given in Table 5-4. The variables selected for monitoring of penicillin cultivation process is given in Table 5-5.

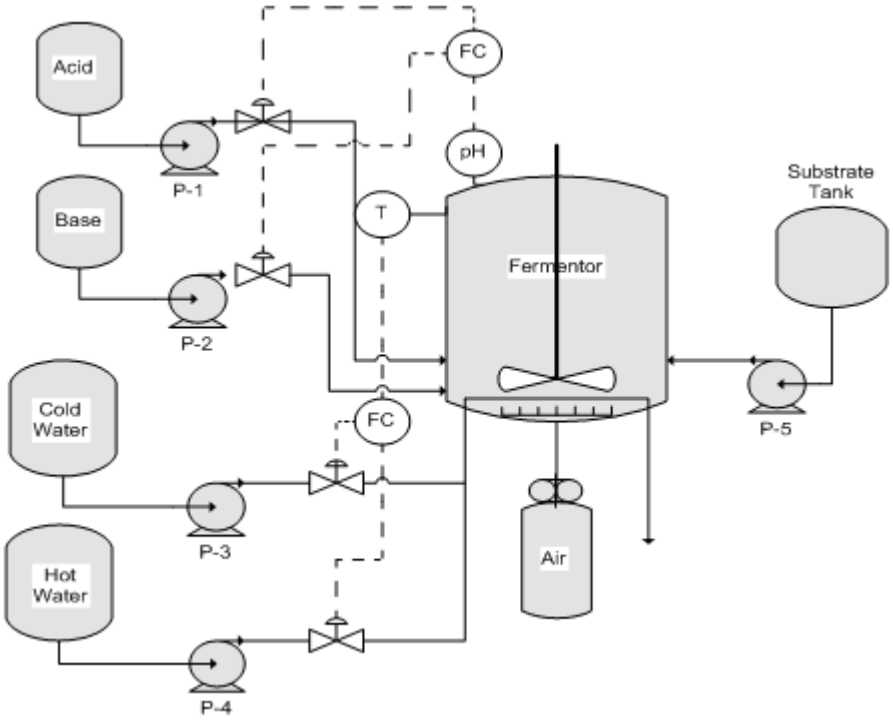


Figure 5-15: Process flowsheet of Penicillin cultivation process

Table 5-4: Summary of the eight fault scenarios considered in the penicillin cultivation case study

Case	Fault type	Occurrence time (h)
SIM01	pH controller failure	0.5
SIM02	Temperature controller failure	0.5
SIM03	15% step decrease in aeration rate	60
SIM04	15% step decrease in agitation power	30
SIM05	15% step decrease in substrate feed rate	50
SIM06	Ramp increase in aeration rate	70
SIM07	Ramp increase in agitation power	40
SIM08	Ramp increase in substrate feed	30

Table 5-5: Variables used monitoring of penicillin cultivation process

Number	Variables	Units
1.	Aeration rate	$1h^{-1}$
2.	Agitation power	W
3.	Substrate feed rate	$1h^{-1}$
4.	Substrate feed temperature	K
5.	Dissolved oxygen concentration	% saturation
6.	Culture volume	litre
7.	Carbon dioxide concentration	mmol l^{-1}
8.	pH	-
9.	Fermenter temperature	K
10.	Generated heat	kcal
11.	Cooling water flow rate	$1h^{-1}$

Process Monitoring based on AdPCA

In this section, the proposed AdPCA method is tested with the Penicillin cultivatioin case study. A total of ten normal batches are simulated to create the reference data based on an integration step size of 0.02h and a sampling interval of 0.5h. The state variables during each normal run are highly nonlinear and many of them show strong discontinuities during the period of cultivation (see Figure 5-16). In all runs, an initial batch culture is followed by a fed-batch operation based on the depletion of the carbon source (glucose). The process switches to the fed-batch mode of operation when the level of glucose concentration reaches 0.3g/l.

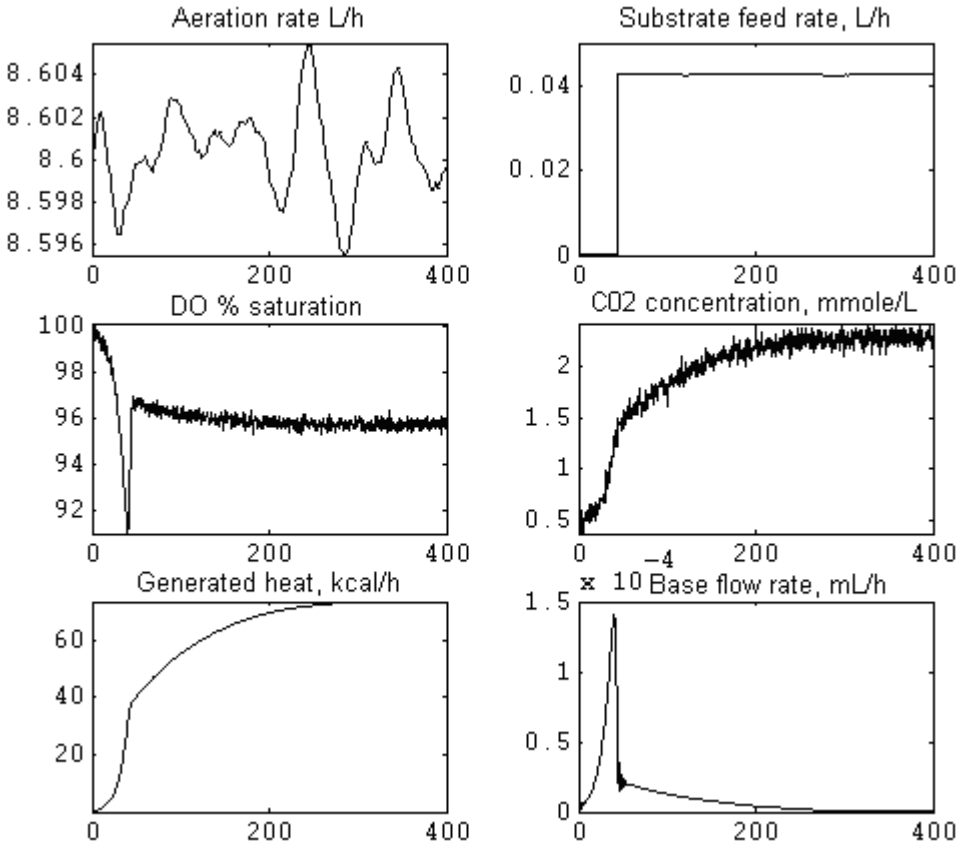


Figure 5-16: State variables of the penicillin cultivation process during a normal run

Next, the AdPCA model is constructed using the training data based on the state variables listed in Table 5-5. Similar to the previous case study, the AdPCA is initialized with K=15 groups, and K reduced when high similarity ($S_{\lambda}^{PCA} > 0.85$) is

found between any two models. Based on the proposed training criterion with δ=0.2, eight adjoined groups are created by making the samples whose $B_i^k - B_i^{k-1} < \delta$ exist in multiple groups. The eight models within the AdPCA overlap given the presence of some samples in multiple models. The assignment of training samples to different groups for one normal run is shown in Figure 5-17. Model M_3 corresponds to batch operation while the remaining correspond to fed-batch operation. All eight models show low similarity and the highest S_λ^{PCA} is 0.7694 (Table 5-6).

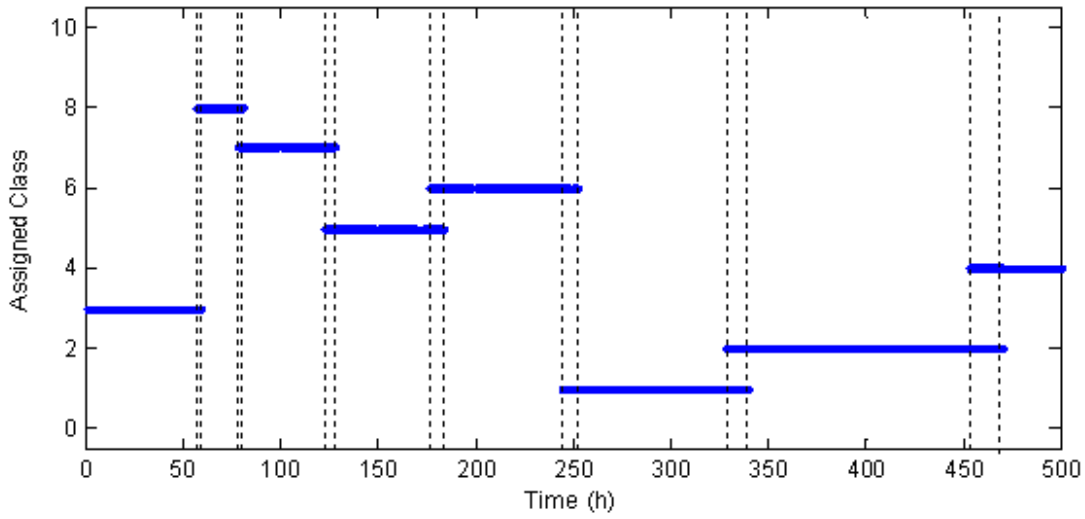


Figure 5-17 : Class assignment of data from one normal run of the Penicillin Cultivation Process

The constructed AdPCA is then tested with a control dataset (normal run) and the monitoring results shown in Figure 5-18. As can be seen, the proposed method shows high reliability with no occurrence of false positives. Different phases of the fermentation are also accurately detected by the proposed AdPCA model. The performance of classical PCA-based techniques is also studied for comparison. All MPCA, DPCA, and DisPCA are trained using the same training data. The MPCA requires 7 PCs and DPCA 13 (with *l*=2). Both MPCA and DPCA techniques are prone

to false positives at t~36.5h (Figure 5-19 for MPCA). Such observations are consistent with the analysis of Lee et al. (2004) and Doan and Srinivasan (2007). When tested with DisPCA with the same settings, 3 false positives are observed at t~45.0h and t~230.0h when the model switching occurred. The proposed method hence gives the best performance compared to other PCA-based techniques. Next, the efficiency of the proposed AdPCA method is evaluated using three types of disturbances, (i) controller failure, (ii) step changes, and (iii) ramp changes in process manipulated variables. In this study, the eight different disturbances listed in Table 5-4 have been tested. Scenarios SIM05 and SIM07 are described in detail next.

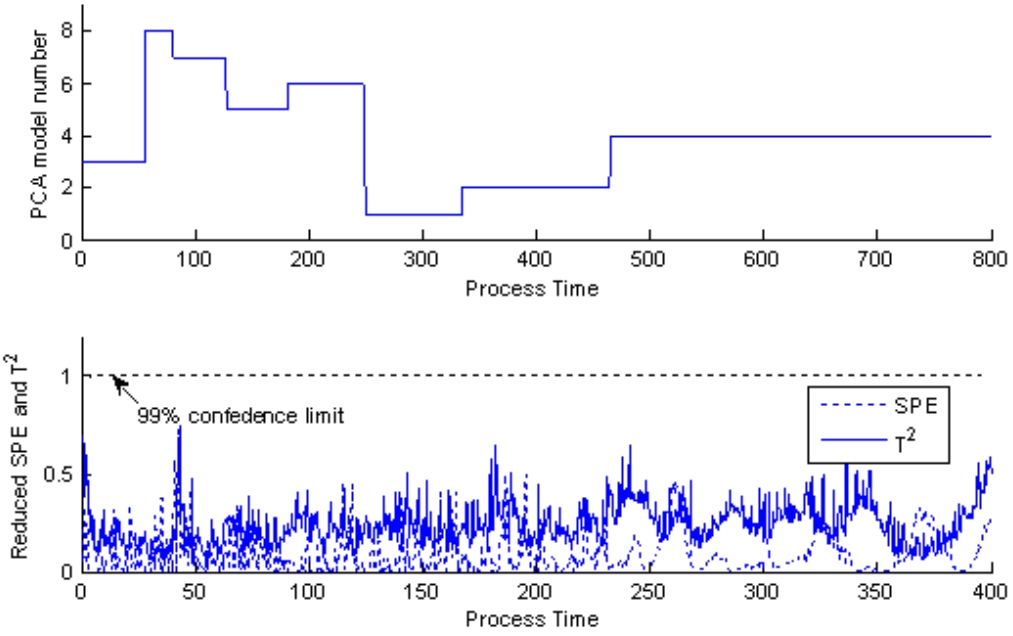


Figure 5-18 : Monitoring of a normal run of Penicillin Cultivation Process using AdPCA

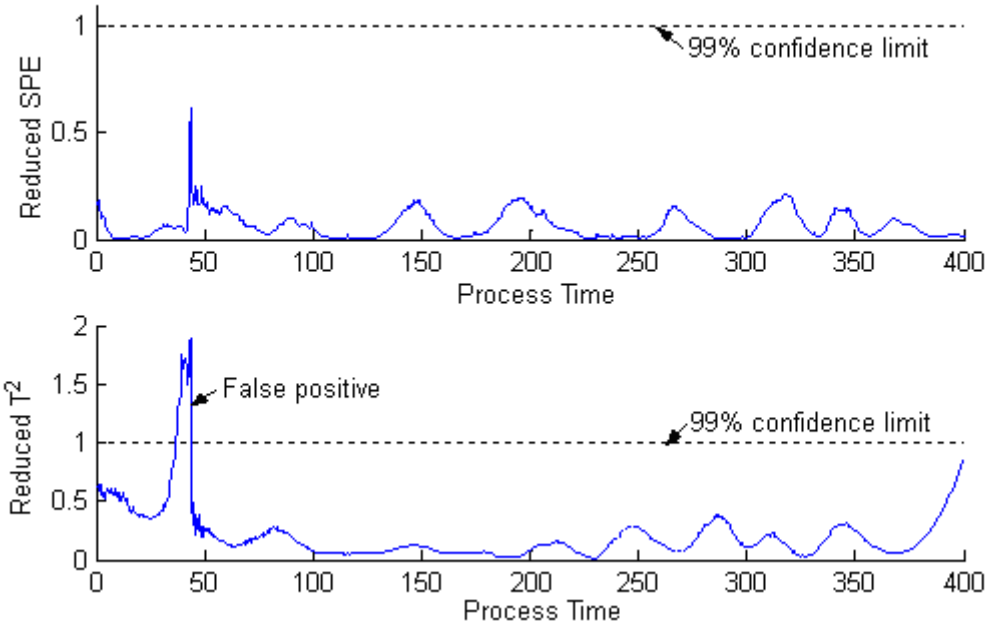


Figure 5-19 : Monitoring of a normal run of Penicillin Cultivation Process using MPCA

Scenario 1: Low aeration rate

SIM05 corresponds to a step decrease in substrate feed rate at t=50.0h. This reduction decreases the final product yield since the growth of the biomass is affected. MPCA and DPCA fail to detect this disturbance while the multi-model based approaches DisPCA and AdPCA detect the fault promptly at t=63.0h. During the batch, MPCA flags 13 false positives, DPCA 22 (see Figure 5-20), and DisPCA 2 – both due to discontinuity in modeling the transient operation. In contrast, AdPCA has no Type-I error during the run (see Figure 5-21).

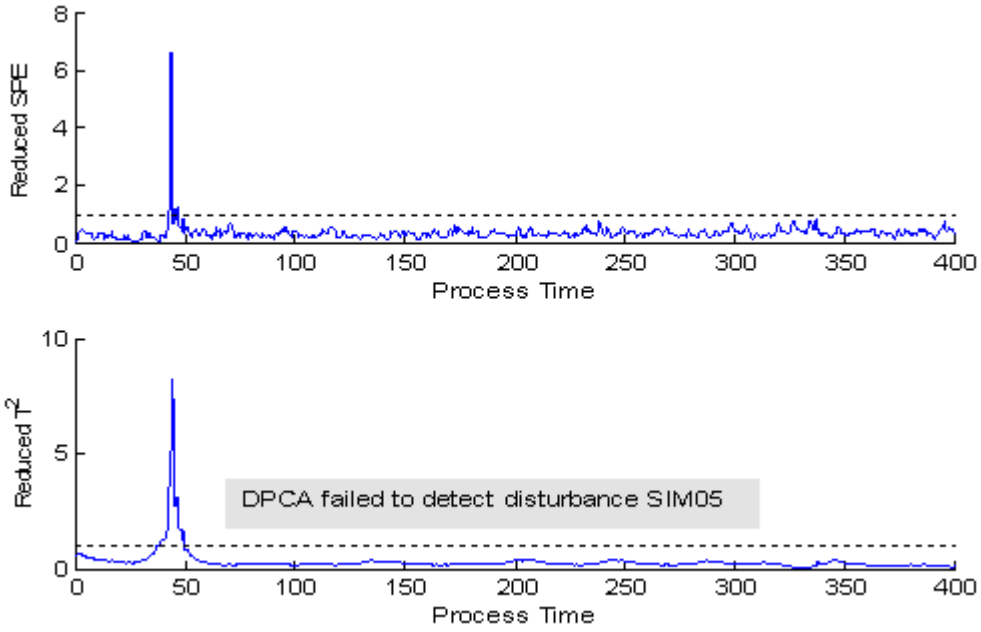


Figure 5-20: Monitoring of SIM05 using DPCA

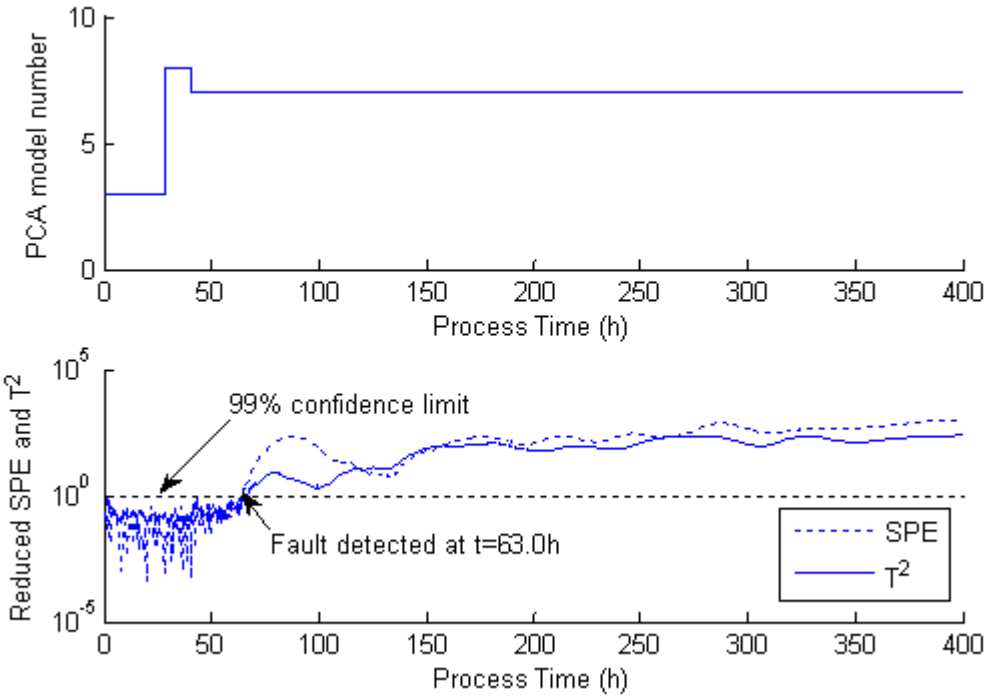


Figure 5-21: Monitoring of SIM05 using AdPCA

Scenario 2: High agitation power

SIM07 corresponds to a ramp increase in agitation input power which increases the shear on the biomass. MPCA detects the fault at t=248.0h when the SPE exceeds the 99% confidence limit. DPCA detects the fault earlier at t=241.0h based on the T^2 statistic. Both DisPCA and AdPCA show significantly better results and detect the fault earlier at t=64.0h. The number of Type-I errors observed in this scenario are 11 (1.38%) for MPCA, 13 (1.63%) for DPCA, 2 (0.25%) for DisPCA, and none for AdPCA.

The summary of results for all disturbances is presented in Table 5-7. All disturbances are detected by AdPCA and DisPCA, while MPCA and DPCA fail to detect SIM05. Both the single-model PCA approaches are prone to Type-I errors with an average error rate of 1.00% for MPCA and 1.63% for DPCA. In general, DPCA detects drift/ramp disturbances faster than MPCA because of the time-lag information incorporated in the model. However, this subjects the DPCA to more false positives. DisPCA flags Type-I errors during model switching and has an average error of 0.19%. The proposed AdPCA technique gives the best performance throughout. It is able to detect all disturbances promptly with an average detection delay of 10.0h compared to 45.0h and 36.1h for MPCA and DPCA, respectively.

The robustness of AdPCA to δ and K is also evaluated. As expected, a large number of Type-I errors is observed when the models are disjoint (12 false positives when $\delta = 0$ as shown in Figure 5-22). A sharp decrease is observed for all $\delta \geq 0.04$ (only 2 false positives for all scenarios tested). Table 5-8 shows the effect of the number of clusters. Both selectivity and sensitivity improve with K. When K=1, the method reduces to MPCA and shows similar performance (64 false positives with average detection delay of 45.0h). For $K \geq 8$, the method exhibits 2 false positives and

the average detection delay improves to 10.0*h*. With a further increase (*K*=20), the average detection delay improves to 8.9*h* with 3 false positives.

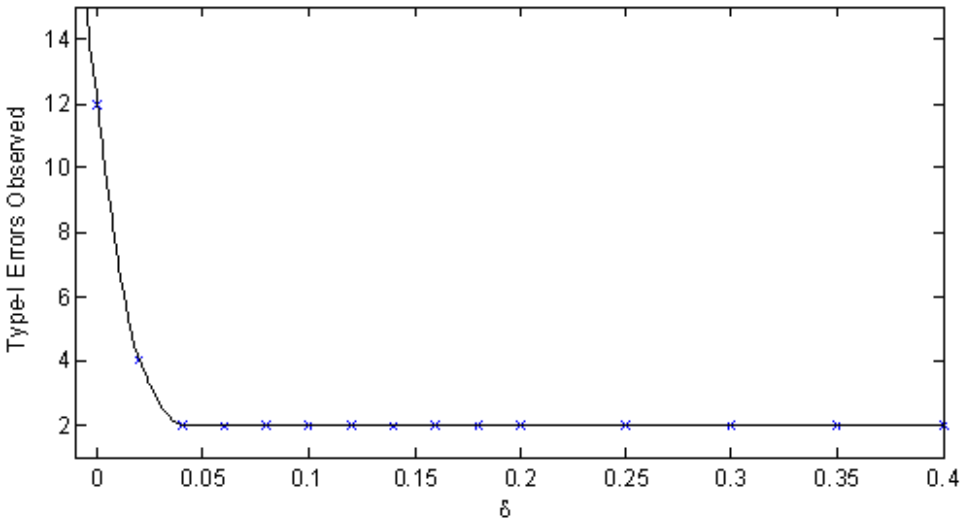


Figure 5-22: Type-I errors observed for different δ in Penicillin Cultivation Process

Table 5-6: $S^{PCA}(M_k, M_{k'})$ between PCA models for penicillin cultivation process

PCA Model #	M1	M2	M3	M4	M5	M6	M7	M8
M1	1.0000	0.6943	0.6253	0.7694	0.6048	0.5622	0.6370	0.5403
M2		1.0000	0.7009	0.6183	0.7344	0.6752	0.6097	0.6675
M3			1.0000	0.6191	0.6346	0.7287	0.5360	0.6696
M4				1.0000	0.6758	0.6680	0.6712	0.6510
M5					1.0000	0.6728	0.7574	0.5982
M6						1.0000	0.6208	0.6458
M7							1.0000	0.6187
M8								1.0000

Table 5-7: Summary of monitoring results for penicillin cultivation process

Fault ID	Time Fault Introduced	Multiway-PCA		Dynamic-PCA		Disjoined-PCA		Adjoined-PCA	
		Time Fault Detected	False Alarms	Time Fault Detected	False Alarms	Time Fault Detected	False Alarms	Time Fault Detected	False Alarms
SIM01	0.5	2.5	0	2.0	0	1.5	0	1.5	0
SIM02	0.5	5.0	0	3.0	0	2.5	0	2.5	0
SIM03	60.0	61.0	13	61.0	23	61.0	3	61.0	0
SIM04	30.0	31.0	0	31.0	0	31.0	0	31.0	0
SIM05	50.0	-	13	-	22	63.0	2	63.0	0
SIM06	70.0	86.5	13	86.5	24	82.5	3	82.5	0
SIM07	40.0	248.0	11	241.0	13	64.0	2	64.0	0
SIM08	30.0	112.0	14	59.5	8	58.5	2	58.5	2

Table 5-8: Selectivity vs. sensitivity analysis in penicillin cultivation process

Model #	**Detection delay in min (misclassified samples)**							
	***K*=1**	***K*=2**	***K*=5**	***K*=8**	***K*=10**	***K*=15**	***K*=20**	***K*=50**
SIM01	2.0 (0)	3.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.5 (0)	1.5 (0)	2.0 (0)
SIM02	4.5 (0)	2.5 (0)	1.5 (0)	2.0 (0)	2.0 (0)	2.0 (0)	1.0 (0)	2.0 (0)
SIM03	1.0 (13)	1.0 (3)	1.0 (2)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
SIM04	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)	1.0 (0)
SIM05	- (13)	188.5 (2)	32.0 (2)	13.0 (0)	13.0 (0)	13.0 (0)	13.0 (0)	13.0 (0)
SIM06	16.5 (13)	17.0 (4)	12.5 (2)	12.5 (0)	12.5 (0)	12.5 (0)	12.5 (0)	8.5 (0)
SIM07	208.0 (11)	197.5 (2)	108.0 (2)	24.0 (0)	24.0 (0)	24.0 (0)	24.0 (0)	24.0 (0)
SIM08	82.0 (14)	33.0 (2)	36.5 (3)	28.5 (2)	28.5 (2)	26.5 (2)	26.5 (3)	24.0 (3)
Avg Delay	**45.0**	**36.4**	**23.1**	**10.0**	**10.0**	**9.8**	**9.6**	**8.9**

5.9 Summary

Online monitoring of transient operations is important to detect abnormal events and enable timely recovery. Transient operations commonly follow a nonlinear, discontinuous trajectory in the principal components subspace. Therefore, classical PCA approaches such as MPCA and DPCA are unable to adequately model such operations. The run-length variations common across different instances of transient operations restricts the application of time-wise unfolding methods with MPCA unless explicit synchronization of trajectories is performed (Doan and Srinivasan, 2007). Further, reliable estimation of the underlying distribution of a multivariate transient operation is difficult. These difficulties are overcome when multiple models are used for modeling such non-stationary, and non-Gaussian systems. When the multiples models are disjoint, they become prone to Type-I errors, especially in the interregnum between models. To overcome this, a new monitoring technique, called Adjoined PCA, based on overlapping PCA models has been developed. In the proposed technique, multiple overlapping PCA models are constructed using membership information obtained though fuzzy clustering. Each constituent PCA model in AdPCA is allowed to overlap with its neighbors – this enforces continuity while modeling and monitoring transient operations. The monitoring statistics constructed for each constituent model hence comprehensively covers all relevant portion of the training data, so false positives do not occur during model switching. Additionally, since the proposed AdPCA classifies the region of the training data more accurately, unknown regions in the PC subspace are better excluded from the model. This reduces the occurrences of Type-II errors as well. This is not achievable with single model methods where a tradeoff between selectivity and sensitivity occurs. During online application, the optimal PCA model is selected at every instant and the process state monitored based

on it. The proposed AdPCA method is hence applicable to operations with different run-length since it switches among the models automatically based on the distance of the sample from the different models. Therefore, it can inherently adapt to varying durations by extending or reducing the period for which a model is used. Extensive testing of the proposed method using two case studies clearly demonstrates the method's ability to red reduce both Type-I and Type-II errors.

Nomenclature

Indices

c	principal components
i	process time (rows of X)
k, k'	model within a bank of models
n	process variables

Parameters

$\mathbf{1}$	identity matrix
C	total number of principal components retained
I	total number of samples (rows)
K	number of clusters used to construct AdPCA model
l	time-lag incorporated in dynamic-PCA model
N	total number of variables (columns)
v	fuzzifier used during fuzzy c-means clustering
β	weighting factor used for combining $T^2$and SPE
δ	adjoining threshold

Variables

B_i^k	the k^{th} best cluster that data x_i is belonging to
c_k	centroids of k^{th} cluster
G_k	a matrix composed of data from X that belong to cluster k
M_k	k^{th} PCA model
M^{opt}	nearest PCA model to a new sample x_i
p_c	loadings
$Q(\alpha)$	upper control limit for *SPE* at confidence level 1- α

$S_{kk'}^{PCA}$	similarity observed between two PCA models, namely M_k and $M_{k'}$
SPE_i	squared prediction error for sample x_i
SPE_{ik}	reduced *SPE* observed for sample x_i based on k^{th} model
t_c	scores
$T^2(\alpha)$	upper control limit for T^2 statistic based on 1- α confidence level
T_i^2	Hotelling's T^2 statistic for sample x_i
T_{ik}^2	T^2 observed for sample x_i based on k^{th} model
u_{ik}	membership of x_i in each k group identified from clustering algorithm
v	fuzzifier used for c-means clustering
X	a multivariate time series data
x_i	i^{th} sample in X
$\tilde{Y}^{3d}$	a three-dimensional training data collected from plant historian
$\tilde{Y}$	a two-dimensional raw training data collected from plant historian
Y	auto-scaled data used for AdPCA model training
Λ	covariance matrix of X
λ_c	eigenvalue associated with p_c
σ_{Y_n}	standard deviation for variable Y_n
Φ_{ik}	combined discriminant similarity index

Chapter 6 Pattern Recognition based on Binomial Combination of Non-parametric Confidence Bounds

6.1 Need of Non-parametric Approach for Fault Recognition

A diagnostic classifier based on Kernel Density Estimator (KDE) is developed in this chapter for disturbance detection and identification. The proposed method can be used to substitute Hotelling's T^2 statistic in detecting deviations from reference PCA models. As described in Chapter 2.5, the popular Hotelling's T^2 statistic sustains unusual amount of Type-I and Type-II errors when applied to transient processes. Since obtaining a general data density model for all types of transient operations is difficult, a non-parametric approach such as KDE that does not require a prior specification of any parametric density model can be used. Given its non-parametric form, KDE-based method can be applied to both the domain of steady-state and temporal phase as in the case of transitions/batch operations. Reference data can first be projected to a latent subspace and the reference model created. Since KDE is bi-variate in nature, different combinations of the latent variables have to be analyzed and merged for multi-dimensional analysis. The KDE-based statistic has been shown to be an effective tool for fault identification by rendering a database search possible through pattern recognition. The use of fault database for fault identification is justified through the continuous growth in availability of quality database in the process industries, either through long period of accumulated operations or development of high-fidelity simulator by the process engineers. Additional information such as fault

description and rectification strategies can also be associated with the fault-database to facilitate fault recovery in real-time.

6.2 Fault Diagnosis based on KDE

KDE is a density estimation technique that creates bound based on the training data without the imposition of any priori knowledge of the distribution of the training data, eg: Gaussian, *F*-distribution, etc. The monitoring bounds constructed can thus describe the behavior of process transitions more accurately. It can hence be used to substitute the Hotelling's T^2 statistic in detecting deviations from a trained model during process transitions.

Though the proposed approach can be applied to most statistical model-based approaches, multivariate statistical methods is used here for illustration, specifically PCA. As noted earlier, since DPCA can capture the dynamics of process variables, DPCA model was developed for a transient process and then construct various KDE-bounds for all combinations of the bi-variate PC pairs observed.

The KDE bounds are created based on the projection of normal operating data in the score space. Let the score matrix from a PCA/DPCA model be $t = \{t_1,...,t_j,...,t_J\}$, where t_j being the j^{th} column of t with J number of columns. For the purpose of monitoring, a total of ${}^{J}C_2 = 2!/((N-2)!2!)$ bounds can be constructed through KDE based on the *binomial combinations* from t_j vs $t_{j'}$ pair, $\forall j, j' = \begin{pmatrix} j \\ j' \end{pmatrix}$, by selecting two components from the scores matrix t in each combination. Let the collection of these KDE bounds be noted as $M^{jj'} \in M^{\Lambda}$, where each $M^{jj'}$ corresponds to the binomial j *vs* j' combination of the score of principal components. Here, Λ is the collection of subsets for the binomial coefficients of the score matrix t, i.e.,

$\Lambda = [1,2; 1,3; ...; j-1, j]$. Each of the bound created within M^{Λ} , i.e., $[M^{1,2}, M^{1,3}, ..., M^{j-2,j}, M^{j-1,j}]$, corresponds to a contour in the two dimensional space and inspects variations in plane $t^{jj'}$, $j \neq j'$.

Since the constructed bounds are very dependent on the bandwidth matrix, *H*, an additional scaling factor, Φ, was introduced to the *H* obtained to allow adjustment of the width of $M^{j,j'}$ used for monitoring:

$$H^{opt} = \Phi \times H \qquad \text{(Eq 6-1)}$$

Here, H^{opt} is the final smoothing parameters used for bounds generation. The scaling factor, Φ, plays an important role in adjusting the selectivity and sensitivity of $M^{j,j'}$. A $\Phi < 1$ makes the KDE bound $M^{j,j'}$ tighter and more sensitive, hence reduces the rate of Type-II errors (false negatives) while $\Phi > 1$ makes $M^{j,j'}$ larger, hence less prone to Type-I errors (false positives).

As noted in Chapter 2, the amount of variance described by each PC differs significantly, *i.e.*, abnormality shown in the scores from the first few PCs normally indicate a more severe indication of faults compared to subsequent PCs. In order to accommodate such differences in the significance of PCs, their corresponding eigenvalues can be used as a weighting function. Since there exist ${}^{J}C_{2}$ bounds for a PCA/DPCA model with *J* latent variables (PCs), a sample *x* has to be investigated with all of the KDE bounds created, $M^{j,j'}$. In this work, an index was proposed to combine the results from all dimensions into a single, unified index by weighting each dimension in the binomial combination with their corresponding eigenvalues.

In the simple two dimensional case involving PC *j* and PC *j'*, the *distortion index*, τ, which measures the total amount of undesired changes in the PCs domain can be defined as the weighted Euclidean distance of the bivariate score of the online

sample, $t_i^{j,j'}$, compared to the closet point identified from the scores of the center trajectory (center line of the training data), and the closest point identified from the KDE bound $M^{j,j'}$. Let $C_*^{j,j'}$ be the closest point of the center trajectory from $t_i^{j,j'}$, and $M_*^{j,j'}$ be the closest point of the KDE bound $M^{j,j'}$ from $t_i^{j,j'}$ (see Figure 6-1), both $C_*^{j,j'}$ and $M_*^{j,j'}$ can be identified through:

$$M_*^{j,j'} = \arg\min_r(|\, t_i^{j,j'} - M_r^{j,j'}\,|) \qquad \text{(Eq 6-2)}$$

$$C_*^{j,j'} = \arg\min_{i'}(|\, t_i^{j,j'} - C_{i'}^{j,j'}\,|) \qquad \text{(Eq 6-3)}$$

Since the information contain in each dimension of the PC is proportional to their corresponding eigenvalue of the covariance matrix, the distance across each PC was weighted with its corresponding eigenvalue. The distortion index, τ, which calculates the relative distance between $t_i^{j,j'}$ to points $C_*^{j,j'}$ and $M_*^{j,j'}$ can then be evaluated as:

$$\begin{aligned}\tau_i &= \frac{d(t_i^{j,j'}, C_*^{j,j'})}{d(M_*^{j,j'}, C_*^{j,j'})} \\ &= \sqrt{\frac{(\lambda_j(t_x^{j,j'} - C_{*x}^{j,j'}))^2 + (\lambda_{j'}(t_y^{j,j'} - C_{*y}^{j,j'}))^2}{(\lambda_j(M_{*x}^{j,j'} - C_{*x}^{j,j'}))^2 + (\lambda_{j'}(M_{*y}^{j,j'} - C_{*y}^{j,j'}))^2}}\,.\end{aligned} \qquad \text{(Eq 6-4)}$$

Here, the λ_j, $\lambda_{j'}$ is the eigenvalue for PC *j* and *j'*, while the subscript *x* and *y* denote their corresponding *x* and *y* coordinate in the PC space (see Figure 6-1). In the above two-dimensional scenario, the $\tau_i \approx 0$ if the bivariate scores $t_i^{j,j'}$ are close to the scores of the center trajectory of the training data, $\tau_i \le 1$ if the sample is within the KDE bounds identified, while $\tau_i > 1$ if $t_i^{j,j'}$ is exceeding the allowable limits of the KDE bounds.

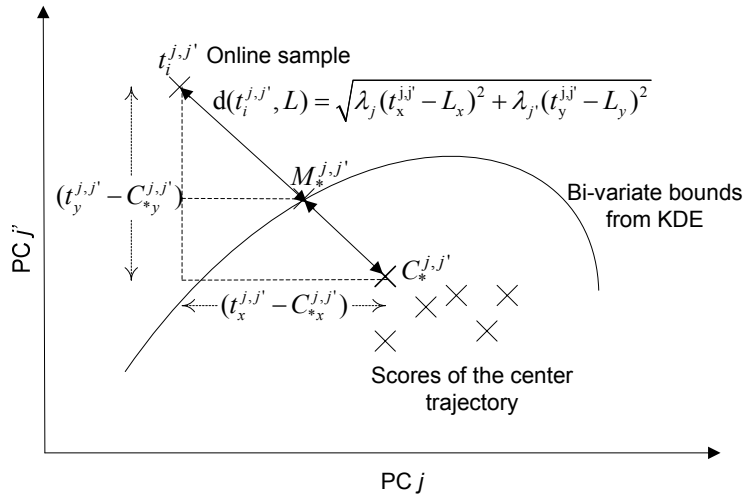


Figure 6-1: Illustration of relative distance calculation between (i) $t_i^{j,j'}$ and $C_*^{j,j'}$, and (ii) $M_*^{j,j'}$ and $C_*^{j,j'}$

When J dimensions are involved, a total of ${}^{J}C_2$ bounds are produced and their monitoring results have to be combined. For cases involving J PCs, the τ at time i instant can be evaluated as:

$$\tau_i = \sqrt{\frac{\sum_{j=1}^{J}\sum_{j'=1}^{J'}\left((\lambda_j (t_x^{j,j'} - C_{*x}^{j,j'}))^2 + (\lambda_{j'} (t_y^{j,j'} - C_{*y}^{j,j'})^2)\right)}{\sum_{j=1}^{J}\sum_{j'=1}^{J'}\left((\lambda_j (Q_x - C_{*x}^{j,j'}))^2 + (\lambda_{j'} (Q_y - C_{*y}^{j,j'})^2)\right)}}, \quad \forall j, j' = \begin{pmatrix} j \\ j' \end{pmatrix} \in [1,\ J] \quad \text{(Eq 6-5)}$$

where $\begin{pmatrix} j \\ j' \end{pmatrix}$ corresponds to the binomial coefficients of J variables.

When analyzed across all $t^{jj'}$ pairs, the τ_i quantifies the total amount of undesired changes of x_i from all nominal $t^{jj'}$ pair in the PC subspace. The summation of the denominator in Eq 6-5 serves as a normalization function for τ. An upper control limit τ^{β} can be defined for τ for the purpose of monitoring.

6.3 Pattern recognition through fault distortion index

The proposed index can also be used as a pattern recognition metric. Upon the detection of fault, the fault patterns or features observed in real-time can be compared

with previous runs to predict the type of fault. Fault isolation is equivalent to pattern matching with the available multidimensional datasets. The fault isolation scheme considered here is based on the precept that different classes of faults always show unique fingerprints in the score space. The distortion index, τ, described in previous section can then be used for pattern recognition through discrimination of faults in the score space.

Here, a statistic-based fault reconstruction approach was used for fault identification. Fault identification is based on measuring the similarity between the online measurements x_i with each F_w in the fault database. Let there be W classes of known faults, F_w, where $w \in [1\ ,\ W]$. The PCA model, PCA_w, that shows an in-control status for majority of the fault samples observed can be identified as the possible candidate for x.

Multiple KDE bounds M^{F_w} can be constructed for each PCA_w with the distortion index between x and each M^{F_w} quantified. The similarity between x_i and each M^{F_w}, $w \in [1\ ,\ W]$, is computed as a fault similarity measurement, S_{iw}:

$$S_{iw}(x_i, M^{F_w}) = \frac{\psi_{iw}}{L_i},\ \forall w \in [1\ ,\ W] \qquad \text{(Eq 6-6)}$$

Here, ψ_{iw} is the number of samples within an evaluation window (from the time of fault detection till time *i*) that show in-control status with M^{F_w}, and L_i is the length of the evaluation window used. The ψ_{iw} can be computed at each instant as:

$$\psi_{iw} = \psi_{(i\text{-}1)w} + \begin{cases} 1\ , & if\ \ \tau_{iw}(x_i, M^{F_w}) < \tau^{\beta} \\ 0\ , & otherwise \end{cases},\ w = [1\ ,\ W] \qquad \text{(Eq 6-7)}$$

The best matching fault candidate from the fault database at time *i*, F_i^{opt}, can then be extracted as the best matching candidate:

$$F_i^{opt} = \arg\max_w (S_{iw}),\ w = [1,\ W]. \qquad \text{(Eq 6-8)}$$

The fault can be accurately identified if S_w shows high level of similarity across the evaluation window used. In general, the initial stages of similarity search might return more than one successful candidates as there might exist multiple fault models M^{F_w} that show high degree of overlap in their PC subspace. However, this ambiguity is usually resolved with the analysis of additional samples.

6.4 Implementation Algorithm

The KDE-based PCA monitoring and diagnosis method is implemented in practice as follows:

Offline Training:

Step 1: *Normalization*: Each variable of X^D is first autoscaled to alleviate the varying scales of different process variables.

Step 2: *Latent space projection*: A dynamic matrix X^D is constructed from the autoscaled training data through Eq 2-14. The X^D is then projected to the principal components subspace as scores, *t*, and loadings, *p*.

Step 3: *KDE Bounds Construction*: A collection of KDE bounds, M^{Λ}, is constructed for different binomial coefficients of scores $t^{jj'}$, $\forall j, j' = \begin{pmatrix} j \\ j' \end{pmatrix}$ obtained from X_n^D. The bandwidth selection technique as described in Appendix D is used for calculation of the bandwidth matrix, *H*. *H* can be further fine-tuned with a scaling factor, Φ, as described in Chapter 6.2. Monitoring bounds correspond to 95% and 99% contours are then constructed based on the estimated density of the scores $\hat{f}(t, H^{opt})$. The algorithm

for offline training is shown in Figure 6-2. An upper control limit, τ^{β}, is selected and used for fault detection based on the anticipated noise level.

Step 4: *Fault database preparation*: The available training data for different classes of faults, F_w, is organized and stored in a fault database, DB, where $DB = \{F_1 \cup ... \cup F_w\}$. Each class of the fault is PCA/DPCA projected and the bounds M^{F_w} constructed for each F_w in DB similar to Steps 1 to 3.

Online Monitoring &Fault Diagnosis

Step 1: *Online monitoring*: New process measurements, designated as x_i, is first (a) augmented with time-lag information, and then (b) autoscaled before being projected to the PCA model. The score for x_i, designated as t_i, in the PC subspace is calculated based on the loadings obtained from offline training.

Step 2a: *Fault detection*: The distortion index at time *i*, τ_i, is quantified by combining all in-control/out-of-control status of all available KDE bounds (Eq 6-5). The underlying process is deemed abnormal if $\tau_i > \tau^{\beta}$.

Step 2b: *Fault diagnosis*: Fault diagnosis involves iterating over the fault database, DB, to identify the root cause based on the available fault patterns (see Figure 6-3). The fault similarity measurement, S_w, $w = [1, W]$ is calculated for each F_w by comparing x_i^D with each fault model constructed, M^{F_w}, through $\tau_{iw}(x_i, M^{F_w})$ over a defined time window. The fault candidates that contain high degree of similarity (e.g.: $S_w > 80\%$) are listed as possible candidates.

In the following sections, the efficacy of the proposed metrics are illustrated using two case studies: a fed-batch penicillin cultivation testbed, and a pilot-scale distillation unit case-study.

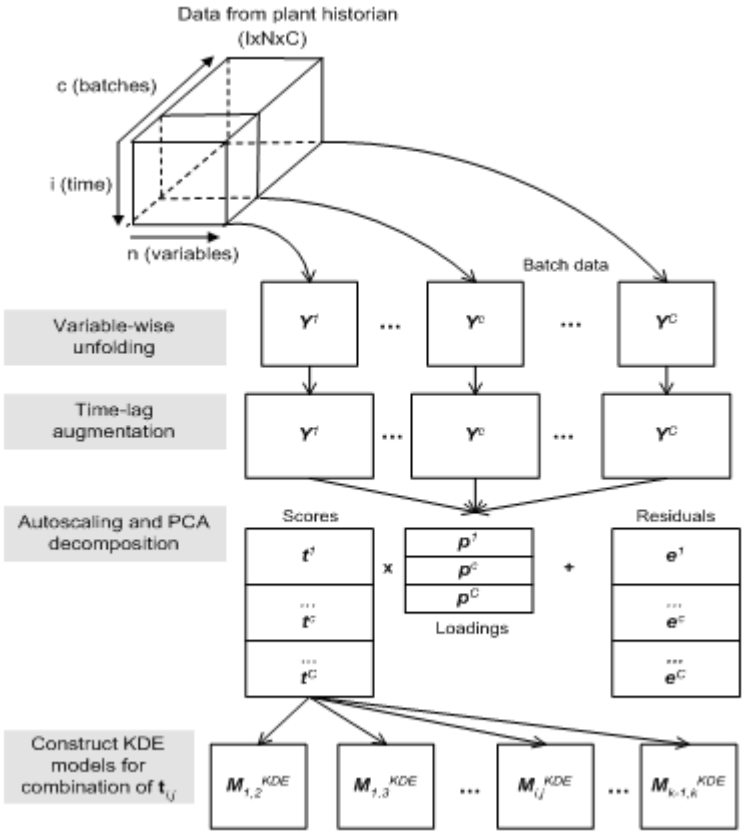


Figure 6-2: Offline methodology for kernel-density models construction

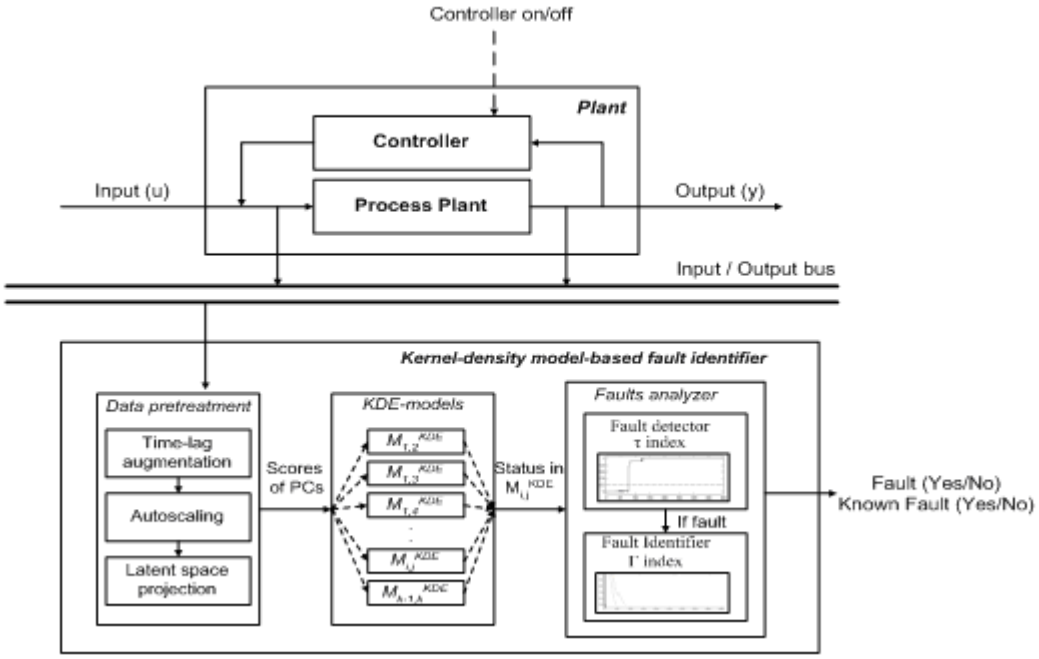


Figure 6-3: Online architecture for kernel-density model-based fault diagnosis

6.5 Case Study 1: Fault Diagnosis during Penicillin Cultivation

Ten normal fed-batches were simulated to create the reference data (with an integration step size of 0.02*h* and a sampling interval of 0.5*h*). The variables selected for the monitoring of penicillin cultivation process are listed in Table 5-5. The state variables during each normal run are highly nonlinear and many of them show strong discontinuities during the period of cultivation. In all runs, an initial batch culture is followed by a fed-batch operation based on the depletion of the carbon source (glucose). The process switches to the fed-batch mode of operation when the level of glucose concentration reaches 0.3g/l.

Next, the PCA model is constructed using the training data based on the state variables listed in Table 5-5. A DPCA model with time lag l =2 was trained. Eight PCs are needed to capture enough variance of the training data based on criteria $\sum_{j=1}^{J}\lambda_j \Big/ \sum_{j=1}^{N}\lambda_j > 95\%$. The final constructed DPCA model captures 98.77% variance of the fed-batch operations. In order to develop the KDE bounds, different binomial combinations of the PCs are used. A total of 28 KDE bounds are created based on the bivariate scores of the eight PCs, i.e., the KDE bounds M^{Λ} are hence consisted of $M^{\Lambda} = \{M^{1,2}, M^{1,3}, ..., M^{j,j'}, ..., M^{7,8}\}$. The optimal bandwidth, H^{opt}, is determined based on the normal scale rule, H_{AMISE}, by setting $\Phi = 1$ using Epanechnikov kernel. Based on Eq D-3, the values of *H* for the first two PCs are identified as [1.618, 1.371], and the monitoring bounds are created by using the contour of the estimated density from KDE.

When tested with a control dataset, the normal fed-batch operation is represented as a trajectory on the scores plot of the PCs. The monitoring bounds based

on KDE orient themselves around the trajectory produced, and occupy only the relevant normal operating region in the score space.

Eight process disturbances were studied as summarized in Table 5-4. All training data that are used for fault identification are simulated separately. The training data used for pattern recognition hence consist of eight groups. In this study, the performance of Hotelling's T^2 and bivariate-KDE based on PC1 and PC2 are compared with the proposed methodology. Two runs correspond to Run-06 and Run-07 are illustrated next.

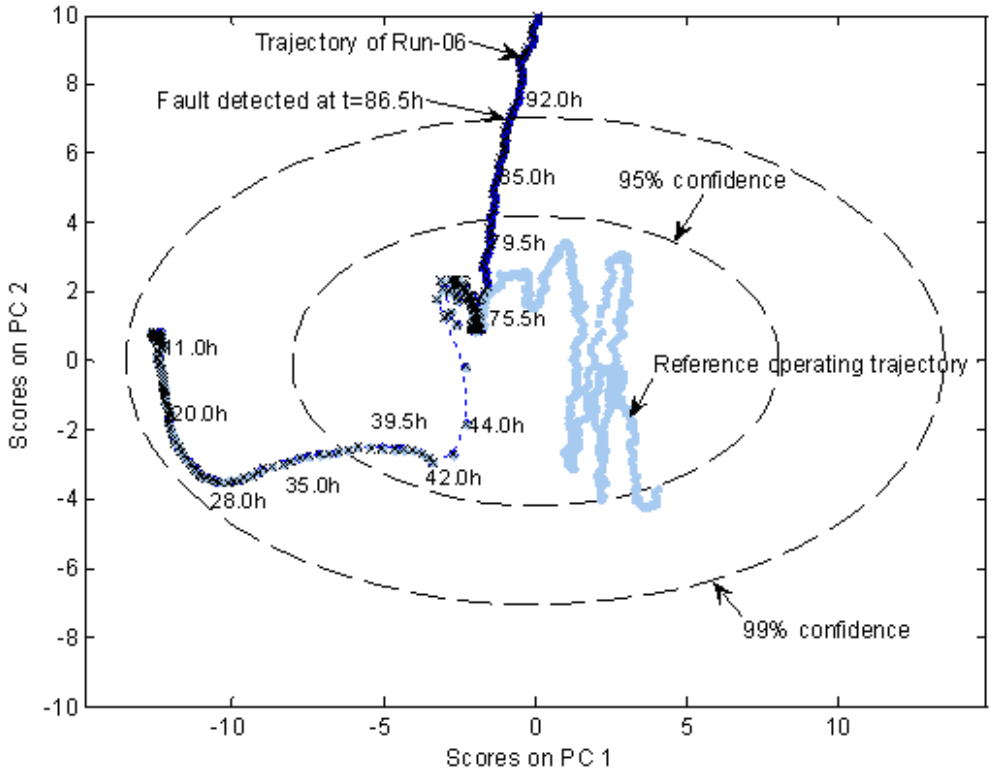
Scenario 1: Monitoring of a ramp increase in aeration rate

Run-06 corresponds to a ramp increase in aeration rate with a slope of +0.05. The fault was introduced at t=70.0h, by allowing the oxygen flow to increase continuously over time. The monitoring bounds of Hotelling's T^2 are reflected as ellipses in the PC subspace (see Figure 6-4a & 6-4b). The ellipses produced based on Hotelling's T^2 (based on 99% confidence) cover a large area on the PC subspace, rendering the method prone to Type-II errors. When the process trajectory deviates the 99% confidence bound, Hotelling's T^2 flags the fault at t=86.5h.

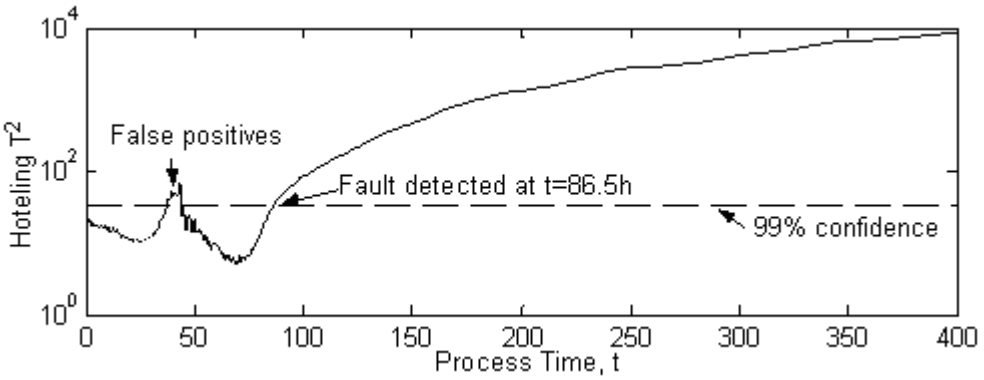
In contrast, bivariate-KDE and the proposed KDE approach generate bounds based on the presence/absence of training data in the PCA score space. The bounds created based on 99% confidence limit thus fit the reference data well and hence avoids the occurrence of both Type-I and Type-II errors. The KDE bounds for the first two PCs are shown in Figure 6-5a. In this scenario, the same fault can be detected 7.5h earlier at t=79.0h (bivariate-KDE with PC1 vs PC2) by improving the bounds used for process monitoring.

The same fault is also monitored using the proposed distortion index, τ, based on 99% confidence limit. The τ gives similar performance compared to bivariate KDE

and a better performance compared to T^2 in this scenario by flagging the fault at an earlier time without any false positives (see Figure 6-5b). The improvement in detection speed over T^2 is due to usage of better monitoring bounds as the bounds from KDE give better representation of the training trajectory in the PC subspace.

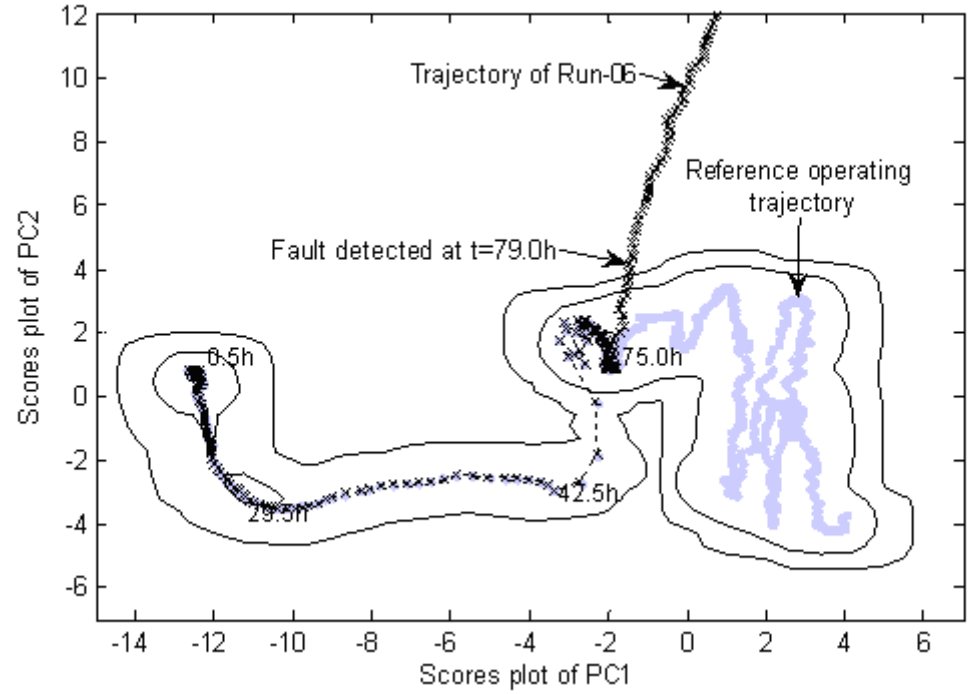


(a)

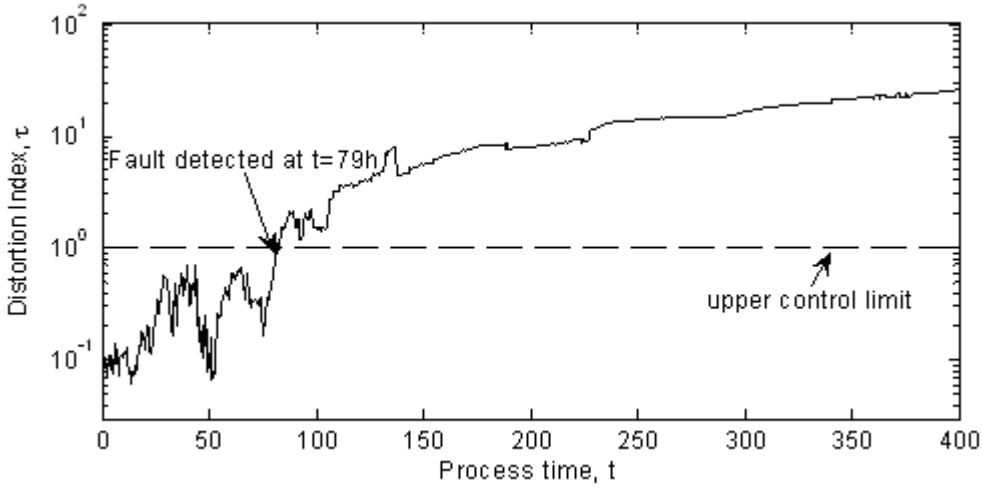


(b)

Figure 6-4: Monitoring on Run-06 of penicilin cultivation case study using Hotelling T^2



(a)



(b)

Figure 6-5: Monitoring on Run-06 of penicilin cultivation case study using KDE and Distortion Index

In order to test the fault isolation property of the proposed τ statistic, the online signals are used for fault identification based on a pattern recognition approach by matching the faults with known fault patterns in the database. For the purpose of fault diagnosis, the τ index is generated for each available fault patterns and its similarity with all faults S_{iw} generated. Three fault candidates, namely, SIM06, SIM07,

and SIM08 are identified as potential candidates (see Figure 6-6) at time of fault detection as initial phase of these three classes of faults are similar given the ramp type of fault considered (characteristic of the fault become more obvious with t). Eventually, the similarity degree of SIM07, S_7, and SIM08, S_8, drops below the threshold (0.8) after the analysis of 46 additional samples, and enables the successful isolation of the fault at t=102h. The proposed τ statistic hence provides good classification property to segregate the fault classes in the PC subspace. In contrast, isolation based on bivariate KDE (using PC1 and PC2) fails to distinguish SIM03 from SIM06 (Figure 6-7) since a significant portion of the bound for SIM03 is overlapping with the bound of SIM06 in the PC1 and PC2 subspace. The SIM03 corresponds to a step change while SIM06 corresponds to ramp change of the aereation rate variable.

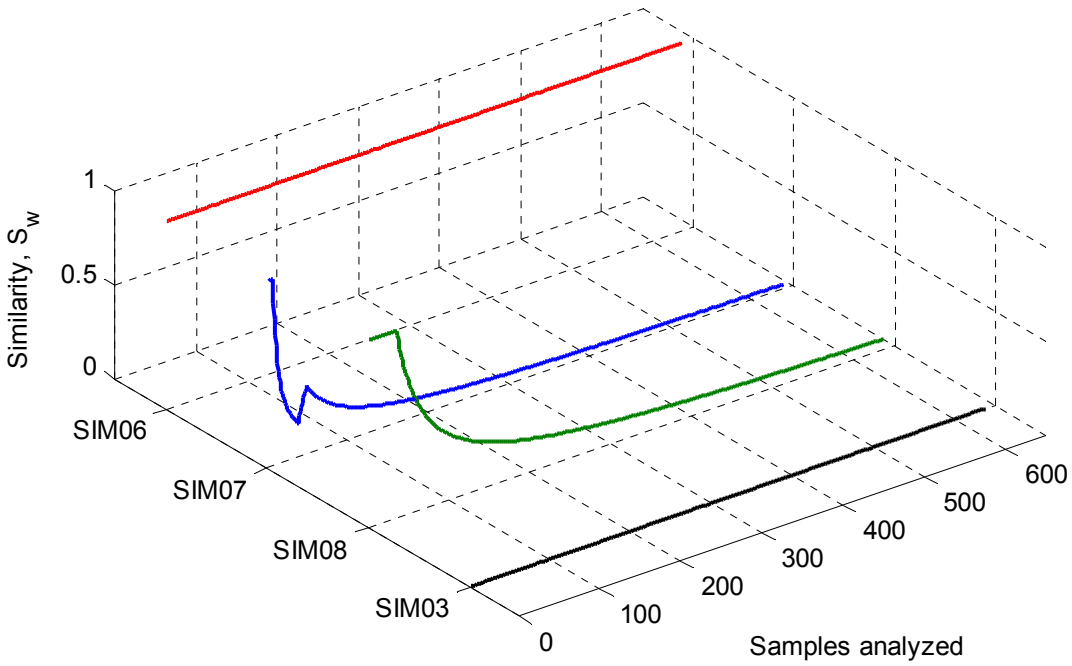


Figure 6-6: Fault isolation during Run-06 using similarity based on DI method

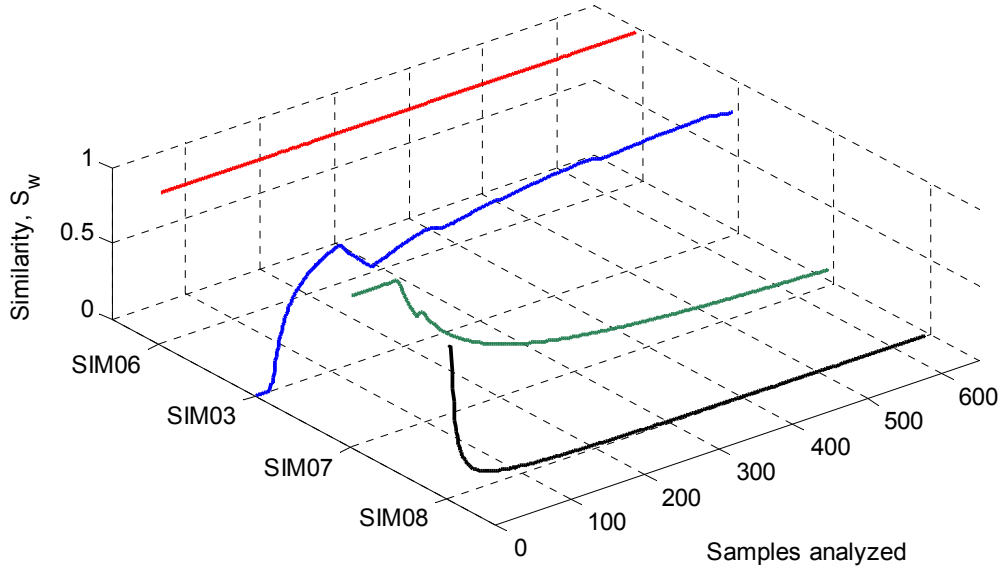


Figure 6-7: Fault isolation during Run-06 using similarity based on first two PCs

This shows the need of combining various combinations of PCs to improve the class discrimination property of KDE. Using additional combinations of PCs help to improve the separability of the fault classes. Since it is very difficult to visually monitor so many different bivarate combinations of KDEs, the proposed approach provides a convenient way of automating the analysis of KDE plots and facilitates its actual implementation for multivariate analysis in practice.

Scenario 2: Monitoring of a ramp increase in agitation input power

Run-07 corresponds to a ramp increase in agitation input power starting at t=40.0h which increases the shear on the biomass. Hotelling's T^2 causes significant delay in fault detection time with a delay of 198.0*h* from time of fault introduction (Figure 6-8). The detection speed of Hotelling's T^2 can be improved if a 95% confidence limit is used. However, using low confidence for control limit subjects the initial trajectory of the operations prone to false positives (Figure 6-8). In contrast, the KDE-based approaches successfully detects the fault at *t*=133.0h without any Type-I

errors (Figure 6-9) when the online trajectory deviates from the 99% confidence limit of the model.

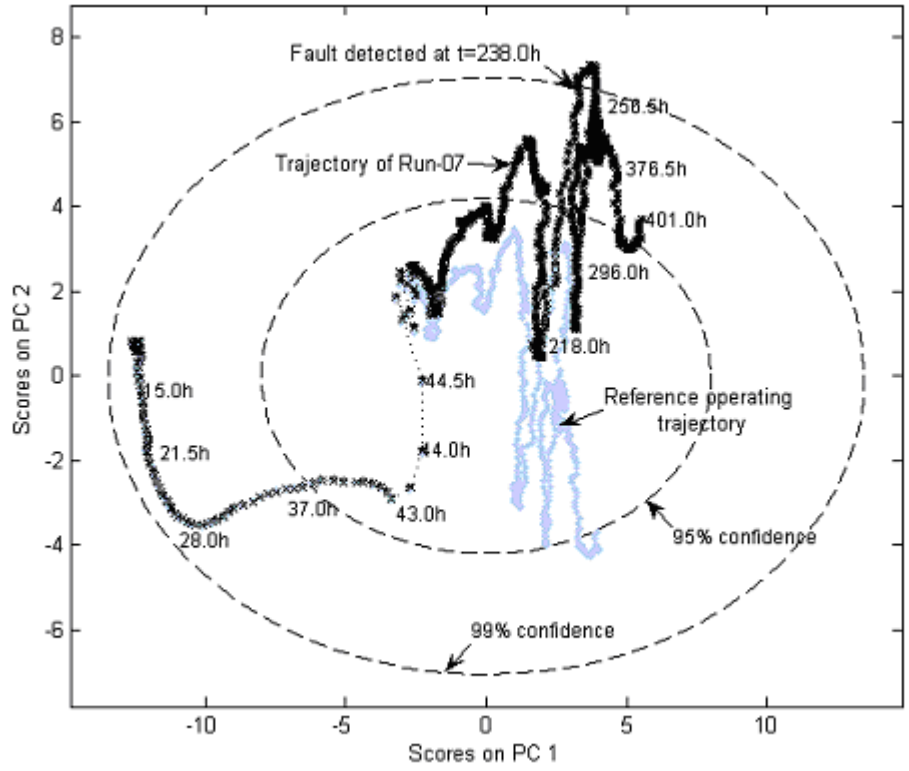


Figure 6-8: Monitoring on Run-07 of penicilin cultivation case study using Hotelling T^2

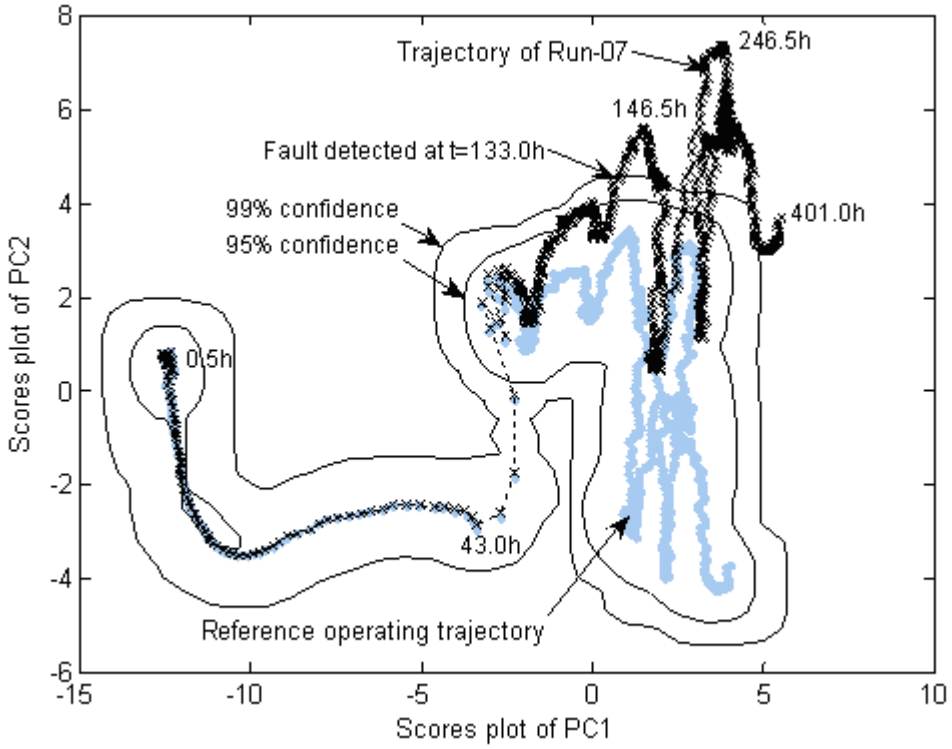


Figure 6-9: Monitoring on Run-07 of penicilin cultivation case study using KDE

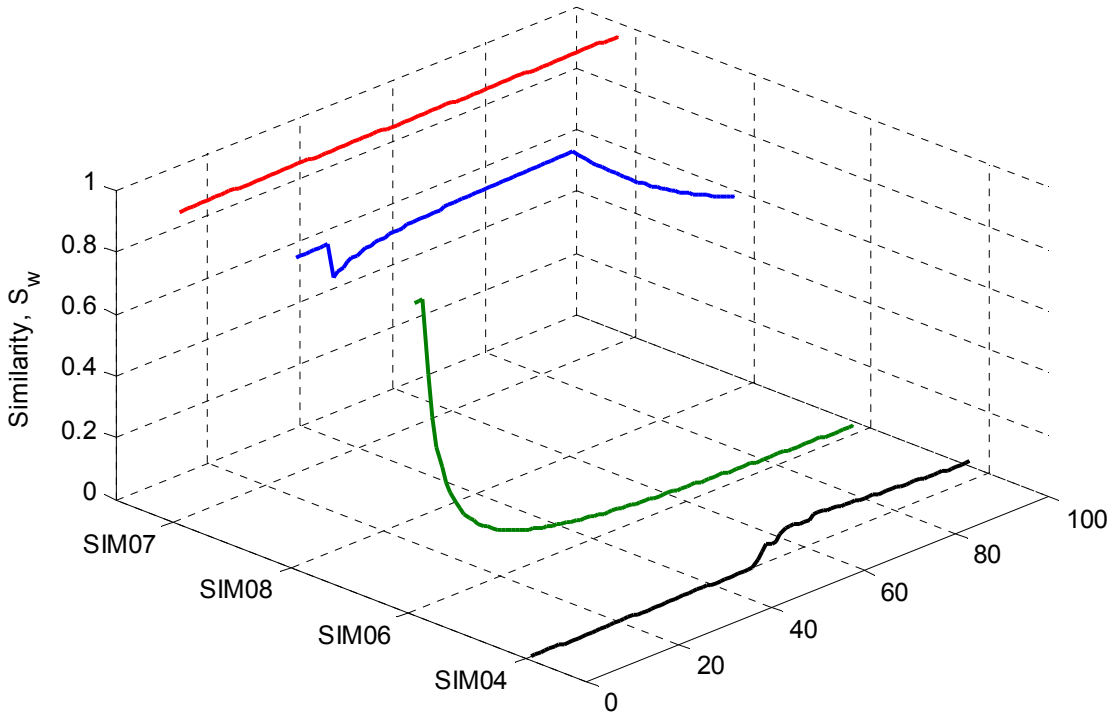


Figure 6-10: Fault isolation during Run-07 using similarity based on DI method

Fault identification based on the online signature using bivariate KDE identifies four candidates – SIM04, SIM06, SIM07, and SIM08 at time of fault detection. The fault is isolated at t=238.5h when the similarities of other faults drop below the threshold (S_w<0.8). The proposed fault isolation approach using binomial combination of different PCs improves the initial selection of fault candidate and fault isolation time. SIM06, SIM07 and SIM08 are identified as potential candidates at time of fault detection (see Figure 6-10). The time of fault isolation also improves to t=170.0h (18.5h earlier) for the fault similarity of other candidates to drop below the selected threshold (S_w>0.8).

Table 6-1: Monitoring results for penicillin cultivation process

Run #	Time fault introduced	Hotelling T^2		Bivariate-KDE		DIBC	
		Time of detection (h)	Detection Delay (h)	Time of detection (h)	Detection Delay (h)	Time of detection (h)	Detection Delay (h)
Run-01	0.5	2.0	1.5	1.5	1.0	1.5	1.0
Run-02	0.5	20.0	19.5	7.5	7.0	7.5	7.0
Run-03	60.0	60.5	0.5	60.5	0.5	60.5	0.5
Run-04	30.0	30.5	0.5	30.5	0.5	30.5	0.5
Run-05	50.0	-	-	50.5	0.5	50.5	0.5
Run-06	70.0	86.5	16.5	79.0	9.0	79.0	9.0
Run-07	40.0	238.0	198.0	133.0	93.0	133.0	93.0
Run-08	30.0	177.5	147.5	107.0	77.0	86.0	56.0
Avg Delay			**54.86h**		**26.86h**		**23.86h**

Table 6-2: Fault diagnosis results for penicillin cultivation process

Run #	Bivariate KDE			DIBC		
	Fault Candidates	Time Fault Isolated	Isolation Delay	Fault candidates	Time fault isolated	Isolation Delay
Run-01	F1	1.5	1.0	F1	1.5	1.0
Run-02	F2	7.5	7.0	F2	7.5	7.0
Run-03	F3	60.5	0.5	F3	60.5	0.5
Run-04	F4	30.5	0.5	F4	30.5	0.5
Run-05	F5,F8	205.0	155.0	F5,F8	205.0	155.0
Run-06	F3,F6,F7,F8	-	-	F6,F7,F8	102.0	32.0
Run-07	F4,F6,F7,F8	238.5	198.5	F6,F7,F8	170.0	130.0
Run-08	F4,F7,F8	131.5	101.5	F7,F8	94.0	64.0
Avg Delay			**66.28h**			**51.14h**

The summary of monitoring and fault diagnosis to other scenarios of the fed-batch penicillin cultivation case study is shown in Table 6-1 and Table 6-2. It was observed that the occurrence of the phase shift in each batch is around 86th–92nd samples. The Hotelling's T^2 identifies six of the disturbances while the bivariate-KDE and the proposed KDE-based approach identify all of the disturbances introduced.

Hotelling's T^2 is unable to detect the process disturbance of Run-05, which corresponds to a step decrease in substrate feed rate. The average detection delay based on Hotelling's T^2 is 54.8h for all detectable faults, while the average detection delay for the bivariate KDE and the proposed KDE-based approach is 26.9h and 23.9h

respectively (see Table 6-1) based on the same classes of faults detected through Hotelling's T^2. Also, kernel density-based approaches are able to reduce the rate of Type-I and Type-II errors compared to Hotelling's T^2. This improvement is due to the better bounds created for the training data by the KDE approach. The proposed KDE-based approach is able to detect the faults faster compared to bi-variate KDE by incorporating additional information from other PCs

In this case study, all faults are correctly identified and isolated with the proposed KDE-based fault pattern recognition approach with an average diagnosis delay of 51.14*h* from time of fault introduction. Bivariate KDE-based approach fails to isolate fault SIM06 and requires an additional 15.14*h* (avg delay of 66.28h from time of fault introduction) to isolate the faults in average (Table 6-2). In summary, the proposed approach based on binomial combination of KDE bounds give the best results for both monitoring and fault identification compared to Hotelling's T^2 and bivariate-KDE.

6.6 Case Study 2: Fault Diagnosis during Distillation-unit Startup

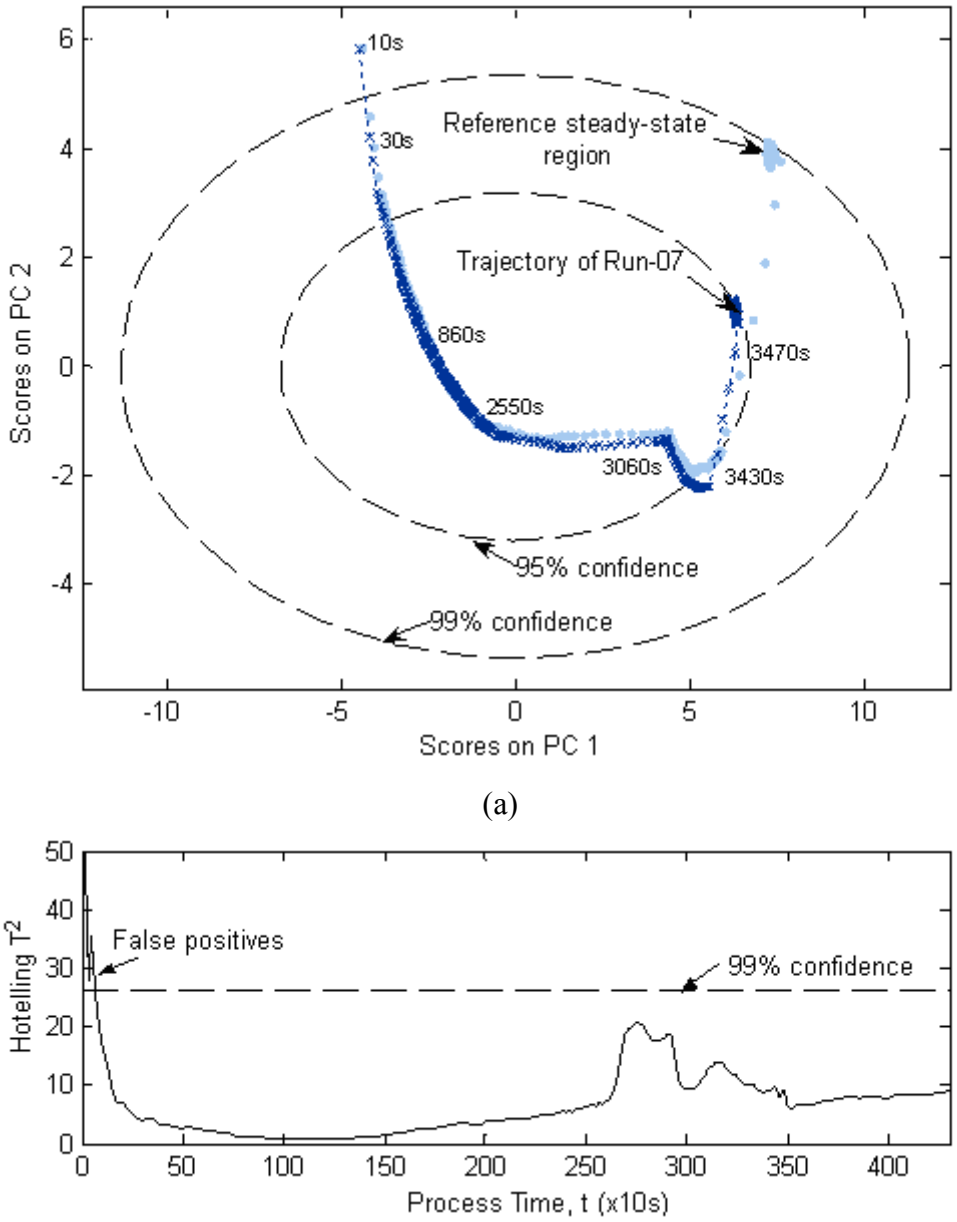
The proposed FDI method was also tested with the pilot-scale distillation unit. Nominal dataset from six normal startups are used as training data. The training data set is augmented with time-lag samples (l=2), before they are projected to the PC subspace. Eight PCs are retained with a total cumulative variance of 95.52%. Similar to previous case study, a total of 28 KDE models are created based on the reference data for monitoring the startup operation.

The optimal bandwidth, H^{opt}, is determined based on the *least squared cross-validation,* H_{LSCV}, by setting the scaling factor, $\Phi = 0.8$. The calculated H^{opt} for the KDE bound in the first two PCs are $H^{opt} = [0.005, 0.008]$. Ten process disturbances

have been tested in this study with their summary given in Table 3-2. The training data is first created by conducting separate experiments with their measurements taken at 10-second interval. Two of the faults are described in detail.

Scenario 1: Monitoring Low Reflux Ratio

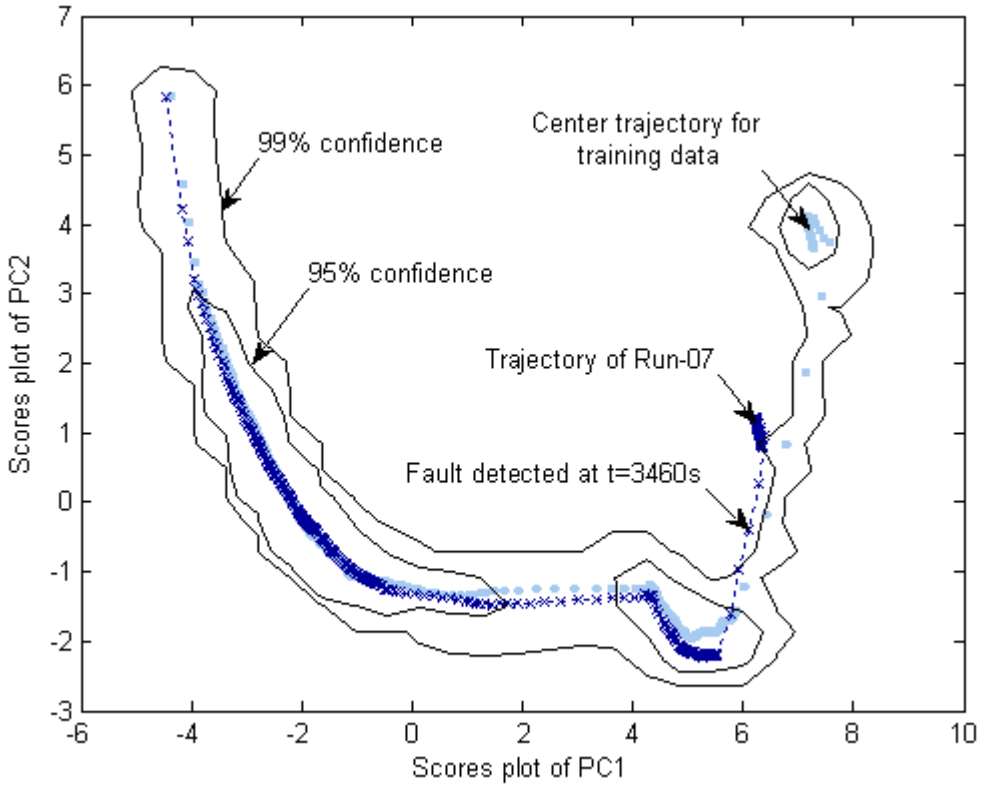
Run-07 corresponds to a low reflux ratio. The fault was introduced at Step 8 (t=3450s) when the plant operator sets the reflux ratio at 50% lower than the nominal value. Setting a low reflux ratio for the process has a negative impact on the resulting product quality. However, the effects are not immediately observable as product concentration is analyzed offline. The Hotelling's T^2 approach is unable to detect the fault throughout the run (Figure 6-11a), as the fault is contained within the space occupied by the monitoring boundary. In contrast, the process disturbance is detected successfully by the proposed kernel density-based approach at $t = 3460s$ (Figure 6-12a) when the online trajectory deviates from the 99% confidence limit. The T^2 and τ statistics are shown in Figure 6-11b and Figure 6-12b respectively. Fault identification at time of fault detection suggests two candidates as possible root causes, namely DST06, and DST07. This ambiguity is resolved at t=3490s after analyzing three additional samples.



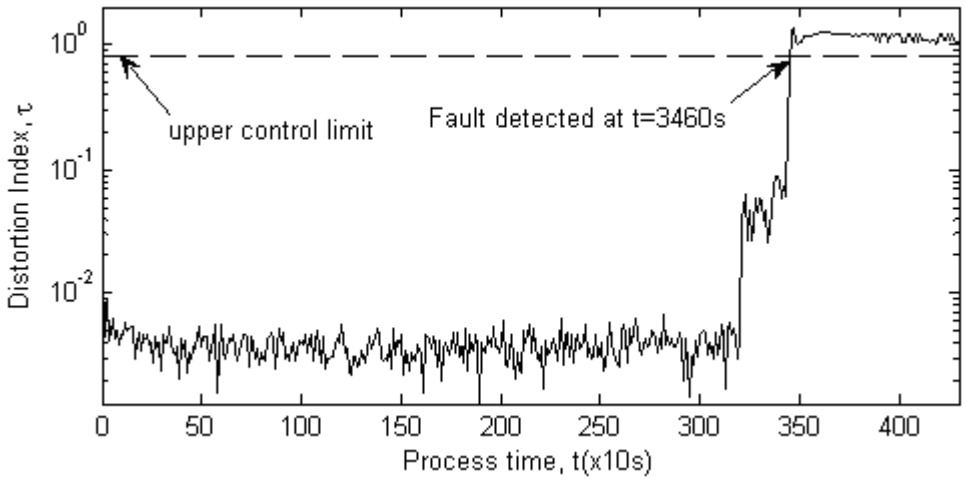
(a)

(b)

Figure 6-11: Monitoring on Run-07 during distillatioin startup using Hotelling T^2



(a)



(b)

Figure 6-12: Monitoring on Run-07 during distillation startup using Distortion Index

Scenario 2: Monitoring of a Complex Fault

Run-10 corresponds to a complex fault arising from low flow of condenser cooling water and feed pump malfunction. The first fault occurred right after the startup (t=10s), and the second fault occurred during Step 7 of SOP (*t*=4000s) when the pump failed, thus cutting off the feed to the column. Since the heat exchanger is not

equipped with a flow meter, the fault cannot be observed directly. The symptoms of the fault start around t=3000s when the outlet temperature of the condenser cooling water – T12 shows abnormality. In this case study, Hotelling's T^2 is unable to detect both failures. In contrast, the kernel density-based approaches (both bivariate and binomial combination) detect the fault at t=3040s. The fault signals at time of fault detection are used for fault identification. With bivariate KDE (using only PC1 and PC2), two fault candidates - DST04, and DST10 are identified as similar to the online signal. The DST10 is identified as the only remaining candidate after the analysis of 19th additional samples at t=3190s. The proposed KDE-based fault isolation approach improves the class discrimination by identifying DST10 promptly at the time of fault detection (t=3040s).

Similar studies were conducted for all other faults as shown in Table 6-3 and Table 6-4. Hotelling's T^2 was unable to detect three faults – DST05, DST07 and DST10. Most faults were successfully detected and diagnosed by the bivariate and the proposed KDE-based monitoring techniques. The KDE-based monitoring techniques failed to detect disturbance DST05, which corresponds to a sensor fault. The bivariate KDE performs poorly during Run-04. Its performance can be improved using the proposed distortion index where contributions from all PCs are considered. The average detection delay of Hotelling's T^2 is 356s. Detection delay based on bivariate-KDE and the proposed approach are 350s and 200s respectively (Table 6-3). The proposed KDE method is also less prone to Type-I errors compared to Hotelling's T^2. Hotelling's T^2 shows unusual number of false positives during early stage of the startup ($t \sim [10, 30]$s). In this case study, the bivariate-KDE approach fails to isolate fault DST08. All other faults are isolated with an average diagnosis delay of 217.5s based on the faults identifiable with bivariate KDE. On the other hand, the proposed

fault isolation strategy successfully diagnosed all detectable faults with an average diagnostic delay of 41.3s (Table 6-4). This case study hence further illustrates the good class separable feature of the proposed KDE-based pattern recognition method.

Table 6-3: Monitoring results for the distillation unit startup

Run #	Time fault introduced (x10s)	Hotelling T^2		Bivariate KDE		DIBC	
		Time of detection (x10s)	Detection Delay (x10s)	Time of detection (x10s)	Detection Delay (x10s)	Time of detection (x10s)	Detection Delay (x10s)
Run-01	1	6	5	2	1	2	1
Run-02	1	2	1	2	1	2	1
Run-03	359	370	11	370	11	370	11
Run-04	356	462	106	462	106	357	1
Run-05	425	-	-	-	-	-	-
Run-06	353	354	1	354	1	354	1
Run-07	345	-	-	346	1	346	1
Run-08	347	472	125	472	125	472	125
Run-09	1	1	0	1	0	1	0
Run-10	300	-	-	304	4	304	4
Avg Delay			**35.6**		**35.0**		**20.0**

Table 6-4: Fault diagnosis results for distillation unit startup

Run #	**Bivariate KDE**			**DIBC**		
	Fault candidates, F^{opt} (DST)	**Time fault isolated (x10s)**	**Isolation Delay (x10s)**	**Fault candidates, F^{opt} (DST)**	**Time fault isolated (x10s)**	**Isolation Delay (x10s)**
Run-01	F1	2	1	F1	2	1
Run-02	F2	2	1	F2	2	1
Run-03	F3,F6,F7	390	31	F2,F3,F5	380	21
Run-04	F4	463	107	F4	357	1
Run-05	-	-	-	-	-	-
Run-06	F6,F8,F10	359	6	F6	354	1
Run-07	F3,F6,F7	354	9	F6,F7	349	4
Run-08	F4,F8	-	-	F4,F8	482	135
Run-09	F9	1	0	F9	1	0
Run-10	F4,F10	319	19	F10	304	4
Avg Delay			**21.75**			**4.13**

6.7 Summary

Fault detection and diagnosis of transient operations are important to identify abnormal events and enable their timely recovery. Currently, most statistical methods for process monitoring, specifically the PCA approach, is based on Hotelling's T^2. However, Hotelling's T^2 is unable to generate reliable bounds for a model when the underlying operations are non-stationary, or non-Gaussian. The run-length variations common across different instances of transient operations restricts the application of time-wise unfolding methods with PCA unless explicit synchronization of trajectories is performed (Doan and Srinivasan, 2008). One alternative to the above problem is through the use of multiple-models. However, most multi-model approaches suffer from high rate of errors during model switching (Srinivasan *et al*, 2005). Also, the design of a robust model selection algorithm for online applications can be difficult. Given the complexities associated with multi-model approaches, single model-based approaches offer a more reliable means of monitoring transient operations.

Since reliable estimation of the underlying distribution is also difficult, non-parameteric approach such as Kernel Density Estimator (KDE) is hence a better alternative for FDI during transient mode of operations. Previous forms of KDE are constrained by their bi-variate analysis and manual interpretation of the bounds. To overcome this, a novel monitoring index called Distortion Index has been developed by combining the results from all binomial combinations of KDEs and weights them according to their level of significance. Since KDEs classifies the region of the training data more accurately, unknown regions in the PC subspace are better excluded. This reduces the Type-II errors. Though efficient in detecting abnormal samples in high dimensional space, binomial combination could easily lead to the problem of

"combinatorial explosion" when high number of PCs are used (J>20) and necessitate powerful computing resources for its online execution.

When a suitably annotated historical database is presented, both normal and abnormal, the proposed index can be used to locate the best matching fault by comparing the similarity of real-time measurements with the signals in the database. This can be used to identify known operating faults which occurrence is frequent. Extensive testing of the proposed method using two case studies clearly demonstrates the methods' ability to reduce both Type-I and Type-II errors when compared to Hotelling's T^2. Throughout this paper, the focus has been on the improvement of classifying the normal/abnormal samples in the scores space. Combining the SPE statistic, or using additional information from the residue space, might improve the classification further and worth further studies.

Nomenclature

Indices

i process time representation (row of a given matrix)

j principal components

n process variables

r points that constitute the KDE bounds

Parameters

J number of principal components retained with a PCA model

l time lag parameter used to construct DPCA model

N number of variables within a matrix

W total number of fault classes in the database

Φ scaling factor used to scale bandwidth matrix H

Variables

$\mathbf{1}$	an identity matrix
F_i^{opt}	best matching fault candidate for data x_i
F_w	fault dataset for w^{th} class of fault
H	bandwidth matrix used to construct KDE model
H^{opt}	optimal bandwidth matrix for bivariate KDE
H_{LSCV}	bandwidth matrix identified with LSCV method
$K(x)$	a kernel function used for density estimation of x
L	number of fault samples observed
$M_{jj'}^{KDE}$	KDE model for the j^{th} and j'^{th} variable in the PCA scores matrix $t_{jj'}$
M^{Fw}	KDE model constructed with F_w
S_{iw}	similarity between x_i and w^{th} KDE model M^{Fw}
S^{PCA}	PCA similarity measurement between two models
T^2	Hotelling's T^2 statistic
t_i	score vector for data x_i
$t^{jj'}$	the matrix constructed with j and j' columns from the original scores matrix t
x_i	online measurements observed from plant sensors
x_i^D	autoscaled online measurement augmented with time lag data
Y^D	autoscaled training data augmented with time lag data
Y	autoscaled training data used for training of KDE model
$\tilde{Y}$	raw data used for training of KDE model
λ_j	eigenvalues for j^{th} principal components
τ_i	distortion index
τ^{β}	upper control limit for τ
ψ_{iw}	1 if τ_{iw} observed between data x_i and M^{F_w} is in control

Chapter 7 Collaborative Agents for Managing Efficient Operations

7.1 Introduction

The strength of the different FDI methodologies described in the previous chapters can be combined to further enhance the performance of the diagnostic system. As described in Chapter 2, each FDI approach has its own strengths and shortcomings that are process dependent. A method that works well under one circumstance might not work well under another when different features of the process come to the fore. For instance, the multi-faceted nature of a process operation under different operating modes and transitions complicates the task of fault diagnosis, as different types of data analysis methods might be required under different conditions of operations. Since no single FDI method is able to address the numerous facets of transient processes, collaboration among heterogeneous methods is needed to bring forth the benefits of each method so that monitoring resolution and robustness of the FDI system can be brought to a higher ground. The rationale for such an approach is based on the precept that the strengths of different methods can be integrated to bear on the problem and the drawbacks of an individual method can be overcome through *collaboration*. Most monitoring and fault diagnosis algorithms are computationally complex while the answers are often needed in real-time. An efficient means of speeding up the execution time of the integrated FDI method is thus required. Towards this end, a multi-agent architecture is described in this chapter to realize the integration in practice.

7.2 Collaborative Agents for Managing Efficient Operations

7.2.1 Agent Environment

An agent-based framework called *Collaborative Agents for Managing Efficient Operations* (CAMEO) is described here to render effective management of process operations possible. The proposed framework can be abstracted hierarchically as environment, hosts, and agents (Figure 7-1). An *environment* in CAMEO can be thought as a neighborhood of entities that supports plant operation. All entities within the plant can be part of the environment, *i.e.*, software, hardware, controllers, humans, etc. Each agent environment might contain a number of *hosts* where each host determines the suitability of hosting a certain type of agents. The host contains relevant methods to identify the relevant agents within it, and it keeps track of any inter-host communication among agents. More importantly, the host contributes processing capability to the agent environment by sharing available processors. High performance computing units such as computer clusters or supercomputers are excellent candidates for hosts.

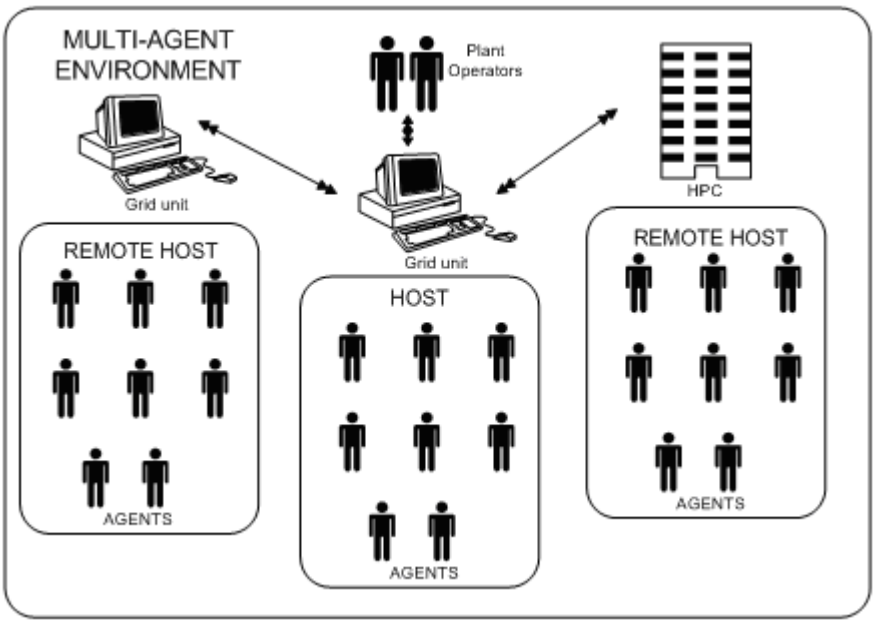


Figure 7-1: Hierarchical abstraction of the proposed agent-based framework

7.2.2 Agents Classification

The agents within CAMEO form the primary entities of the proposed framework. Each agent encapsulates knowledge in certain areas and contains routines/algorithms capable of solving a certain task. Each agent does not necessarily form a complete application but rather as a reusable, self-contained piece of routine that can rationale its own decision based on the condition arises during process operation. In CAMEO, six classes of agents have been incorporated, each of which addresses a different aspect of process operation (see Table 7-1). Each class of agent is also platform independent, they can be initiated and executed from different hosts.

Table 7-1: Classes of agents implemented in CAMEO

1.0 Data Management Agents
2.0 Operation and Control Agents
2.1 Regulatory Control Agents
2.2 Sequential Control Agents
2.3 Alarm management agents
3.0 Supervision Agents
3.1 State identification Agents
3.2 Monitoring Agents
3.3 Diagnostic Agents
3.4 Process Optimization Agents
4.0 Consolidator Agents
5.0 Fault Tolerance and Recovery Agents
6.0 Visualization and User Interface Agents

Data management agents provide the conduit to the process and its states for all other agents. Raw sensors data are read, de-noised, and reconciled by data management agents and the smoothed data is made available to other interested agents as messages.

The *operation and control agents* aid the operator in executing and controlling the different steps of a process transition. Some examples of these follow: *Regulatory control agents* interact with the DCS and perform actions such as reconfiguring the controller settings based on the current state of the process. The *sequential control*

agents coordinate the discrete steps required for executing sequential operations. The *alarm management agents* help reconfigure the alarm management system to current state and thus prevent alarm floods.

The goal of *process supervision* agents is to ensure safe operation. For effective process supervision, knowledge about the current process state is essential. Also, one way to prevent abnormal situations and make plant automation applications to function appropriately is to make them cognizant of the domain of their applicability. These applications can then enable or disable themselves in specific modes and reconfigure themselves with the correct settings when different states have to be handled differently. In order to automate such switching, context-sensitive information regarding the process state should be provided to the plant automation applications, which would then reconfigure themselves to the operating state. This is possible only if the different process states known *a priori* and their occurrence can be identified online. In CAMEO this role is performed by the *state identification agents.*

Other supervisory agents are *monitoring agents*, which detect abnormalities, and *diagnostic agents*, which are responsible for diagnosing the root cause. Monitoring and diagnostic agents can incorporate various FDI methods such as qualitative trend analysis, neural networks, rule-based system, etc. Successful fault isolation is achieved by collaboration between supervisory agents and consolidator agents. Results from monitoring agents are sent to consolidator agents to resolve possible conflicts generated from the multiple monitoring techniques that maybe applicable in any given state. Having solved the conflicts, the consolidator agents entrusts diagnostic agents to locate the possible root cause(s) of the situations. Subsequently, further appropriate corrective actions are planned by *fault tolerance and recovery agents*. This may require context specific tasks which are performed by sequential control agents.

When multiple methods are used concurrently, conflict resolution is needed to arbitrate among the different solutions proposed by the different FDI methods and provide one consolidated solution to the operator. In CAMEO, this role is performed by *consolidator agent*. The consolidator agent contains the logic to enforce consistency among different agents and are the bedrock for the collaboration mechanisms. Details on the algorithms used are described in Chapter 8 of this thesis. Other agents including supervision agents, operation and control agents, and visualization agents also interact with consolidator agent to enable consistency. Consolidator agents in turn use process state, historical performance, domain knowledge and other basis to fuse independent pieces of information contributed by other agents. The consolidated conclusion regarding the normality of the process operation and the root causes of any deviation provide the basis for planning corrective actions.

Having identified the root cause, the *fault tolerance and recovery agents* use process knowledge, available preplanned SOPs for specific situations, as well as historical records of corrective control actions to guide the process into a safe state to recover from an abnormal situation. The sequence of corrective measures generated by these agents can be implemented directly by the operation and control agents or communicated to the operations personnel through *user interface agents*.

A state specific context-sensitive graphical user interface is provided by the *visualization agents* in CAMEO. Visualization agents use powerful visualization tool in facilitating plant personnel to visualize the progression of process more easily. The visualization agents incorporate methods for projecting high dimensional data onto a much lower dimensional grid and thus provide a means for plant personnel to visualize even long duration process transitions effectively. *User interface agents* can also

automate email services to inform the relevant authorities regarding the events occurring in a plant, e.g: detection of fault, successful diagnosis of faults, state change operation, etc. The interactions among different classes of agents are illustrated in Figure 7-2.

Apart from the six classes of operational oriented agents, another class of agent: *mobile agents* are required to facilitate implementation. *Mobile agents* are responsible for speeding up computation time by reducing the workload of the local host. Mobile agents provide a means for inter-platform transportation among agents. Agents that require high computing resources can be transported to other platforms through mobile agents. The mobile agents use Message Passing Interface (MPI) for portability and platform independence. In general, the agents developed in this work are able to mimic some aspects of a human by being able to: (1) react to various circumstances due to changes in plant operating conditions, and (2) exhibit social ability such as cooperation, negotiation and migration.

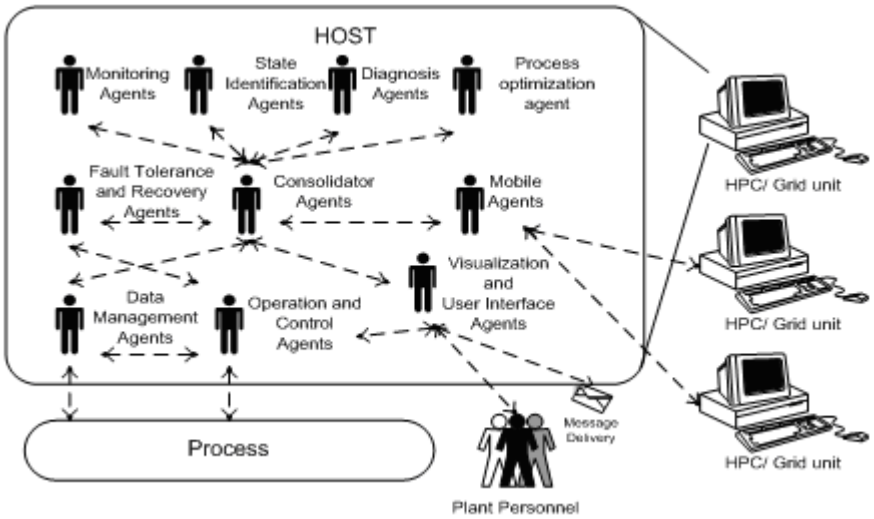


Figure 7-2: Interaction among heterogeneous types of agents

7.2.3 Agent Communication

As described in Section 2.7, FDI algorithms can be computationally expensive and a parallel implementation of the framework is essential to enable decision support

in real-time possible. To realize the speedup in practice, the mobile agents use Message Passing Interface (MPI) to exploit multiple processors (Kepner and Ahalt, 2004). MPI is a popular interface to send messages across a network of computers. It allows the coordination of programs running on either distributed memory computers or on shared memory systems. Each agent within CAMEO is platform independent, and can be executed from any distributed location through mobile agents. The agents on different processors communicate with each other by exchanging messages based on a purpose-designed ontology (see Figure 7-3). Since agents of different classes require different data types in their methods, each agent needs to produce results in a specific format to enforce consistency. Knowledge-base approach based on string recognition is developed for each class of agents for ontology synchronization. Each class of agents has a vocabulary (list of strings) containing the queries the agent is capable of responding to, e.g.: for monitoring agent, the agent is capable of recognizing "*monitoring*", "*request_fault_diagnosis*", "*transport_agent*", "*reload_agent*", "*get_data*", "*send_results*", "*activate*", "*deactivate*" etc, each representing various tasks the agent is capable of performing. An agent can communicate with another by sending it appropriate queries within the latter's vocabulary.

An example of an inter-agent communication is shown in Figure 7-4. Suppose there are two agents, Agent-A and Agent-B, located at different hosts. Let the agents be denoted as A^a and A^b respectively, where A^a intends to request A^b to perform monitoring on a certain dataset, x_i. A common directory is used by A^a and A^b for communication. A^a first writes the message $\Psi^{A,buffer}$, which contains essential data (information) for a certain task to be executed by A^b to the common directory before creating an empty message, $\Psi^{A,lock}$ after the buffering of $\Psi^{A,buffer}$ is completed. The

agent A^b continually checks the common directory for existence of $\Psi^{A,lock}$ if its status is active. Once $\Psi^{A,lock}$ is detected, A^b loads the message $\Psi^{A,buffer}$ and executes the requested tasks based on the data in $\Psi^{A,buffer}$ received. Both messages can be deleted by agent A^b after the completion of data transfer. With this message passing architecture, the agents in CAMEO can function freely across different hosts in a predefined agent-environment. The distributed multi-agent framework thus enables speedup of computational demanding routines by exploiting multiple processors. Two performance measurement indicators are used in this work to measure the impact of speedup, namely, *overall speed enhancement index*, S_p, and the *overall system efficiency index*, E_p. Both S_p and E_p are given as:

$$S_p = \frac{\text{Average time required for single machine}}{\text{Average time required for p machines}}, \qquad \text{(Eq 7-1)}$$

$$E_p = \frac{\text{Speed up with } p \text{ processors}}{p}. \qquad \text{(Eq 7-2)}$$

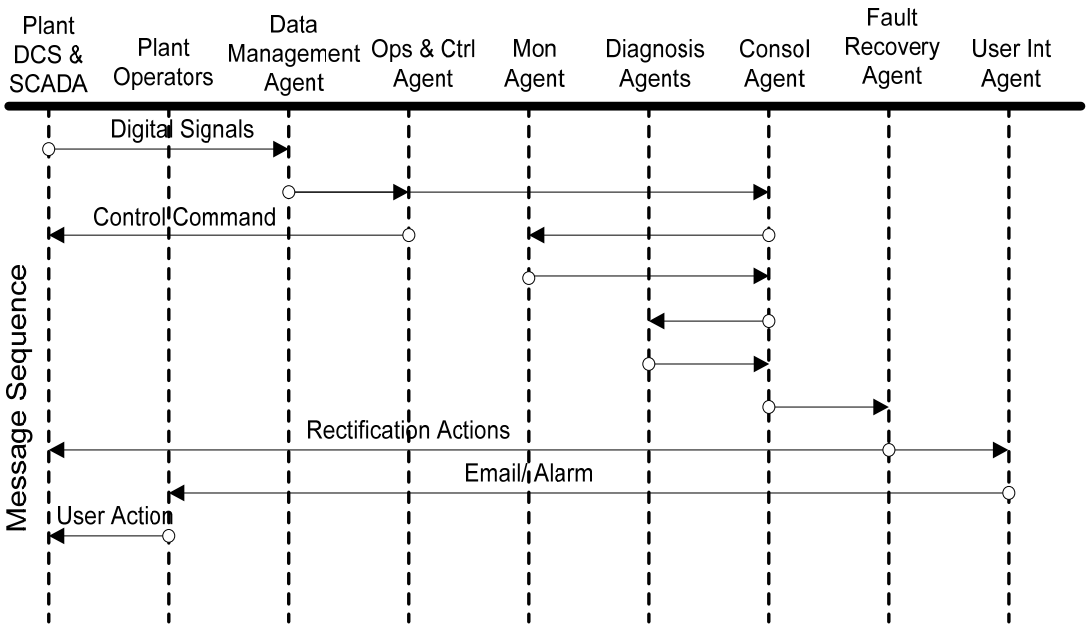
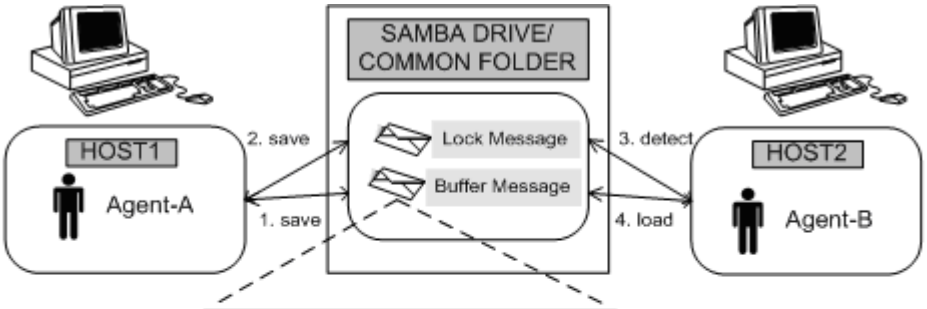


Figure 7-3: A typical flow of messages during online application



A message	
Attributes	***Value***
Recipient	"Agent-D"
Recipient Class	"Monitoring_agent"
Sender	"Agent-A"
Sender Class	"Data_management_agent"
Type	"Data"
Method	"Monitoring"
Status	"Ready"
Sending Time	"11.30am March"
Data	Matrixes/Vectors/Structures/Cells

Figure 7-4: Inter-hosts message passing among heterogeneous type of agents

7.2.4 Implementation of Multi-agent Architecture

The necessary steps for FDI based on the multi-agent formalism is described in this section. The system is implemented in Matlab and uses *Secure Shell command* (SSH Communication Security, 2006) to launch additional Matlab processes in other modes. The program can be executed on Windows and for Unix platforms. The necessary steps for implementing the multi-agent architecture are as follows:

Step 1: *Offline FDI models creation*: various FDI models for process monitoring, M^r, $\forall r = FDI$ methods implemented, are first created based on available training data, F_j, $j \in [1, J]$, with their parameters and thresholds properly tuned.

Step 2: *Agents initialization*: The constructed FDI models M^r, $r = [1, R]$, from step 1 can be used to initialize the agent environment. The initialization of agent-environment is usually done through the master node by spreading the corresponding monitoring and diagnostic agents across the subnodes. A common directory Ω^{dir} for communication needs to be established for message passing among the distributed agents (could be a Windows directory or Samba drive on Unix machines).

Step 3: *Data acquisition and preprocessing*: The online measurements, $\tilde{y}_i$, are obtained from plant sensors/simulators by data management agents. Appropriate filters are implemented by data management agents to filter out high frequency noise, and the resulting smoothed data, y_i, is normalized to appropriate format x_i before being distributed to monitoring agents, A_r^m, in the form of messages, Ψ^{x_i}.

Step 4: *Data analysis*: Each monitoring agent A_r^m collects the message Ψ^{x_i} from Ω^{dir} and performs analysis on x_i based on the queries received. The completed monitoring results from each agent, $e^r(x_i)$, are sent back to the consolidator agent A^c for decision fusion.

Step 5: *Fusion of monitoring results*: Consolidator agent A^c collects $e^r(x_i)$ from all A_r^m as message $\Psi^{e^r(x_i)}$. Monitoring results are fused and the status of the plant is decided. The decision from fusion of messages $\Psi^{e^r(x_i)}$ can be in either of the following states: normal i.e., $S(e^1(x_i) \cup ... \cup e^R(x_i)) = 0$, or abnormal, when $S(e^1(x_i) \cup ... \cup e^R(x_i)) = 1$. The A_r^m repeats Step 4 if the process is normal, otherwise A^c triggers diagnostic agents A_r^d for root cause identification.

Step 6: *Fault classification and identification*: All diagnostic agents A_r^d are initiated by A^c through message Ψ^{A^c}. Each A_r^d performs a classification on x_i based on its diagnostic methods, usually some form of similarity search with all available fault data, F_j, $j = [1, J]$, will be conducted. The completed diagnostic results $e^r(x_i)$ from each A_r^d are then sent to A^c for decision fusion.

Step 7: *Fusion of diagnostic results*: A^c collects Ψ^{A^d} from each A^d with their corresponding $e(x_i)$ fused. The fused diagnostic decisions (as a set of fault candidates)

are presented to users by the visualization and user interface agents.

Step 8: *Visualization and fault rectification*: The plant personnel interact with user interface agents to validate the faults identified from previous step. Once the real cause of abnormality has been identified, the actual recovery strategy is implemented through fault tolerant and recovery agents.

CAMEO has been designed to run optimally by spreading the processing load among six processors. The proposed method has also been tested on a Linux cluster with varying number of processors (1 to 8). The Linux cluster is made up of 16 Intel nodes containing 2 Pentium Xeon 3.06GHz processors each and 2GB memory. The nodes are connected together via high-speed, low-latency Myrinet interconnects. The theoretical performance of the 32-CPU cluster is estimated at 195840 MFlops (SVU webpage, 2006). Figure 7-5 shows the proposed system architecture when executed on Linux Cluster. The proposed multi-agent framework is tested next with a fed-batch penicillin cultivation process and a transition operation comprising startup of a pilot-scale distillation unit.

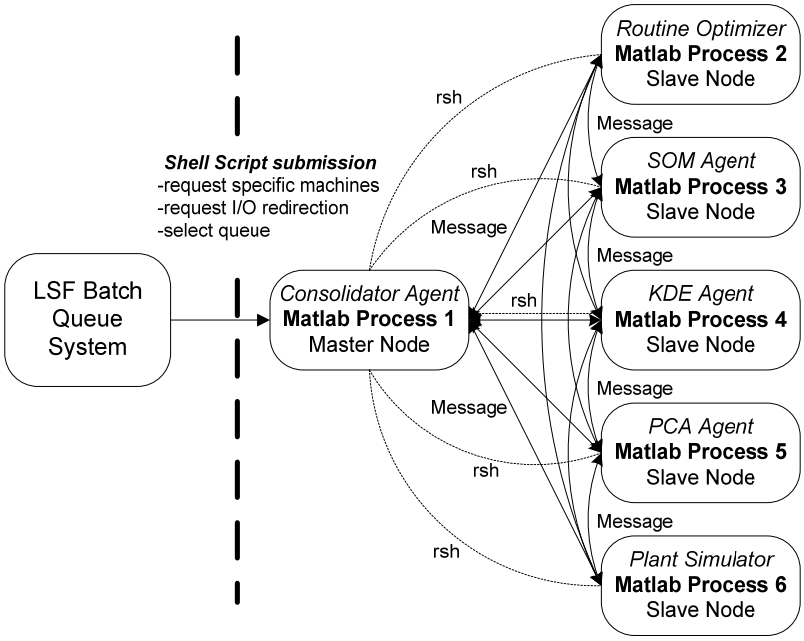


Figure 7-5: Overview of system architecture on Linux Cluster

7.3 Case Study 1: Fault-diagnosis for a Fed-batch Penicillin Cultivation Operation

The proposed multi-agent framework is illustrated with a fed-batch penicillin cultivation process in this section. Three monitoring agents have been implemented based on Dynamic Principal Components Analysis A^m_{DPCA}, kernel density estimate A^m_{KDE} and self-organizing map A^m_{SOM}. Out of the three monitoring techniques, two of them (SOM and KDE) have diagnostic capabilities and are subsequently implemented as diagnostic agents, represented here as A^d_{SOM} and A^d_{KDE}. The self-organizing map constructed from A^m_{SOM} for a normal run is shown in Figure 7-6. The normal operating trajectory of SOM follows a predefined sequence during normal operations. The trajectory starts evolving from cluster 28 to cluster 145 when the system switches to fed-batch mode, and finally stops at cluster 56 when the fed-batch operations are completed successfully. The state-signature obtained in real-time, *i.e.*, by tracking the cluster evolution, can be used to detect and identify known process disturbances. In contrast to A^m_{SOM}, A^m_{KDE} constructs non-parametric bounds on the principal components subspace (Figure 7-7) based on the normal fed-batch trajectory, while A^m_{DPCA} uses T^2 and *SPE* for monitoring purposes. The agent environment created thus consists of the process simulator (PENSIM v2.0) and the Linux computing cluster, which is the main host for the multi-agent system. The agents are distributed across the computational environment during initialization phase by placing them in different nodes. Eight process disturbances have been tested for this fed-batch operations with the summary of each fault given in Table 5-4. Three case studies namely, SIM02, SIM05, and SIM07 are illustrated next.

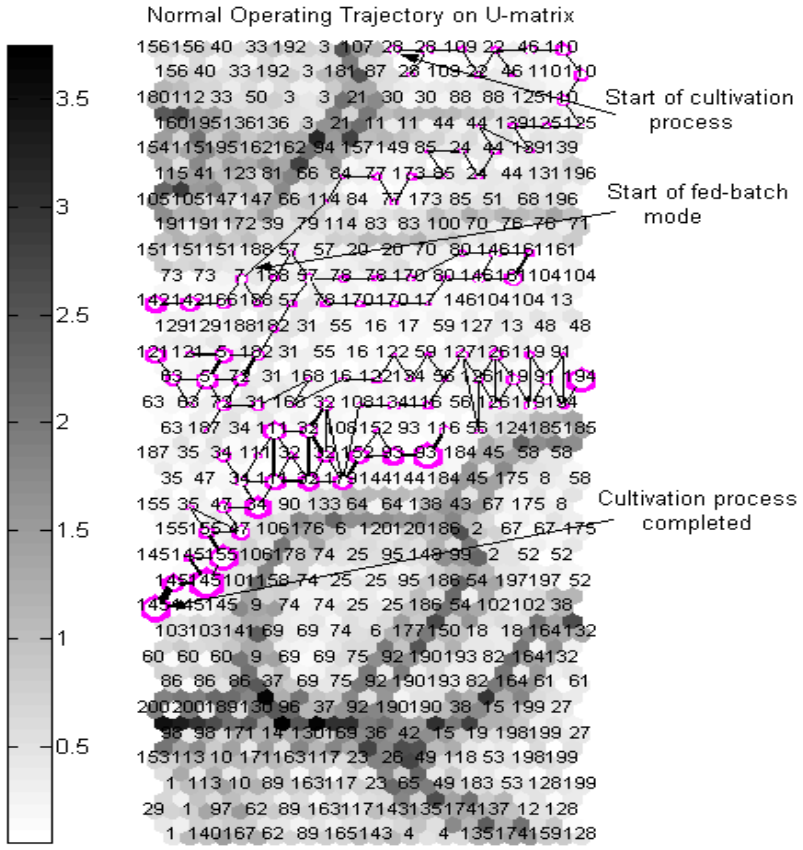


Figure 7-6: Normal operating trajectory of penicillin cultivation process on SOM-monitoring-agent

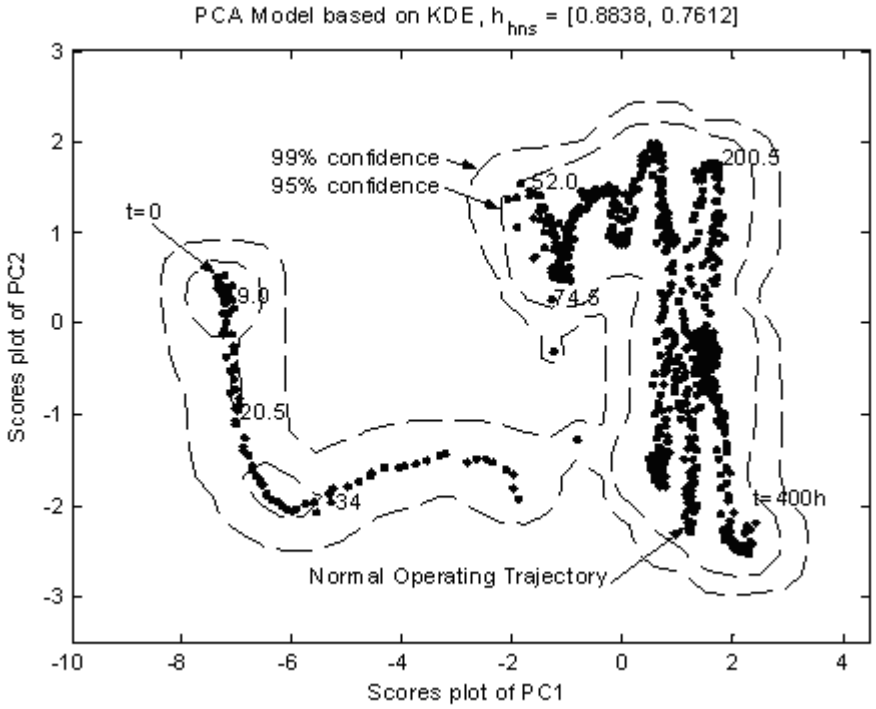


Figure 7-7: Normal operating trajectory of penicillin cultivation process based on KDE-monitoring-agent

Scenario 1: SIM02- temperature controller failure

SIM02 corresponds to a failure in temperature controller and the fault was introduced at *t*=0.5h of production time. The failure of the PID controller causes the temperature in the fermentor to increase from 298K to 327K. CAMEO was used for fault diagnosis and the corresponding timeline of events shown in Figure 7-8.

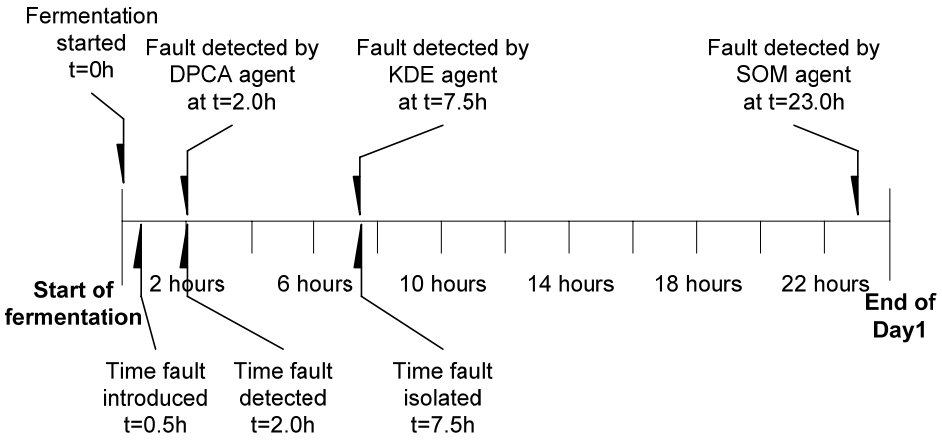


Figure 7-8: Timeline of events during run SIM02

The fault was first detected by A^{m}_{DPCA} at *t*=2.0*h* (Figure 7-9), with the consolidator agent A^{c} informed. Since both A^{m}_{SOM} and A^{m}_{KDE} could not detect the controller failure, only the variable's residuals are generated to facilitate fault identification by human operators (Figure 7-10a). The variables residuals are generated by A^{d}_{SOM} based on time synchronization between a reference run selected a priori, X^{R} with the online data collected, X^{o}. The residuals are shown as percentage deviation from their *optimal* operating conditions, m^{opt} on the SOM space. From Figure 7-10a, it can be seen that water flow (variable 11) has not been established, rendering heat removal from the fermentor unsuccessful, which in turn affects the growth of the biomass. Since A^{m}_{DPCA} is not equipped with fault identification features, the fault is unidentified at this stage.

At t=2.5h, A^{d}_{KDE} successfully gathers enough confidence to distinguish SIM02 from other fault candidates after additional data points have been obtained. A positive match from the fault database, SIM02, suggests that the temperature controller is the root cause of the abnormal event. A^{m}_{SOM} shows the worst performance in this case study by detecting the fault only at t=23.0h (Table 7-2). Late detection of this abnormal event might lead to a lost opportunity for early correction, and resulting poor quality batches that are unable to meet production specifications. If the fault is left unattended, more deviations can be seen (Figure 7-10b at t=200.0h) due to fault propagation. The low heat and carbon dioxide generated in comparison to the normal run clearly indicate that the yield is low in this run compared to normal batches. The operation is also unable to switch to fed-batch mode in this run as the condition for feeding is not met with a faulty temperature controller.

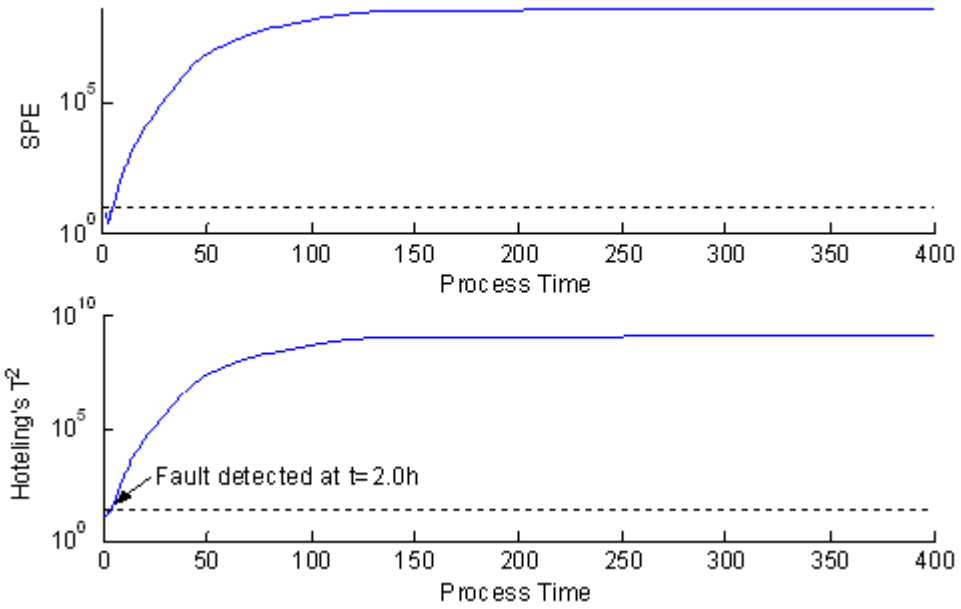
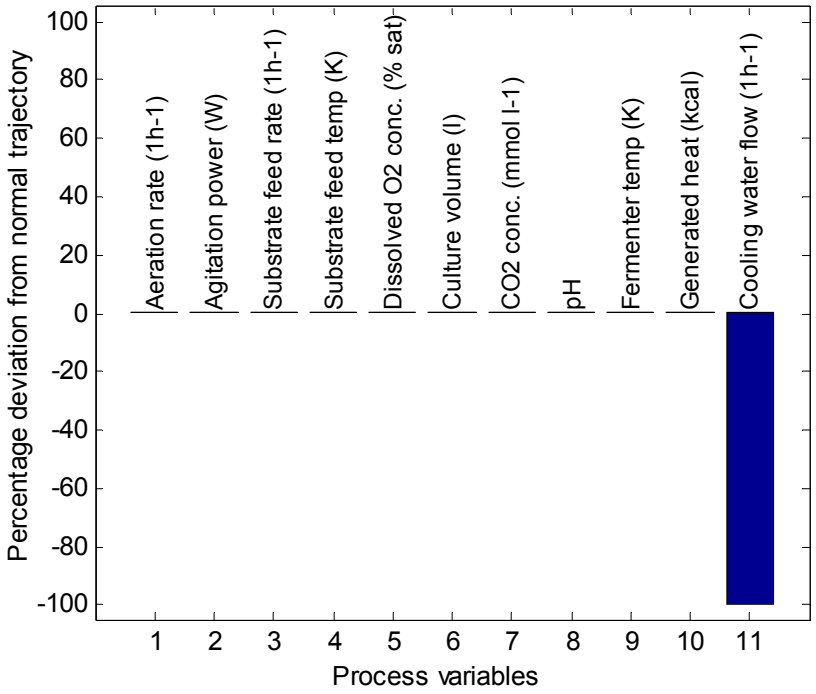


Figure 7-9: Monitoring results of SIM02 based on DPCA-monitoring-agent



(a) Percentage deviation from normal operating trajectory at time of fault detection t=5.0h (SOM-diagnostic-agent)

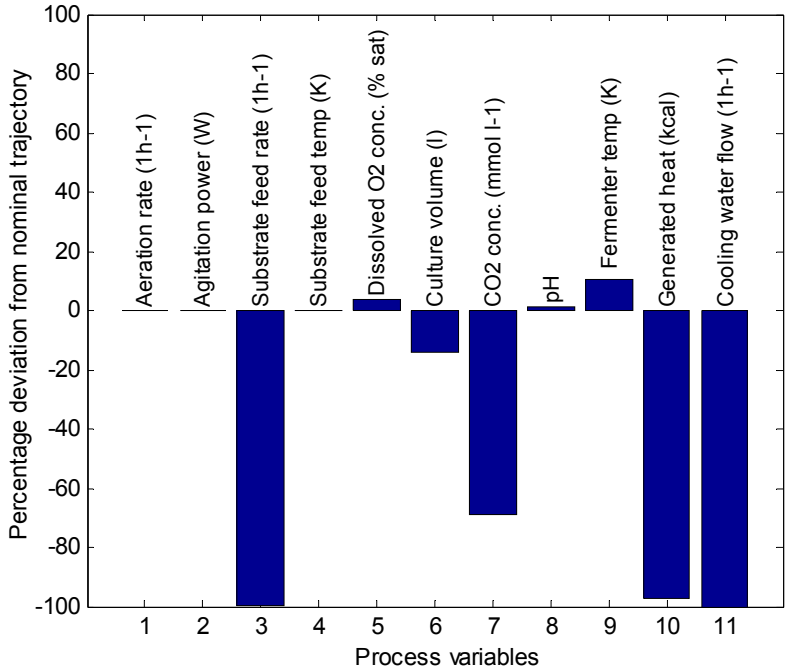


Figure 7-10: (b) Percentage deviation from normal operating trajectory at t=150h (SOM-diagnostic-agent)

Scenario 2: SIM05- Low substrate feed

Controlling the substrate feed to biomass during fed-batch phase has drawn a lot of attention lately in order to optimize the growth of biomass. The effect of human error on this important variable is studied by introducing SIM05, which corresponds to a 15% step decrease in substrate feed (introduced at t=50.0h). The decrement in feed rate reduces the food supply to the biomass and subsequently affects the amount of biomass cultivated. Consequently, the yield is much lesser compared to batches operated under optimal operating condition. The timeline of events during monitoring with CAMEO is shown in Figure 7-11. For this scenario, A_{SOM}^{m} first detected the disturbance at t=57.5h (Figure 7-12) when the online trajectory deviates from the nominal state-sequence. The fault was also isolated very quickly by A_{SOM}^{d} . The variable's residuals generated from A_{SOM}^{d} (Figure 7-13) show that the substrate feed rate is 15% lower than the optimal value and is affecting the biomass growth. The identified disturbance SIM02 and the variable's residuals are hence sent to plant operators for further actions.

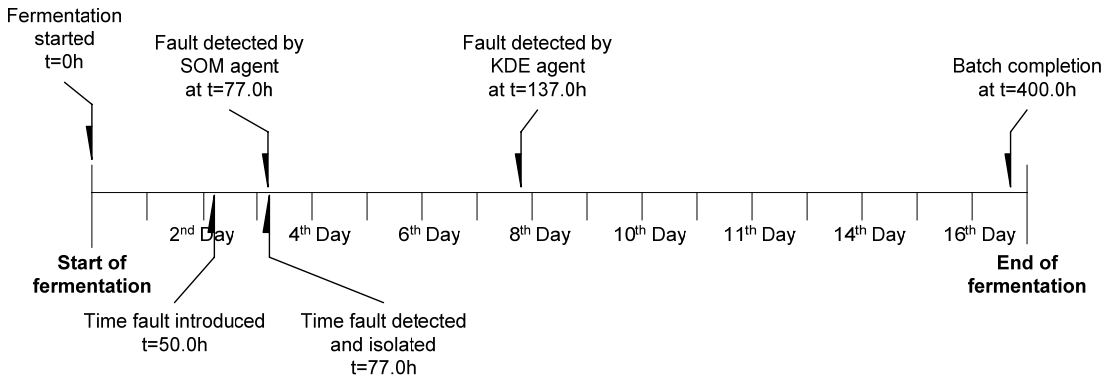


Figure 7-11: Timeline of events during run SIM05

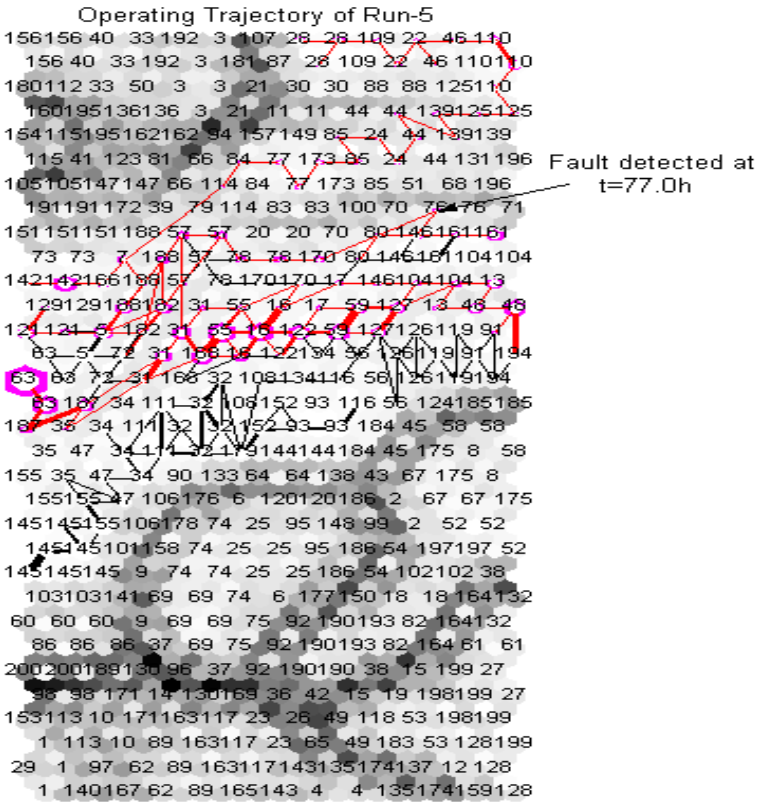


Figure 7-12: Monitoring results of SIM05 based on SOM-monitoring-agent

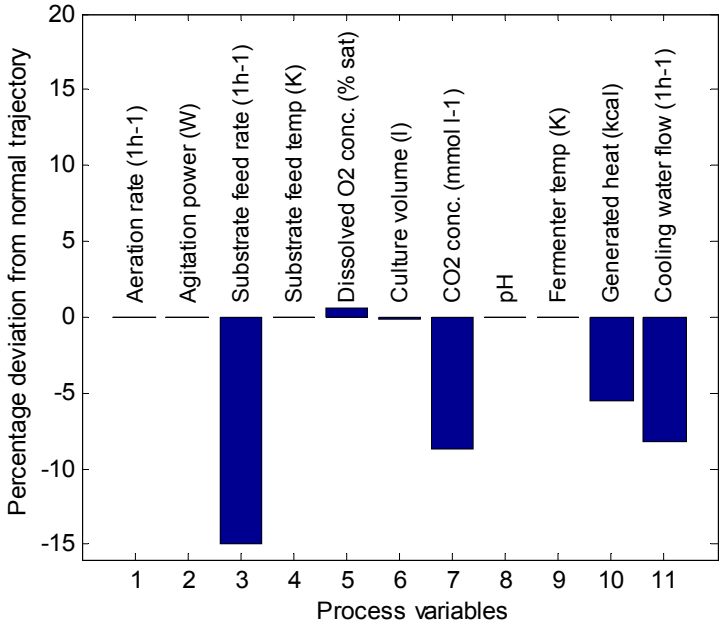


Figure 7-13: Percentage deviation from normal operating trajectory at time of fault detection (SOM-diagnostic-agent)

Scenario 3: SIM07- Ramp increase in agitation power

SIM06 corresponds to a ramp increase in agitation power. The fault was introduced at t=70h and was not resolved throughout the production period. The continuous increment in agitation power causes the stirrer to impose a high level of shear force on the biomass, which affects the growth of the biomass negatively under extreme conditions. For this scenario, the fault was first detected by A^m_{KDE} at t=133.0h with a detection delay of 63.0h (timeline of events shown in Figure 7-14). The fault was subsequently isolated by A^d_{KDE} at t=135.0h. Both A^m_{SOM} and A^m_{DPCA} showed poor performance in this scenario with detection time at t=194.0h and t=233.0h respectively.

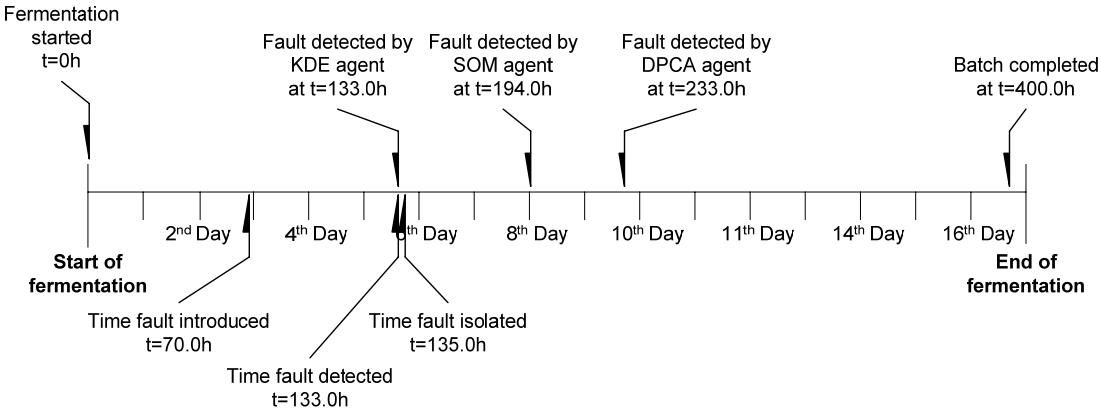


Figure 7-14: Timeline of events during run SIM07

Similar analysis was carried out for the rest of the faults and the FDI summary obtained is shown in Table 7-2. The proposed multi-agent approach detects all faults successfully with an average detection delay of 25.75h and diagnostic delay of 26.7h. The average detection delay of each independent agent, A^m_{DPCA}, A^m_{SOM}, and A^m_{KDE} is 50.9h, 50.1h, and 40.4h respectively. The proposed multi-agent approach thus offers a minimum improvement of 36.3% in terms of speed of fault detection. The three scenarios presented show that no single FDI method can guarantee optimality in

reducing fault detection and identification delay. Some FDI methods are able to perform better than their peers in certain scenarios but poorly in other cases.

The S_p observed using multiple processors are shown in Figure 7-15. The system benefited with a maximum speedup of 3.3 times when $p \geq 4$. The S_p is limited by the response time from A_{SOM}^m and A_{SOM}^d, in which the sequential nature of calculations prevent further speedup. As can be seen from Figure 7-15, the S_p becomes constant for $p \geq 4$, rendering further addition of processors not useful.

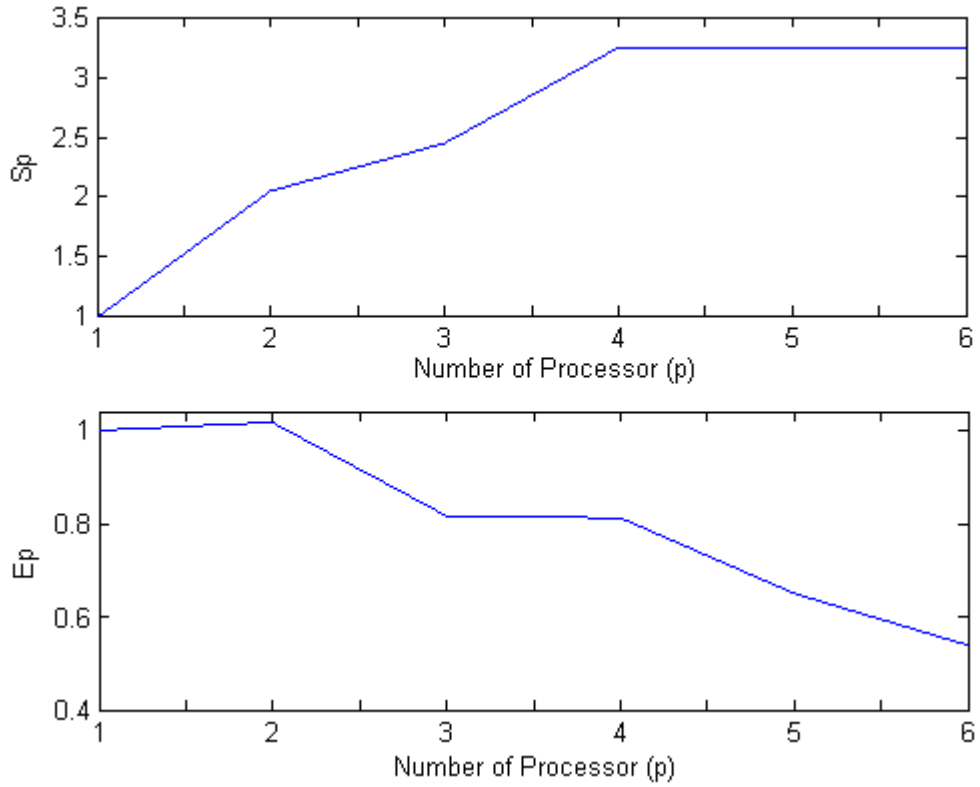


Figure 7-15: S_p and E_p observed during online implementation

Table 7-2: Fault-diagnosis results for penicillin cultivation case study

Case	Time fault introduced (hr)	Time fault detected (hr)			Possible candidates (KDE)	Time fault isolated (KDE)	Possible candi-dates (SOM)	Time fault isolated (SOM)	Time fault isolated (combine)
		SPE/T^2	SOM	KDE					
SIM01	0.5	1.5	19	1.5	F1	1.5	F1	19	1.5
SIM02	0.5	2	23	7.5	F2	7.5	F2	23	7.5
SIM03	60	60.5	60.5	60.5	F3	60.5	F3	60.5	60.5
SIM04	30	30.5	30.5	30.5	F4	30.5	F4	30.5	30.5
SIM05	50	-	77	189	F5,F8	205	F5	77	77
SIM06	70	82	154	77	F6,F8	81	F6	154	81
SIM07	40	233	194	133	F6,F7,F8	135	F7	194	135
SIM08	30	177.5	124	105.5	F6,F8	156	F8	124	124

7.4 Case Study 2: Fault-diagnosis during Distillation-unit Startup

In this section, the proposed multi-agent architecture is tested with the distillation unit startup case study. Fault diagnosis using CAMEO to disturbance DST08 will be illustrated next.

DST08 corresponds to a disturbance of high reflux ratio. The disturbance was caused by human error at *t*=3460s when the reflux ratio was set to twice its nominal value. This causes a reduction in product yield and increases the load of the reboiler. The timeline of events for this case study is shown in Figure 7-16. In this scenario, the DPCA monitoring agent, A^m_{DPCA}, detects the abnormality first at *t*=3470s (57.8min) with the diagnostic agents initiated for root cause identification. The fault is successfully isolated at *t*=4700*s* when the SOM diagnostic agent, A^d_{SOM}, retrieves a fault candidate with high confidence from the fault database. The time taken for fault detection using just one technique is longer, *i.e.*, *t*=4220*s* (SOM) and *t*=4720*s* (KDE).

The summary of fault-diagnosis for other cases studied is shown in Table 7-3. Both A^m_{KDE} and A^m_{DPCA} fail to detect disturbance DST05. All faults are successfully detected and isolated based on CAMEO with an average detection and diagnostic delay of 31.1 samples and 44.3 samples respectively. Average detection delay based on single FDI approach is 35.2 samples (SPE/ T^2), 35.9 samples (SOM), and 49.0 samples (KDE). Minimum improvement of 11.64% is achieved in detection time by combining these three FDI methods compared to independent implementation. The S_p and E_p observed using multiple processors are shown in Figure 7-17. The use of a routine optimizer which further parallelizes A^d_{SOM} has been able to achieve a speedup of ~4.4x based on the hardware configurations as specified in Section 7.2.3. Average

processing time has been reduced from ~12s per sample to ~2.7s per sample during abnormal events (when high level of CPU flops are required).

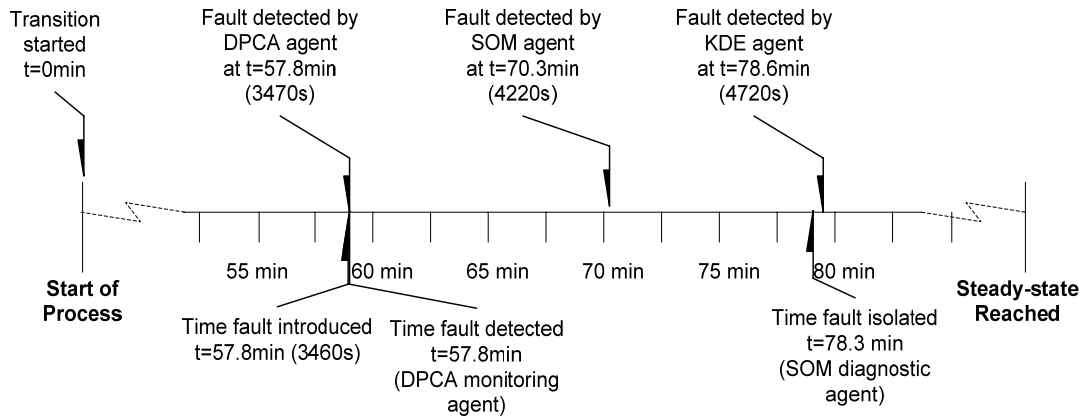


Figure 7-16: Timeline of events during run DST08

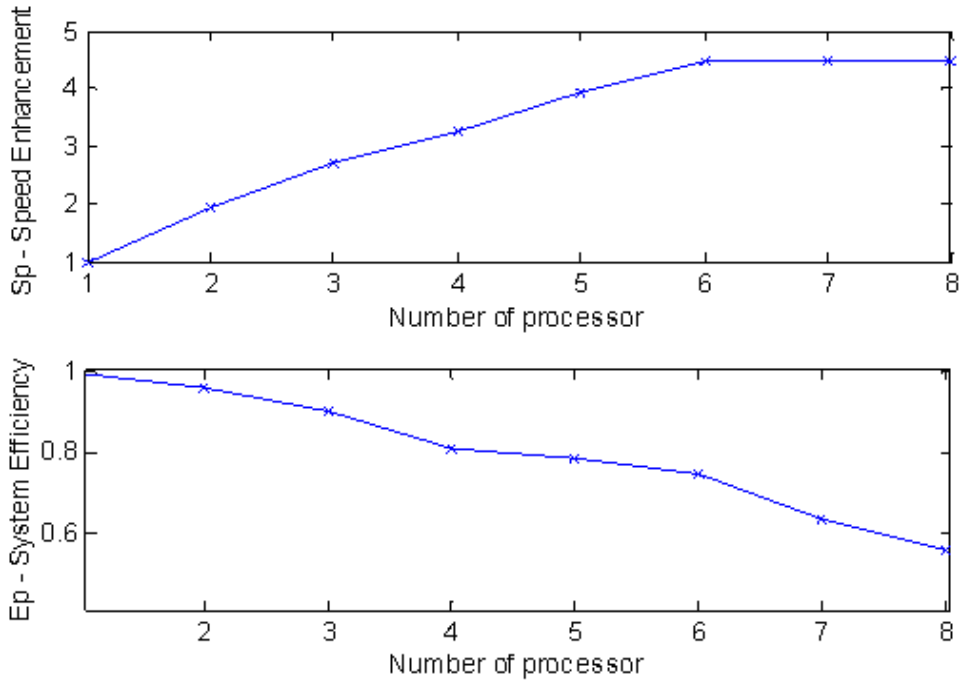


Figure 7-17 : Speed enhancement and system efficiency measured on the *Linux cluster* during fault diagnosis of distillation-unit startup

Table 7-3: Summary of fault-diagnosis for startup of distillation-unit case study

Scenario #	***Time fault introduced (x10s)***	***Time fault detected (x10s)***				***Fault Candidates (KDE)***	***Time Fault Isolated (KDE)***	***Fault Candidates (SOM)***	***Time Fault Isolated (SOM)***	***Combined Isolation Time***
		SPE/T2	***SOM***	***KDE***	***Combined detection***					
DST01	1	1	10	1	1	F1	1	F1	10	1
DST02	1	1	1	1	1	F2	1	F2	1	1
DST03	359	372	365	369	365	F3,F5,F6,F8,F9	399	F3,F4,F8	371	371
DST04	356	357	360	357	357	F4,F9	359	F3,F4,F8	389	359
DST05	425	-	428	-	428	-	-	F5	428	428
DST06	353	353	354	354	353	F6,F8,F9	357	F6	354	354
DST07	345	349	347	346	346	F3,F6,F7,F10	350	F7	347	347
DST08	347	347	422	472	347	F4,F8	475	F3,F4,F8	470	470
DST09	1	1	1	1	1	F9	1	F9	1	1
DST10	1	300	300	304	300	F4,F5,F9,F10	347	F10	300	300

7.5 Summary

Though the focus of this study is oriented towards the domain of fault detection and isolation, the CAMEO multi-agent based approach is intended to be a general framework to efficiently manage process operations. CAMEO offers the flexibility to bring together different techniques in a concerted way with the shortcomings of each entity overcome through collaboration. Different algorithms can be combined easily with an agent wrapper and the results from each agent fused through a consolidator agent based on messages as foundation for communication. The two cases studied have clearly illustrated the three key benefits of the proposed multi-agent approach:

1. improvement in the speed of fault detection and isolation,
2. flexibility in integrating the strengths from numerous methods to provide a multi-faceted analysis, and
3. reducing processing time and hence improving screen refresh rate of DCS.

Three fault detection and diagnosis methods have been integrated in this chapter, namely, FDI with T^2 and *SPE* statistics, self-organizing map, and kernel density estimator. In all cases studied, CAMEO successfully detected and diagnosed all faults with minimum detection and diagnostic delay compared to independent application of each FDI method. Early warning and fault recovery can thus be initiated quickly to prevent fault propagation.

The introduction of parallel programming technique (through Message Passing Interface) reduces computational time of the above algorithm and renders integration of time-consuming FDI algorithms possible. CAMEO allows allocation of computing resources from numerous hosts, and is able to reduce memory requirements on the main host by distributing the agents across various nodes (hosts) in the computing

cluster. The proposed multi-agent approach shows major dissimilarity with previous attempts in integrating FDI methods that were often based on hybrid approaches (Mylaraswamy and Venkatasubramanian, 1997; Özyurt and Kandel, 1996; Vedam and Venkatasubramanian, 1999). Hybrid systems for FDI can be considered as a deterministic approach where different FDI methods are often integrated layer-by-layer, *i.e.*, results from trend analysis integrated with secondary signed-digraphs, etc. Any failure of one of the entity would cause an abrupt failure to the system as subsequent layer of FDI could not function normally without the inputs from previous layer of FDI unit. Also, reorganizing the set of FDI entities and parallelizing the codes for distributed computing are major challenges for such highly integrated systems. The restricted scalability of hybrid systems prevents them from integrating a high number of fault classifiers. In contrast, the proposed multi-agent architecture offers the opportunity to solve a problem through distributed entities of collaborative agents. Since the underlying agents use messages for data exchange, the agents need not be situated within the same host to access critical information (data), hence enabling parallelism among the FDI methods possible through exploitation of multiple processors.

Nomenclature

Indices

i process time

j fault ID

p number of processors

r number of FDI agents

Parameters

I total number of samples (rows) in X

J total number of faults in the training database

Symbols

A^m_{DPCA} DPCA-based monitoring agent

A^d_{KDE} KDE-based diagnostic agent

A^m_{KDE} KDE-based monitoring agent

A^d_{SOM} SOM-based diagnostic agent

A^m_{SOM} SOM-based monitoring agent

A^c results consolidator agent

$e^r(x_i)$ monitoring results for x_i from agent r

E_p overall system efficiency index

F_j training data for fault j

m^{opt} optimal operating conditions identified from SOM

M^r FDI model constructed for process monitoring and fault identification

S_p overall speed enhancement index

x_i process data at i^{th} time interval

X^o online data collected from plant DCS

X^R reference data used to train a FDI model

y_i smoothed data obtained after noise removal

$\tilde{y}_i$ raw data collected from plant DCS

$\Psi^{A,buffer}$ Buffer message from Agent A containing the essential data for a certain task from

$\Psi^{A,buffer}$ Lock file from Agent A to inform the writing and deleting of buffer file

Ω^{dir} common directory used for agent communication

Chapter 8 Decision Fusion Strategies for Integration of Heterogeneous Diagnostic Fault Classifiers

8.1 Introduction

Previous literature on monitoring and fault diagnosis of chemical processes often depends on a single FDI method. Some popular FDI approaches include statistical analysis, neural-networks, signal processing methods, etc. It has been well illustrated in Chapter 2 that each FDI method has its associated strengths and shortcomings. There is hence a strong motivation for developing collaborative approach for FDI to bring together the strengths from different classes of FDI methods. In Chapter 7 of this thesis, a multi-agent framework was described to combine heterogeneous types of these FDI methods. However, when multiple FDI methods are used in parallel, a conflict resolution strategy is needed to arbitrate among the contradictory decisions generated by the various FDI methods so that one consolidated solution can be presented to the plant personnel. In this chapter, three decision fusion strategies for combining predictions from heterogeneous diagnostic classifiers namely, voting strategy, Bayesian combination strategy, and Demster-Shafer combination strategy, are studied and thoroughly analyzed. It was found that combining diagnostic classifiers of different types through any of decision fusion strategies improve diagnosis performance compared to approaches based on any single method.

8.2 Decision Fusion Methodologies

Consider a classifier κ being used for classification of input sample *x*. Suppose there are *J* classes and each class is represented by C_j, $j \in \Lambda = [1, J]$. The task of κ is then to assign each of the input sample, x, to one of the *J+1* classes, *i.e.*, $e(x) = C_j$, $j \in \Lambda = [1, J+1]$ with the $(J+1)^{th}$ class denotes that κ rejects x, implicating that the sample is not assigned into any known classes. In general, any classifier is able to provide output in one of the following forms (Xu, *et al.*, 1992):

1. *Abstract form*: only a unique class *j* or some subsets of the available classes are produced in its predictions $e(x) = C_j$, $j \in \Lambda = [1, J+1]$.
2. *Rank form*: all available classes C_j, $\forall j \in \Lambda = [1, J+1]$ are ranked and sorted, *i.e.*, $e(x) = [C_j^1, C_{j'}^2, ...C_{j''}^{J+1}]$, $\forall j, j', j'' \in \Lambda = [1, J+1]$.
3. *Measurement form*: a measurement, s, is given to each class C_j, $\forall j \in \Lambda = [1, J]$ to quantify the degree of similarity between the sample x and C_j, $j \in \Lambda = [1, J]$, *i.e.*, the output from a classifier κ, $e_\kappa(x)$, will take the form of a vector, $e_\kappa(x) = [s_1, ..., s_j, ..., s_J]$, $j \in \Lambda = [1, J]$, with s being continuous, $s_j \in [0, 1]$.

Among the three forms of predictions, predictions generated in measurement form contain the highest amount of information while predictions generated in the abstract form contain the least. If there are *R* classifiers denoted by κ_r, $r = [1, R]$, a total of *R* predictions will be generated from the classifiers, denoted here as $e_r(x)$, $r = [1, R]$, where $e_r(x)$ can be within any of the prediction forms above. The problem of decision fusion is then on locating the best possible class information C_j,

$j \in \Lambda = [1\,,\, J+1]$ for an input sample x by evaluating the combined predictions, $E(x) = \{e_1 \cup e_2 \cup ... \cup e_{R-1} \cup e_R\}$ from all classifiers, *i.e.*, κ_1 to κ_R. Different decision fusion methods exist for this purpose and three of the most popular methods are described next.

8.2.1 Voting-based Fusion

The most commonly used method to combine predictions is through voting. Voting-based approaches are also referred to as heuristic methods as voting-based methods often use procedures that mimic the behavior of humans while making joint inferences (Hall, 1992). When R predictions, *i.e.*, $e_1,...,e_R$, are produced from R classifiers, the decisions from all classifiers can be counted as votes through application of a majority or plurality decision rule. To implement voting, the results produced from all classifier $e_r(x)$, $r \in [1\,,\, R]$, on a class C_j are represented as a vector in binary form T_j^r:

$$T_j^r = \begin{cases} 1, & when\ e_r(x) = C_j,\ j \in [1, J] \\ 0, & otherwise \end{cases}. \qquad \text{(Eq 8-1)}$$

In most voting-based approaches, the final decision (class assignment) needs to be approved by at least half of the classifiers (majority voting). It is expressed in mathematical form as (Suen *et al.,* 1990; Xu *et al.,* 1992):

$$E(x) = \begin{cases} j, & if\ C_j = \arg\max_{j \in \Lambda} T_j^E(e_r(x) \in C_j) > R/2 \\ J+1, & otherwise \end{cases}, \qquad \text{(Eq 8-2)}$$

where $T_j^E(e_r(x) \in C_j) = \sum_{r=1}^{R} T^r(e_r(x) \in C_j)$. Eq 8-2 can be relaxed from its majority threshold of *R/2* if plurality voting is applied. In such a scenario, the expression for voting can be simplified to:

$$E(x) = C_j, \ \ if \ \ C_j = \arg\max_{j \in \Lambda} T_E(e_r(x) \in C_j), \ j \in [1, J], \ r = [1, R]. \text{(Eq 8-3)}$$

The combined predictions from $E(x)$ is then the resulting class C_j which receives maximum number of votes.

8.2.2 Bayesian-Inference based Fusion

The voting technique is based solely on the predictions produced from each classifier, and each classifier is treated equally without considering its previous performance. However, it has been well recognized that predictions from some classifiers generally outperform other classifiers in certain areas and should be given more weight when they are used on certain classes. The class-specific performance of each classifier κ_r, $r = [1, R]$, can then be obtained based on its previous performance using conditional probability, with the final predictions estimated through Bayes rule of calculating posteriori probability. The conditional probability quantifies the probability of occurrence of an event based on some gathered evidences. For example, given two events A and B, with probability of B as $P(B)$ such that $P(B) > 0$, the conditional probability of A given B follows the following expression (Haddad, 2006):

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)}, \qquad \text{(Eq 8-4)}$$

where $P(B \mid A)$ is the conditional probability that event B will occur given evidence A. For a classification problem of J classes on measurement x, if all classes are mutually exclusive (two classes cannot occur concurrently), the Bayesian inference process for evaluating the conditional probability of a certain class j, $x \in C_j$, $j \in [1, J]$ from κ_r becomes:

$$P(x \in C_j \mid \kappa_r(x) = j) = \frac{P(\kappa_r(x) = j \mid x \in C_j)P(x \in C_j)}{\sum_{i=1}^{J} P(\kappa_r(x) = i \mid x \in C_i)P(x \in C_i)}. \qquad \text{(Eq 8-5)}$$

The conditional probability that implies $x \in C_j$, $j \in [1, J]$, as given by Eq 8-5 can often be estimated from previous performance of κ_r using a confusion matrix (*CM)*. A confusion matrix *CM* stores class-specific performance of a classifier κ_r, and is normally constructed by testing κ_r with some training dataset. Consider a training dataset containing *N* samples which is tested with classifiers κ_r. All predictions from all classifiers κ_r can be recorded in a confusion matrix as follows (Xu *et al.*, 1992):

$$CM^r = \begin{pmatrix} n_{11}^r & n_{12}^r & \cdots & n_{1(J+1)}^r \\ n_{21}^r & n_{22}^r & \cdots & n_{2(J+1)}^r \\ \vdots & \vdots & \ddots & \vdots \\ n_{J1}^r & n_{J2}^r & \cdots & n_{J(J+1)}^r \end{pmatrix} \quad \text{(Eq 8-6)}$$

where n_{ij}^r, $i = [1, J]$ and $j = [1, J+1]$ indicates the number of samples belonging to class C_i, but assigned to class C_j by κ_r. The diagonal elements of CM^r is then the true predictions from κ_r on i^{th} class. If the constructed CM^r from a training dataset is composed of a diagonal matrix (except for $n_{i(J+1)}^r$), one can conclude that the results generated from classifier κ_r is 100% accurate. It follows from Eq 8-6 that the total number of samples, *N*, encountered by κ_r is:

$$N = \sum_{i=1}^{J} \sum_{j=1}^{J+1} n_{ij}^r \quad \text{(Eq 8-7)}$$

in which the total number of samples in each class C_i is:

$$n_{i.}^r = \sum_{j=1}^{J+1} n_{ij}^r, \; i = [1, J] \quad \text{(Eq 8-8)}$$

and the number of samples that is assigned to class C_j by κ_r is:

$$n_{.j}^r = \sum_{i=1}^{J} n_{ij}^r, \; j = [1, J+1]. \quad \text{(Eq 8-9)}$$

Utilizing information stored in CM^r, the conditional probability that implies $\boldsymbol{x} \in C_j$, $j \in [1\ ,\ J]$ given the evidence that $\kappa_r(x) = j$ can be computed as (Xu *et al.*, 1992):

$$P(x \in C_j \mid \kappa_r(x) = j) = \frac{n_{ij}^r}{n_{\cdot j}^r} = \frac{n_{ij}^r}{\sum_{i=1}^{M} n_{ij}^r},\ i, j \in [1\ ,\ J]. \qquad \text{(Eq 8-10)}$$

Analogous to Eq 8-5, $P(x \in C_j \mid \kappa_r(x) = j)$ can be implied as the confidence of a classifier regarding the assignment of sample x to class C_j. When R classifiers are in use, there will be R confusion matrixes, CM^r, $r = [1\ ,\ R]$, and R evidences, $e = e_1(x), ..., e_R(x)$. Each classifier κ_r expresses its predictions supporting the proposition that $x \in C_j$ in the form of conditional probability. The combined probability P_E that supports $x \in C_j$ can be written as:

$$P_E(x \in C_j \mid e_r(x) = j_r, E) = P(x \in C_j \mid e_1(x) \cap ... \cap e_R(x) = j),\ j = [1\ ,\ J] \text{(Eq 8-11)}$$

where E denotes the common classification environment that consists of all R events. If all classifiers are mutually exclusive (the fault classes in use does not overlap among each other), the events $e_1(x) = C_j, ..., e_K(x) = C_j$ will be mutually exclusive as well and the combined belief function of Eq 8-11 can be written as (Xu *et al.*, 1992):

$$\begin{aligned} & P_E(x \in C_j \mid e_r(x) = j, E) \\ & = P(x \in C_j \mid e_1(x) = j, ..., e_R(x) = j, E), \quad j = [1\ ,\ J] \\ & = \frac{P(e_1(x) = j, ..., e_R(x) = j \mid x \in C_j, E) P(x \in C_j \mid E)}{P(e_1(x) = j, ..., e_R(x) = j \mid E)} \\ & = P(x \in C_j \mid E) \frac{\prod_{r=1}^{R} P(x \in C_j \mid e_r(x) = j)}{\prod_{r=1}^{R} P(x \in C_j \mid E)}. \end{aligned} \qquad \text{(Eq 8-12)}$$

Here, $P(x \in C_j \mid e_r(x) = j)$ can be estimated from Eq 8-10, and $P(x \in C_j \mid E)$ is the probability that $x \in C_j$ is true under the environment E. For implementation purpose,

Xu *et al.* (1992) suggested the following estimate to be used for combining individual conditional probability:

$$P_E(x \in C_j \mid e_r(x) = j, E) = \frac{\prod_{r=1}^{R} P(x \in C_j \mid e_r(x) = j)}{\sum_{i=1}^{J} \prod_{r=1}^{R} P(x \in C_j \mid e_r(x) = j)}. \quad \text{(Eq 8-13)}$$

Based on Eq 8-13, a sample x can be classified into class j depending on the calculated combined conditional probability. The class C_j with the highest P_{E_j} can be selected as the optimal combined prediction:

$$E(\boldsymbol{x}) = \begin{cases} j, & \text{if } C_j = \arg\max_{j \in \Lambda}(P_{E_j}) \\ J+1, & \text{otherwise} \end{cases}. \quad \text{(Eq 8-14)}$$

8.2.3 Dempster-Shafer's Fusion

The Dempster-Shafer method is a popular method in uncertainty reasoning. It is often used to combine separate pieces of evidence. The theory of Dempster-Shafer is based on belief functions and was originally developed by Dempster (Dempster, 1968), and later refined by Shafer (Shafer, 1976). The method can be used to combine classifiers of all types. Let Θ be a finite set of elements with members composed by hypothesis, objects, or fault classes. The union of all elements, Θ, is referred to as *frame of discernment* in the context of Dempster-Shafer theory, and the set that contains all the subsets of Θ is called the *power set* of Θ. Let a set of exhaustive and mutually exclusive elements containing J classes be given as A_j, $j = [1, J]$. The universal set containing all singleton elements is then given as $\Theta = \{A_1, ..., A_J\}$. The subset $\{A_{i_1}, ..., A_{i_q}\} \in \Theta$ denotes a combined proposition that support $A_{i_1} \cup ... \cup A_{i_q}$ and is called a superset of Θ. A Θ of size J has exactly 2^J subsets.

The Dempster-Shafer theory normally uses a belief function $bel(A)$ to indicate the belief of a classifier κ_r, $r = [1, R]$, on a proposition $A_j \in \Theta$, $j \in [1\ J]$. $bel(A)$

can also be referred to as basic probability assignment (BPA) or $m(A)$. $m(A)$ can be considered as the degree of belief held by κ_r regarding evidence A. Some properties of BPA are:

$$m(\varnothing) = 0, \quad \sum_{A} m(A) = 1. \qquad \text{(Eq 8-15)}$$

The BPA is a generalization of probability mass distribution and each element of Θ can be assigned a value between [0 , 1] such that the values sum up to 1. Any subset $A \in 2^N$ with $m(A) > 0$ is called a *focal element*. Since a subset A represents the disjunction of all elements in A, the truth of $B \subseteq A$ implies the truth of A (Xu *et al.*, 1992). Consequently, $bel(A)$ can be calculated from all focal elements that support A:

$$bel(A) = \sum_{B \subseteq A} m(B). \qquad \text{(Eq 8-16)}$$

A fundamental difference between the Dempster-Shafer theory and the Bayesian theory is in their treatment of ignorance (Giarratano and Riley, 1998). Bayesian theorem requires the sum of all probabilistic events to be equal to one, or $\sum_{j=1}^{J} P(\boldsymbol{x} \in C_j \mid e(\boldsymbol{x}) = j)$, when there are J possible events. However, Dempster-Shafer theory allows the method to operate under an incomplete probabilistic model by allowing both belief and disbelief of element A_i to be lesser than unity, *i.e.*, $m(A_i) + m(\neg A_i) < 1$, as the set which corresponds to disbelief of A_i, *i.e.*, $\neg A_i = \Theta - A_i$ only consists of two elements of the universal set. Apart from the belief function $bel(A)$, Dempster-Shafer theory also defines a plausibility function, $pls(A)$, which is given by:

$$pls(A) = 1 - bel(\neg A) = \sum_{B \cap A \neq \varnothing} m(B) \qquad \text{(Eq 8-17)}$$

to measure the maximum amount of probability that can be distributed among the elements in A, which constitutes an upper limit function on the probability of A. The belief and plausibility function described can be used for integrating various types of information.

For combination of R classifiers, the performance of each classifier κ_r, $r \in [1, R]$ can be measured in terms of recognition rate, ε_r^r, substitution rate, ε_s^r, and rejection rate, ε_t^r, based on performance from previous predictions, where the superscript r represents that the measurement is obtained from r^{th} classifier. The basic BPA assignment supporting the proposition $\boldsymbol{x} \in C_j$ and $\boldsymbol{x} \notin C_j$, $j \in [1, J]$ for predictions of sample $\boldsymbol{x}$ based on classifier κ^r can then be expressed as (Xu *et al.*, 1992):

$$m^r(\boldsymbol{x} \in C_j) = m^r(A_j) = \varepsilon_r^r,$$

$$m^r(\boldsymbol{x} \notin C_j) = m^r(\neg A_j) = \varepsilon_s^r,$$

$$m^r(\Theta) = 1 - m^r(A_j) - m^r(\neg A_j) = 1 - \varepsilon_r^r - \varepsilon_s^r. \qquad \text{(Eq 8-18)}$$

When R classifiers are applied, $3R$ BPAs ($m^r(x \in C_j)$, $m^r(x \notin C_j)$, and $m^r(\Theta)$) are generated, and the Dempster rule of combination can be used to combine them. The Dempster rule defines the combined BPAs of two classifiers, m_1 and m_2, as (Shafer, 1976):

$$m(A) = m_1 \oplus m_2(A) = k \sum_{X \cap Y = A, A \neq \varnothing} m_1(X) m_2(Y)$$

$$k^{-1} = 1 - \sum_{X \cap Y = \varnothing} m_1(X) m_2(Y) = \sum_{X \cap Y \neq \varnothing} m_1(X) m_2(Y) \qquad \text{(Eq 8-19)}$$

where the symbol $\oplus$ represents the combination of two belief fuctions, with the condition that m is a BPA if $k^{-1} \neq 0$. If $k^{-1} = 0$, then $m_1 \oplus m_2$ does not exist and m_1

and m_2 are said to be totally contradictory. As an illustrative example, to merge R classifiers, the combined BPA of the first two classifiers κ_1 and κ_2, *i.e.*, $m_1 \oplus m_2$ is first calculated as (Xu *et al.*, 1992):

$$k_2 = \frac{1}{1 - m_{\kappa_1}(A_{j_r'}) m_{\kappa_2}(\neg A_{j_r'}) - m_{\kappa_1}(\neg A_{j_r'}) m_{\kappa_2}(A_{j_r'})} = \frac{1}{1 - \varepsilon_r^{\kappa_1}\varepsilon_s^{\kappa_2} - \varepsilon_s^{\kappa_1}\varepsilon_r^{\kappa_2}}$$

$$m_2(\Theta) = k_2 m_{\kappa_1}(\Theta) m_{\kappa_2}(\Theta) = k_2(1 - \varepsilon_r^{\kappa_1} - \varepsilon_s^{\kappa_1})(1 - \varepsilon_r^{\kappa_2} - \varepsilon_s^{\kappa_2})$$

$$m_2(\neg A_{j_k'}) = k_2 m_{\kappa_1}(\neg A_{j_k'})[m_{\kappa_2}(\Theta) + m_{\kappa_2}(\neg A_{j_k'})] + k_2 m_{\kappa_2}(\neg A_{j_k'}) m_{\kappa_1}(\Theta)$$

$$= k_2 \varepsilon_s^{\kappa_1}(1 - \varepsilon_r^{\kappa_2}) + k_2 \varepsilon_s^{\kappa_2}(1 - \varepsilon_r^{\kappa_1} - \varepsilon_s^{\kappa_1})$$

$$m_2(A_{j_k'}) = k_2 m_{\kappa_1}(A_{j_k'})[m_{\kappa_2}(\Theta) + m_{\kappa_2}(A_{j_k'})] + k_2 m_{\kappa_2}(A_{j_k'}) m_{\kappa_1}(\Theta)$$

$$= k_2 \varepsilon_r^{\kappa_1}(1 - \varepsilon_s^{\kappa_2}) + k_2 \varepsilon_r^{\kappa_2}(1 - \varepsilon_s^{\kappa_2}) + k_2 \varepsilon_r^{\kappa_2}(1 - \varepsilon_s^{\kappa_1} - \varepsilon_r^{\kappa_1})$$

$$m_2(A) = 0 \text{ for all other } A \subset \Theta. \qquad \text{(Eq 8-20)}$$

Subsequently, the combination for the rest of the BPAs, $m_3, \ldots, m_k$, is calculated recursively in the same way based on the general formula:

$$k_r = \frac{1}{1 - m_{r-1}(A_{j_k'}) m_{\kappa_r}(\neg A_{j_k'}) - m_{r-1}(\neg A_{j_k'}) m_{\kappa_r}(A_{j_k'})}$$

$$m_r(\Theta) = k_r m_{r-1}(\Theta) m_{\kappa_r}(\Theta)$$

$$m_r(\neg A_{j_k'}) = k_r m_{r-1}(\neg A_{j_k'})[m_{\kappa_r}(\Theta) + m_{\kappa_r}(\neg A_{j_k'})] + k_r m_{\kappa_r}(\neg A_{j_k'}) m_{r-1}(\Theta)$$

$$m_r(A_{j_k'}) = k_r m_{r-1}(A_{j_k'})[m_{\kappa_r}(\Theta) + m_{\kappa_r}(A_{j_k'})] + k_r m_{\kappa_r}(A_{j_k'}) m_{r-1}(\Theta)$$

$$m_2(A) = 0 \text{ for all other } A \subset \Theta. \qquad \text{(Eq 8-21)}$$

Interested readers are referred to Shafer (1976), Smets and Kennes (1994), and Yager (2001) for more details on Dempster-Shafer theory.

8.3 Decision Fusion of Diagnostic Classifiers

Since different classifiers generally exhibit strengths in different areas through extracting and analyzing different features from plant measurements, combining diagnostic classifiers of different forms is highly favorable. In this chapter, the diagnostic classifiers being considered utilize distinct features of the process data, i.e., mainly based on features extracted in the original signals space, in its reduced subspace as principal components, and the sequence of fault progression based on fault signature sequence analysis. The neural-network (Srinivasan *et al.*, 2005a,b), Principal Components Analysis (Qin, 2003), and Self-Organizing Maps (Ng and Srinivasan, 2004a) can be used for these multi-faceted, multi-feature classification of process operating data. Each of the FDI classifier is represented as an agent A in the multi-agent environment, and their final results combined through consolidator agent, A^c based on the decision fusion strategies presented above.

Combination of fault detection and diagnosis results is challenging as compared to ordinary decision fusion of data classifiers. When FDI is performed on chemical processes, there exist two types of results that need to be fused, namely, the monitoring results and the diagnosis results. In the context of CAMEO, monitoring results are produced by monitoring agents A_r^m, while fault relevant diagnostic results are produced by diagnostic agents A_r^d. During online monitoring, the decision fusion algorithm needs to be optimized between the speed of fault detection and misclassification rates. In general, these objectives are often contradictory as classifiers that are sensitive to process disturbances are also prone to higher level of false positives. In contrast, diagnostic classifiers that are less sensitive to process disturbances are in turn prone to false negatives. Decision fusion schemes that strive to

reduce both Type-I (false positives) and Type-II (false negatives) errors are generally deemed more efficient.

Suppose there are R monitoring agents A_r^m, $r \in [1, R]$ as implemented in CAMEO, the monitoring results $S^{A_r^m}$ produced from each monitoring agent A_r^m can be represented as a binary variable:

$$S(A_r^m) = \begin{cases} 1, \textit{ when process is abnormal} \\ 0, \textit{ when process is normal} \end{cases}. \qquad \text{(Eq 8-22)}$$

The $S^{A_r^m}$, $r = [1, R]$ can then be combined through one of the decision fusion techniques as described in Section 8.2. If the combined monitoring results suggest that the process is at fault, *i.e.*, when $E(S(A_r^m)) = 1$, the diagnostic agents in CAMEO will be activated for fault identification. Suppose there are R diagnostic agents A_r^d, $r \in A^d = [1, R]$ in CAMEO and let the fault database consisting J fault classes be $DB = \{F_1, ..., F_J\}$, where each of F_j, $j \in \Lambda = [1, J]$ represents an independent and mutually exclusive class of fault. Each diagnostic classifier A_r^d will try to retrieve the classes of F_j, $j \in [1, J+1]$ within DB that gives high similarity to the measurements x based on its features extraction, and intelligent search method. As mentioned earlier, $F_j = J+1$ corresponds to novel fault (unknown disturbances). Let the class prediction results from each diagnostic agent A_r^d be represented by $e^r(x \in F_j^r)$, the decision fusion process of consolidator agent A_r^c is then to locate the most probable faults, F^{opt} from a list of $\{1, ..., J+1\}$ classes by combining the predictions from all A_r^d:

$$E(x \in F_j) = \{e^1(x \in F_j^1) \cup ... \cup e^R(x \in F_j^R). \qquad \text{(Eq 8-23)}$$

8.3.1 Voting Strategy

The voting strategy is based on calculating the votes contributed by all FDI agents towards each fault classes in the fault database. Throughout this chapter, the strategy of plurality voting is used for decision fusion. Based on the plurality rule, the operations can be considered faulty whenever a plurality of the monitoring agents A_r^m reports an abnormality in the measurement $\boldsymbol{x}$ observed:

$$S^V(x) = \begin{cases} 1, & if \max\limits_{r \in A^m} S(A_r^m) = 1 \\ 0, & if \max\limits_{k \in A^m} S(A_r^m) = 0 \end{cases} \quad \text{(Eq 8-24)}$$

Similarly, the fault diagnosis results can also be combined based on plurality rule. The combined predictions (fault candidates), $E^V(x)$, are determined based on the votes generated from independent predictions $e(x \in F_j^r)$ of all fault diagnosis agents A_r^d.

$$E^V(x) = \begin{cases} j, & if\ E^V(x \in F_j) = \max\limits_{j \in \Lambda, r \in A^d} e(x \in F_j^r) \\ J+1, & otherwise \end{cases} \quad \text{(Eq 8-25)}$$

In general, the voting strategy is accurate in determining the best matching fault classes by being able to combine agreements among heterogeneous diagnostic agents. However, under total conflicting scenarios, *i.e.*, when the results produced by one diagnostic agent is in total conflict with another, *i.e.*, $e(A_1^d) \cap e(A_2^d) = \text{ø}$, voting strategy is unable to resolve such conflicts especially if more than one fault candidates have similar number of votes. In such scenarios, a subset of fault classes that receive equal number of votes is extracted:

$$F_V^{opt} = E^V(x) = \max_{j \in \Lambda, r \in A^d} e(x \in F_j^r) \ . \quad \text{(Eq 8-26)}$$

8.3.2 Bayesian-Combination Strategy

The Bayesian-combination strategy utilizes historical information from each FDI agent for decision fusion. The strategy adopted for detecting and diagnosing process fault is based on evaluating the conditional probability, $P(x \in F_j \mid e_r(x) = j)$, of the fault event. Prior to online implementation, the fault predictions obtained from each diagnostic agent A_r^d to different classes of faults are first analyzed offline and its results collected into a confusion matrix, CM^r, as described in Section 8.2.2. Since the Bayes theorem allows multiple pieces of evidences to be fused by quantifying the product of all conditional probabilities from all classifiers (see Eq 8-13), its direct implementation suffers from longer delay in fault detection as any accurate detection from a monitoring agent will be discounted against the disbelief of other monitoring agents. As an illustrative example, for a system consisting of three monitoring agents, A_1^m, A_2^m, and A_3^m, it follows from Eq 8-13 that the fault can only be detected when all monitoring agents are able to detect the fault (Table 8-1).

Table 8-1: Bayesian combination of three monitoring agents A_1^m, A_2^m, and A_3^m

Scenario #	$S(A_1^m)$	$S(A_2^m)$	$S(A_3^m)$	$E(\boldsymbol{x})$
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0
4	0	1	1	0
5	1	0	1	0
6	1	1	1	1

One way to mitigate this problem is by taking the weighted average of the conditional probability produced by all monitoring agents, A_r^m, $r = [1, R]$, instead of their product. When new measurements *x* are obtained, the conditional probability for a

fault from class j to occur, $P(x \in F_j \mid e_k(x) = j)$, can be calculated based on Eq 8-10. The normal/abnormal status of a process can then be decided based on:

$$S^B(x) = \begin{cases} 1, & if\ P(x \in F_j \mid e_r(x) = j) > \delta,\ \ \forall j \in \Lambda,\ \forall r \in [1, R] \\ 0, & otherwise \end{cases}, \quad \text{(Eq 8-27)}$$

where δ is a threshold selected for determining the presence of fault.

Upon the detection of abnormal events, the results produced by all A_r^d can be collected as a fault candidate pool *CP*, and the fault candidate F_j with highest combined conditional probability selected:

$$F_B^{opt} = \arg\max_{j \in \Lambda, F_j \in CP} \sum_{r=1}^{R} \frac{1}{FC_r} P(x \in F_j \mid e_r(x) = j_r) \qquad \text{(Eq 8-28)}$$

where FC_r is the number of fault candidates recommended by diagnostic agent A_r^d.

8.3.3 Dempster-Shafer Strategy

The Dempster-Shafer strategy uses belief function to merge decisions. The combination strategy collects evidence based on the recognition rate, ε_r^r, substitution rate, ε_s^r, and rejection rate, ε_t^r, of each FDI agent. The level of information being used by Dempster-Shafer technique is therefore more ambiguous compared to Bayesian combination technique as only a broad, superficial, and non-class specific information is utilized in the Dempster-Shafer approach. In this thesis, a fault detection criterion similar to Eq 8-27 is adopted for fault detection. When new measurements x are obtained, the basic BPA assignment $m(A_r^m)$ from each monitoring agent A_r^m supporting the proposition $x \in F_j$, $j \in [1, J]$ can first be obtained based on Eq 8-18. The normal/abnormal status of a process can then be decided based on:

$$S^{DS}(x) = \begin{cases} 1, & if\ m^r(x \in F_j) > \delta,\ j \in \Lambda\ ,\ r \in [1, R] \\ 0, & otherwise \end{cases}. \qquad \text{(Eq 8-29)}$$

When a fault is detected, *i.e.*, $S^{DS}(x)=1$, the fault class F_j that yields the largest combined BPA is selected:

$$F_{DS}^{opt} = \arg\max_{j\in\Lambda, r\in CP} m^1(x\in F_j)\oplus\ldots\oplus m^r(x\in F_j)\oplus\ldots\oplus m^R(x\in F_j), \text{(Eq 8-30)}$$

where *CP* is the candidate pool of diagnosis results produced by all A_r^d. The combined BPA can then be calculated by recursive computation of Eq 8-20 and Eq 8-21 for all fault candidates present in *CP*.

8.4 Measuring Inter-classifiers Agreement

When multiple diagnostic classifiers are involved, it is necessary to measure the reliability of the predictions among different classifiers. For this purpose, the Cohen's Kappa statistic (Cohen, 1960), K^p, can be used as a measure to quantify the inter-classifiers agreement. The Kappa statistic is based on the calculation of the difference between the observed agreement and the agreements expected by chance. The mathematical representation of Kappa is given as (Cohen, 1960):

$$K^p = \frac{p_o - p_c}{1 - p_c} \quad \text{(Eq 8-31)}$$

where p_o is the observed proportion of agreement, and p_c is the proportion of agreement expected by chance. The standard error for an observed K^p for large sample (of size *N*) follows (Cohen, 1960):

$$\sigma_{K^p} \cong \sqrt{\frac{p_o(1-p_o)}{N(1-p_c)^2}} \quad \text{(Eq 8-32)}$$

The value of K^p is bounded in the range of [-1 , 1], where $K^p=1$ represents the case of perfect agreement for all cases, while $K^p=0$ is the exact scenario of agreement by chance, and $K^p<0$ indicates agreement less than chance, possibly a

systematic disagreement between two classifiers (Viera and Garrett, 2005). Since different values of Kappa correspond to different potency of agreements, a commonly used benchmark for interpretation of Kappa statistic has been proposed, as shown in Table 8-2 (Landis and Koch, 1977).

Table 8-2: Interpretation of Kappa value

Kappa Value	**Interpretation**
<0.00	Less than chance agreement
0.00-0.20	Slight agreement
0.21-0.40	Fair agreement
0.41-0.60	Moderate agreement
0.61-0.80	Substantial agreement
0.81-1.00	Almost perfect agreement

An illustrative example as presented in Howell (2001) is used here to facilitate the understanding of calculating the Kappa statistic. Suppose two observers (classifiers) analyzed a group of measurements containing 30 samples with the prediction results shown in Table 8-3.

Table 8-3: Results of analysis presented by two classifiers

	Classifier I			
Classifier II	*Normal*	*Fault I*	*Fault II*	**Total**
Normal	15 (c_{11})	2 (c_{12})	3 (c_{13})	20 (m_1)
Fault I	1 (c_{21})	3 (c_{22})	2 (c_{23})	6 (m_2)
Fault II	0 (c_{31})	1 (c_{32})	3 (c_{33})	4 (m_3)
Total	16 (n_1)	6 (n_2)	8 (n_3)	30 (N)

The diagonal elements in Table 8-3 represents the frequencies that both classifiers agree with each other while the off-diagonal elements represent the frequencies of disagreement. Such a table is often referred to as a *contingency table* (Howell, 2001) or *agreement table* (Cohen, 1968).

The steps for evaluating Kappa statistic can be summarized as follows:

1. Computation of p_o: The observed proportion of agreement, p_o, is simply the summation of the diagonal elements of the contingency table over the total number of samples analyzed (Table 8-3):

$$p_o = \frac{1}{N}\sum_{i-1}^{r} c_{ii} = \frac{1}{N}\sum (c_{11} + c_{22} + c_{33}) = 21/30 = 0.7.$$

2. Computation of p_c : The proportion of agreement by chance, p_c , can be calculated with the probability that a judge would chose a particular class. As such, the p_c can be quantified through its probability: $p_c = \sum_{i=1}^{r} n_i m_i / N^2$, which gives a probability of $(20\times16)/30^2 = 0.3556$ for Normal, $(6\times6)/30^2 = 0.04$ for Fault I, and $(8\times4)/30^2 = 0.0356$ for Fault II as in the case of Table 8-3.

Based on the two steps illustrated above, and utilizing Eq 8-31, the value of Kappa is estimated as $K^p = \frac{p_o - p_c}{1 - p_c} = \frac{0.7 - 0.43}{1 - 0.43} = 0.47$, which corresponds only to the region of moderate agreement if the classification table (Table 8-2) is used. Note that such a value is very different from the simplistic approach of measuring percentage agreement, which would be $(15 + 3 + 3)/30 = 0.7 = 70\%$, suggesting a strong agreements between the two judges.

8.5 Case Study 1: Fault Diagnosis in Tennessee Eastman Plant

The control system used here is based on Lyman and Georgakis (1995), as implemented in the modified simulator of Chiang and Braatz (2003). Fifteen process faults, as proposed by Downs and Vogel (1993) are tested here (Table 8-4). Since the multiple SISO contol strategy is able to provide good recovery actions to disturbances IDV3, IDV4, IDV9, IDV15, IDV16, & IDV19, these six IDVs are excluded from the analysis.

Each training dataset simulates 1500 min of process operations with a sampling interval of 3 min. All faults were introduced at 1 hour of operating time for the training

data. In comparison, the testing dataset used for each run is created by simulating the process for 48 operating hours (2880 min) with the faults introduced at 480 min. The signals for each test run are thus different from the training data in terms of run-length and time of fault introduction.

Table 8-4: Process disturbances considered for TE process

Variable number	**Process variable**	**Type**
IDV(1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV(2)	B composition, A/C ratio constant (stream 4)	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss (stream 1)	Step
IDV(7)	C header pressure loss-reduced availability (stream 4)	Step
IDV(8)	A, B, C feed composition (stream 4)	Random variation
IDV(10)	C feed temperature (stream 4)	Random variation
IDV(11)	Reactor cooling water inlet temperature	Random variation
IDV(12)	Condenser cooling water inlet temperature	Random variation
IDV(13)	Reaction kinetics	Slow drift
IDV(14)	Reactor cooling water valve	Sticking
IDV(17)	Unknown	Unknown
IDV(18)	Unknown	Unknown
IDV(20)	Unknown	Unknown
IDV(21)	Valve for Stream 4 fixed at steady-state position	Constant position

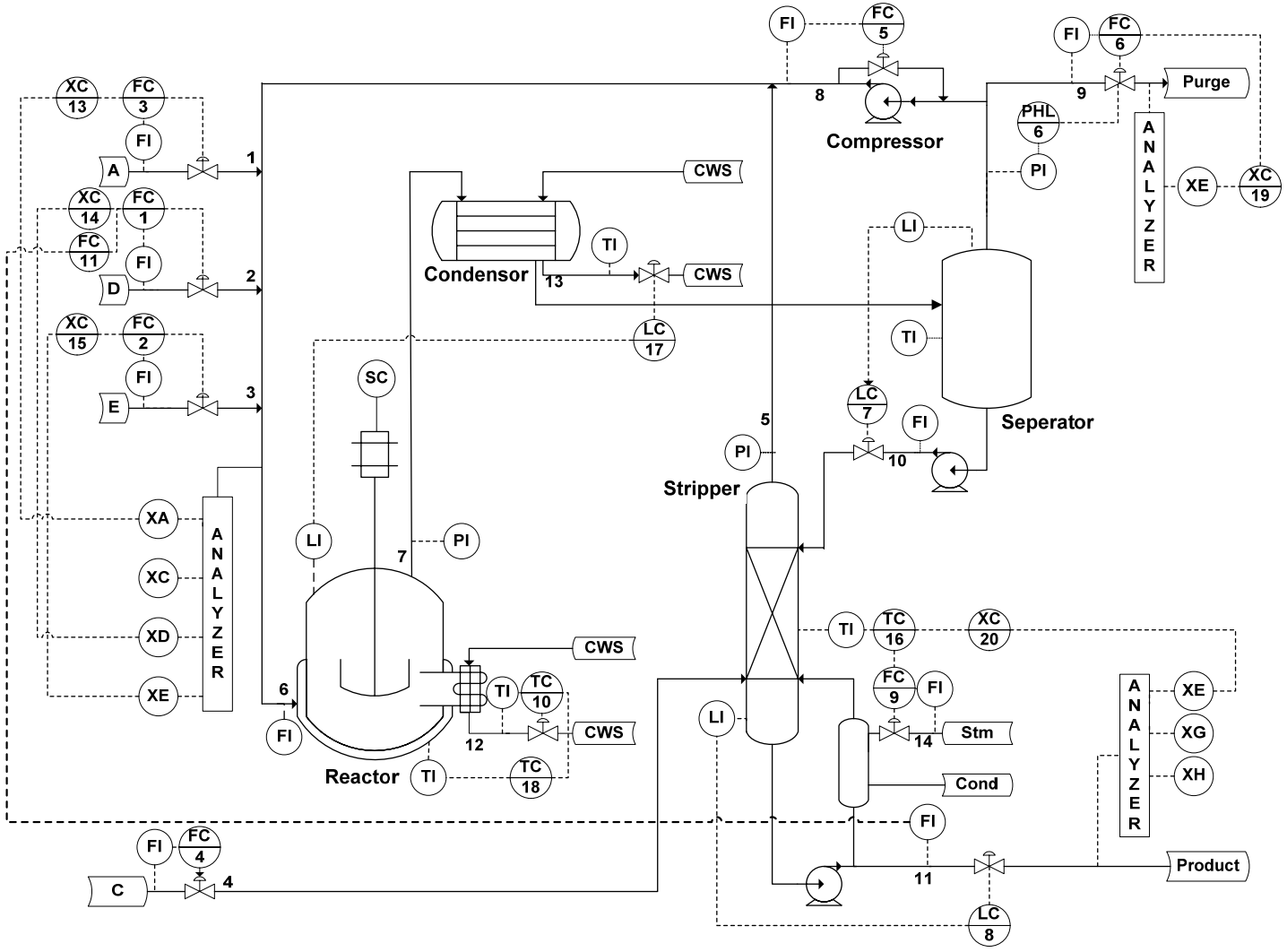


Figure 8-1: Process flowsheet of Tennessee Eastman process

FDI Agents Implemented

Three FDI agents are used in this case study for monitoring and fault diagnosis - Neural-Networks (NN), Principal Components Analysis (PCA), and Self-Organizing Maps (SOM).

1. *NN agent*: The neural-network monitoring agent, A_{NN}^{m}, is trained with the time-series data of the fault F_j, $j \in \Lambda = [1, 16]$. A filtering method is first used to filter out the portion of F_j corresponding to normal operations. The reason for adopting the filtering technique is to segregate the portion of fault data from the training dataset so that only the distinctive fault portion of data is used for neural-network training. During filtering, the Euclidean distance between each sample of F_j is compared to the normal operating condition and the samples that show high similarity with normal operations are removed from F_j. The filtered data are then range normalized before they are trained with a backpropagation neural-network as described in Appendix A of this thesis. For this case study, a three layer neural-network with size [30 30 16], and using tan-sigmoid transfer function for the initial two layer, and linear transfer function for the final layer is constructed for fault identification.
2. *PCA agent*: The PCA monitoring agent, A_{PCA}^{m}, is trained with normal operating data x^R only. The x^R is first autoscaled and a PCA model is developed by retaining the first 16 PCs with the cumulative variance of $\frac{\sum_{i=1}^{16}\lambda_{PCi}}{\sum_{i=1}^{22}\lambda_{PCi}} = 95.18\% > 95\%$. PCA based fault detection is based on the limit violation of T^2 statistic and *SPE* value. For fault identification, the fault

reconstruction approach as proposed in (Qin, 2003) is used by constructing a PCA model PCA^{F_j} for each fault class F_j, $j \in [1, 16]$. The PCA^{F_i} model that shows an in-control status during abnormal operations is considered to flag the process fault.

3. *SOM agent*: The SOM monitoring agent, A^m_{SOM}, uses a two-dimensional map to perform cluster analysis on the online measurement *x*. The SOM is constructed based on the same training data as used for training of neural-network monitoring agent, A^m_{NN} (applying similar filtering and normalization method). The fully trained SOM consists of 26x17 map units. Fault identification is based on abnormal cluster projection while fault identification during online application is based on cluster sequence analysis based on the fault signature generated from $\boldsymbol{x}$ in the SOM space.

Diagnostic Results and Discussion

The fault diagnosis results of each FDI agent working in isolation is shown in Table 8-5 to Table 8-7. When a single FDI agent is used, the SOM agent identifies the highest number of faults (14 IDVs) with an average detection delay of 45.8 samples and an average diagnosis delay of 65.7 samples (Table 8-11).

The PCA agent only distinguishes 10 disturbances (IDVs), but is able to give the shortest detection delay among the three agents with 26.8 samples in average. The reconstructed fault models in the PC subspace are shown in Figure 8-2. It can be observed that majority of the IDVs model overlap with other fault classes in the principal components subspace. Such high degree of overlap results in a low recognition rate (74.8%) when it is used for fault identification as PCA-based fault classifier tend to extract more than one IDVs for a given measurement x.

In contrast, neural-network agent A_{NN}^{d} yields highest recognition rate (95.7%) and fastest diagnostic delay (35.5 samples) compared to other FDI agents. However, the number of faults identified with neural-network agent is low with only eleven IDVs identified successfully.

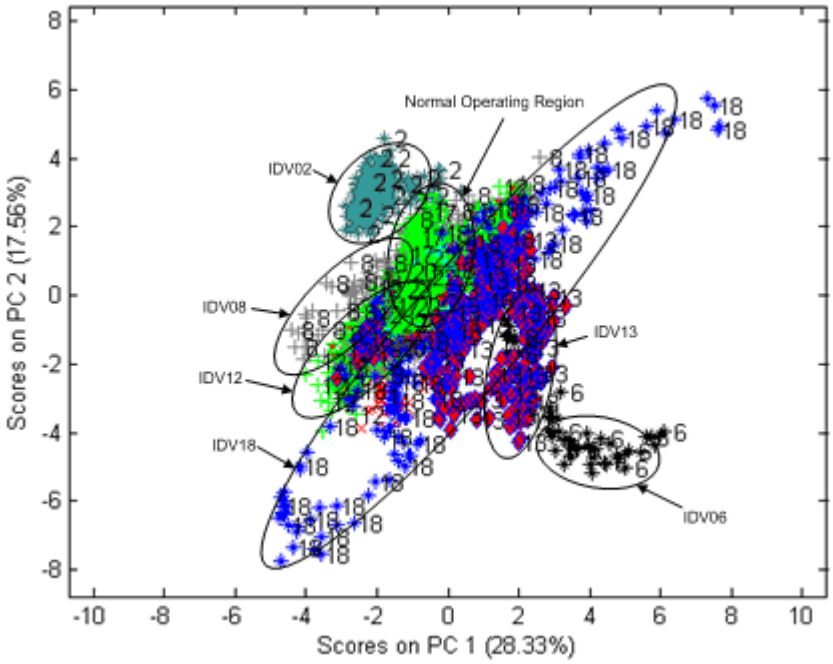


Figure 8-2: Reconstructed fault models in the PC subspace

Table 8-5: Performance of Neural-Network agent A_{NN}^{d} in TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	788	0	1	0	0	7	0	0	0	0	0	0	0	0	0
2	0	773	0	0	0	2	0	0	0	0	0	0	0	0	0
5	0	0	94	0	7	0	0	0	14	0	0	0	0	0	0
6	0	0	0	783	0	7	0	0	0	0	0	0	0	0	0
7	0	0	7	0	174	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	632	0	0	8	0	0	0	0	0	0
10	0	0	6	0	3	2	0	0	36	0	0	0	0	0	0
11	0	0	0	0	1	0	0	6	53	0	0	3	0	0	0
12	0	0	0	0	4	0	0	0	657	0	0	0	0	0	0
13	0	0	5	0	0	1	0	0	0	715	0	0	0	0	0
14	0	0	16	0	2	1	0	0	3	0	0	0	0	0	0
17	0	0	1	0	0	1	0	11	6	0	0	585	0	0	0
18	0	0	0	0	0	0	0	0	5	0	0	0	699	0	0
20	0	0	8	0	3	26	0	0	2	0	0	0	1	0	0
21	0	0	0	0	22	0	0	0	0	0	0	0	0	0	263

Table 8-6: Performance of Principal Components Analysis agent A^d_{PCA} in TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	709	0	1	0	3	52	0	0	3	5	0	0	0	0	1
2	0	782	4	0	6	25	4	0	6	17	1	0	3	3	2
5	0	0	150	0	130	0	0	0	180	0	0	0	5	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	2	751	0	0	0	1	0	0	0	0	0
10	0	0	17	0	157	29	310	0	6	278	0	0	0	0	0
11	0	0	0	0	0	0	0	114	2	0	114	29	0	0	0
12	0	0	0	0	0	0	0	0	739	0	0	0	7	0	0
13	0	0	1	0	0	2	0	0	0	146	0	0	2	0	0
14	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0
17	0	0	0	0	0	0	0	0	0	0	22	704	0	0	0
18	0	1	0	0	0	0	0	0	9	0	0	0	345	0	0
20	0	0	0	0	0	0	0	0	1	4	0	0	0	340	0
21	0	0	32	0	283	22	16	0	47	160	14	1	12	24	66

Table 8-7: Performance of Self-organizing Maps agent A^d_{SOM} in TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	777	0	1	1	1	16	1	1	1	1	1	0	1	1	1
2	0	760	2	0	0	12	2	0	2	2	2	2	2	0	2
5	0	0	66	0	3	4	0	0	4	0	0	0	0	0	0
6	1	0	26	772	1	1	2	26	1	1	2	0	1	26	1
7	2	0	4	2	167	2	4	2	39	2	2	0	2	4	2
8	0	1	1	0	0	332	16	0	16	1	1	1	1	0	1
10	0	0	0	0	0	1	37	0	0	6	0	0	0	0	1
11	0	0	0	0	0	0	0	122	0	0	0	7	0	0	0
12	0	0	4	0	4	13	0	0	427	0	0	0	0	0	0
13	1	0	7	1	2	1	2	1	1	562	1	0	1	6	2
14	0	0	0	0	0	0	0	1	1	0	17	1	0	0	0
17	0	1	1	0	0	1	7	0	1	1	2	549	1	6	1
18	1	0	1	1	3	21	3	3	33	2	1	0	705	3	1
20	1	0	2	1	1	1	2	1	1	1	1	0	1	4	1
21	1	0	1	1	1	1	1	1	1	1	1	0	1	1	267

The combined FDI results based on various decision fusion strategies are shown in Tables 7-7 to 7-9, while the summary of disturbance diagnosis based on various FDI strategies are shown in Table 8-11. The voting based fusion approach failed to diagnose IDV14, while Dempster-Shafer based fusion failed to diagnose IDV 20. In comparison, collaborative fault diagnosis based on Bayesian combination yields

the best results with i) highest number of identified faults (15 IDVs), ii) highest prediction accuracy, and iii) shortest detection and diagnosis delay (25.3 samples and 28.0 samples respectively). Most of the faults can be successfully identified with the decision fusion methods. Bayesian fusion is able to diagnose all of the faults by effectively combining the strengths from each FDI classifiers. Both Dempster-Shafer and voting based decision fusion show improvements in reducing time of diagnosis delay and recognition rate compared to the use of any single approaches. Since voting-based decision fusion requires plurality votes from all agents for fault detection and identification, the method suffers from higher detection delay compared to other fusion methodologies as any early detection of fault by one agent is not a sufficient condition to conclude the existence of fault.

However, voting-based decision fusion yields highest recognition rate among the three decision fusion strategies as the method sought agreements from most FDI agents prior to contributing to the diagnosis results. Misclassification rates are therefore reduced for voting technique at the expense of longer detection delay..

Table 8-8: Performance of Voting-based decision fusion for TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	788	0	0	0	0	32	0	0	0	0	0	0	0	0	0
2	0	773	0	0	0	2	0	0	0	0	0	0	0	0	0
5	0	0	111	0	6	0	0	0	22	0	0	0	0	0	0
6	0	0	0	783	1	7	0	0	0	0	0	0	0	0	0
7	0	0	6	0	212	1	1	0	4	0	0	0	0	1	0
8	0	0	0	0	0	682	0	0	8	0	0	0	0	0	0
10	0	0	0	0	12	4	68	0	32	30	0	0	0	0	0
11	0	0	0	0	1	0	0	88	26	0	6	8	0	0	0
12	0	0	0	0	3	0	0	0	711	0	0	0	0	0	0
13	0	0	5	0	0	1	0	0	0	726	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	1	0	11	3	0	0	624	0	0	0
18	0	0	0	0	0	0	0	0	8	0	0	0	708	0	0
20	0	0	2	0	1	26	0	0	0	0	0	0	1	33	0
21	0	0	0	0	62	0	0	0	0	37	0	0	0	0	290

Table 8-9: Performance of Bayesian-based decision fusion for TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	788	0	1	0	0	10	0	0	0	0	0	0	0	0	1
2	0	788	0	0	0	2	0	0	0	0	0	0	0	0	0
5	0	0	95	0	7	0	0	0	89	0	0	0	0	0	0
6	1	0	0	783	0	7	0	0	0	0	0	0	0	0	0
7	2	0	7	0	215	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	770	0	0	8	0	0	0	0	0	0
10	0	0	6	0	3	29	247	0	36	3	0	0	0	0	0
11	0	0	0	0	1	0	0	126	53	0	0	5	0	0	0
12	0	0	0	0	4	0	0	0	781	0	0	0	0	0	0
13	1	0	5	0	0	3	0	0	0	749	0	0	0	0	0
14	0	0	16	0	2	1	0	0	3	0	17	3	0	0	0
17	0	1	1	0	0	1	0	11	6	0	0	735	0	0	0
18	0	1	0	0	0	0	0	0	5	0	0	0	711	0	0
20	0	0	8	0	3	26	0	0	2	0	0	0	1	311	0
21	1	0	0	0	24	0	0	0	0	13	0	0	0	0	319

Table 8-10: Performance of Dempster-Shafer based decision fusion for TE problem

IDV	1	2	5	6	7	8	10	11	12	13	14	17	18	20	21
1	788	0	1	0	0	7	0	0	0	0	0	0	0	0	0
2	0	773	0	0	0	2	0	0	0	0	0	0	0	0	0
5	0	0	113	0	7	0	0	0	14	0	0	0	0	0	0
6	1	0	1	784	1	8	1	1	1	1	1	0	1	1	1
7	2	0	11	2	213	3	4	2	7	2	2	0	2	4	2
8	0	1	1	0	0	673	1	0	9	1	1	1	1	0	1
10	0	0	6	0	3	2	38	0	36	2	0	0	0	0	1
11	0	0	0	0	1	0	0	81	53	0	0	9	0	0	0
12	0	0	0	0	4	3	0	0	717	0	0	0	0	0	0
13	1	0	6	1	1	2	1	1	1	721	1	0	1	1	1
14	0	0	16	0	2	1	0	1	4	0	17	1	0	0	0
17	0	1	2	0	0	2	1	11	7	1	1	618	1	0	1
18	1	0	1	1	2	2	2	2	7	2	1	0	705	2	1
20	1	0	10	1	4	27	2	1	3	1	1	0	2	2	1
21	1	0	1	1	23	1	1	1	1	1	1	0	1	1	310

Table 8-11: Summary of disturbance diagnosis based on various FDI approaches

FDI Methods	IDVs Identified	Recognition Rate (%)	Avg Detection Delay (sample)	Avg Diagnosis Delay (sample)
SOM	14	92.56%	45.83	65.67
PCA	10	74.85%	26.83	51.17
NN	11	95.72%	28.17	35.50
Voting	14	94.68%	30.33	40.83
Bayesian	15	94.73%	25.33	28.00
Dempster-Shafer	14	94.06%	28.17	35.30

* best results indicated by shading

The agreements between the agents are measured with Cohen's Kappa statistic as shown in Table 8-12. All three diagnostic agents, the A^d_{PCA}, A^d_{NN}, and A^d_{SOM} agents show high level of agreement in majority of the data analyzed, *i.e.*, $K^p > 0.6$, $\forall A^d_{r,r'},\ r,r' \in (PCA, NN, SOM)$. All disagreements among the FDI agents are resolved through the implemented decision fusion methods.

Table 8-12: Kappa statistic observed among heterogeneous fault classifiers

Classifier I	**Classifier II**	**Kappa K^p Value**	**Standard Error σ_{K^p}**	**Agreement Level**
A^d_{PCA}	A^d_{NN}	0.8308	0.0063	Almost perfect
A^d_{NN}	A^d_{SOM}	0.9016	0.0044	Almost perfect
A^d_{SOM}	A^d_{PCA}	0.7193	0.0073	Substantial

8.6 Case Study 2: Fault diagnosis during distillation-unit startup

In this section, the various decision fusion methodologies are tested with the pilot-scale distillation-unit startup case study described in Section 3.8. All ten DSTs are tested and various decision fusion methods are used to combine the diagnosis results produced from all monitoring and diagnostic agents. In this case study, the Self Organizing Map, Kernel Density Estimator, and Neural-network based FDI agents are used to diagnose process faults. The summary of disturbance diagnosis based on single FDI approach is shown in Table 8-21.

FDI Agents Implemented

Three FDI agents were implemented in this case study to detect and diagnose faults, namely, neural-network agent, A^m_{NN}, self-organizing map agent, A^m_{SOM}, and KDE-based principal components agent, A^m_{KDE}. All associated agents have diagnostic capabilities implemented as diagnostic agents, A^d_r, where $r = [SOM, KDE, NN]$.

1. *Neural-network agent*, A_{NN}^{m}: The A_{NN}^{m} is constructed based on three layers of feedforward neural-network with [15 15 11] nodes on each layer with similar transfer function configurations as earlier case study. The fault data from the ten training DSTs and one normal data (available in Ng and Srinivasan, 2005) are used to form the neural-network training matrixes. The training matrix is then of size 11 columns of binary elements (see Appendix A). All state-variables are autoscaled, and the weights and biases of the neural-network are obtained with scaled conjugate gradient training method.
2. *Self-organizing map agent*, A_{SOM}^{m}: The A_{SOM}^{m} is also constructed with similar training data. The constructed SOM consists of 33×22 map units and is used to track process trajectory during transitions. Abnormal operations are detected based on cluster-sequence analysis and state-signature comparison are used to diagnose process faults.
3. *KDE-based Principal components agent*, A_{KDE}^{m}: The A_{KDE}^{m} for monitoring is constructed with normal operation data only, X^{R}. The KDE models are constructed by constructing non-parametric bounds around the boundary of the normal operating region. Fault identification with A_{KDE}^{m} is based on the KDE model created, $M^{KDE_{F_j}}$ for each class of fault, F_j, $j \in [1, 11]$.

Diagnostic Results and Discussions

The fault diagnosis results of each FDI agent working in isolation is shown in Tables 7-12 to 7-14. In this case study, A_{KDE}^{m} is unable to detect DST05. Most faults are successfully detected and diagnosed by other agents.

Table 8-13: Performance of Neural-Network agent A_{NN}^{d} in distillation unit startup case study

DST	1	2	3	4	5	6	7	8	9	10
1	598	0	0	0	0	0	0	0	0	0
2	0	377	0	0	0	0	0	0	0	0
3	0	0	237	0	0	0	0	0	0	0
4	0	0	0	153	0	0	0	0	7	0
5	0	0	0	0	147	0	0	0	0	0
6	0	0	30	0	0	172	0	0	0	0
7	0	0	0	0	0	0	107	0	0	0
8	0	0	0	0	0	0	0	137	0	0
9	0	0	16	0	0	1	0	2	266	2
10	0	0	0	0	0	0	0	0	0	268

Table 8-14: Performance of Kernel Density Estimation agent A_{KDE}^{d} in distillation unit startup case study

DST	1	2	3	4	5	6	7	8	9	10
1	602	0	6	2	0	0	0	9	92	0
2	0	378	1	1	3	2	1	2	0	0
3	0	0	75	0	17	13	0	1	17	2
4	0	0	2	137	105	58	0	127	113	112
5	0	0	0	3	2	0	0	3	3	3
6	0	0	2	0	0	174	1	1	2	0
7	0	0	1	0	0	3	110	0	90	1
8	0	0	0	6	0	0	0	23	3	2
9	0	0	0	0	0	0	0	0	110	0
10	0	0	0	5	7	0	1	0	37	122

Table 8-15: Performance of Self-organizing Maps agent A_{SOM}^{d} in distillation unit startup case study

DST	1	2	3	4	5	6	7	8	9	10
1	594	0	0	0	0	0	0	0	0	0
2	0	378	0	0	12	0	12	0	12	12
3	0	0	177	6	6	0	0	6	0	0
4	0	0	114	159	114	0	0	0	0	0
5	0	0	0	0	208	0	0	0	0	0
6	0	0	0	0	0	172	0	0	0	0
7	0	0	0	0	0	0	109	0	0	0
8	0	0	127	127	127	0	0	148	0	0
9	0	308	0	0	327	0	309	0	563	309
10	0	0	0	0	0	0	0	0	0	172

The summary of disturbance identification results based on isolated FDI agents to each DST analyzed is shown in Table 8-16. Both A_{NN}^{m} and A_{SOM}^{m} are able to detect and diagnose all ten disturbances introduced. The A_{NN}^{m} shows good recognition rate in

this case study with 87.15% accuracy and an average detection and diagnosis delay of 21.0 samples and 51.4 samples respectively (Table 8-17).

Table 8-16: Performance comparison of each FDI agent

DST #	SOM		KDE		NN	
	Time fault detected	Time fault diagnosed	Time fault detected	Time fault diagnosed	Time fault detected	Time fault diagnosed
1	10	10	2	11	2	3
2	2	13	2	4	2	2
3	365	371	370	398	360	360
4	357	358	357	359	357	357
5	426	426	-	-	426	427
6	354	354	351	354	351	354
7	347	347	346	350	347	349
8	348	470	348	474	348	350
9	2	328	2	2	2	298
10	300	300	302	339	203	203

Table 8-17: Performance evaluation of each FDI classifier

FDI methods	DSTs Identified	Recognition Rate	Avg Detection Delay	Avg Diagnosis Delay
SOM	10	56.22	32.3	78.9
KDE	9	66.83	31.8	52.7
NN	10	87.15	21.0	51.4

The combined FDI results based on various decision fusion strategies are shown in Tables 7-17 to 7-19. By combining the three FDI agents, all decision fusion strategies are able to diagnose all ten disturbances with improvements in speed of fault detection and diagnostic (see summary in Table 8-21). The voting strategy shows highest recognition rate with 98.36% as redundant information are filtered most effectively by seeking agreement from majority of the FDI agents (see Table 8-22). Though being able to classify samples more accurately, voting strategy incurs higher duration for successful fault detection and diagnosis compared to Bayesian and Dempster-Shafer fusion strategy as voting strategy requires a plurality of votes to change the belief of the consolidator agent. Shortest detection and diagnostic delay are observed through Bayesian and Dempster-Shafer combination strategies. For this case

study, both decision fusion strategies show no significant difference in terms of speed of fault detection and identification (see Table 8-22). A minimum improvement of 37.9% (-19.5 samples) in speed of fault identification has been observed for strategies based on the multi-agent approach compared to any solitary application of FDI method (SOM, KDE or NN). Since the Bayesian fusion strategy utilizes a more explicit, class specific information (stored as Confusion Matrix as shown in Eq 8-6) for decision fusion, it is able to achieve a higher recognition rate compared to Dempster-Shafer strategy, which is based on lumping overall performance from a FDI classifier based on its previous predictions.

Table 8-18: Performance of Voting-based decision fusion for distillation-unit startup case study

DST	1	2	3	4	5	6	7	8	9	10
1	600	0	0	0	0	0	0	0	0	0
2	0	378	0	0	1	0	1	0	0	0
3	0	0	177	0	0	0	0	0	0	0
4	0	0	7	157	8	0	0	7	7	0
5	0	0	0	1	150	0	0	1	1	1
6	0	0	1	0	0	172	0	0	0	0
7	0	0	0	0	0	0	109	0	0	0
8	0	0	0	2	0	0	0	141	0	0
9	0	0	0	0	0	0	0	0	348	2
10	0	0	0	0	0	0	0	0	0	172

Table 8-19: Performance of Bayesian-based decision fusion for distillation-unit case study

DST	1	2	3	4	5	6	7	8	9	10
1	600	0	0	0	0	0	0	0	0	0
2	0	377	0	0	0	0	0	0	0	0
3	0	0	237	0	0	0	0	0	0	0
4	0	0	0	153	0	0	0	0	7	0
5	0	0	0	0	147	0	0	0	0	0
6	0	0	30	0	0	172	0	0	0	0
7	0	0	0	0	0	0	107	0	0	0
8	0	0	0	0	0	0	0	137	0	0
9	0	0	16	0	0	0	0	1	266	2
10	0	0	0	0	0	0	0	0	0	268

Table 8-20: Performance of Dempster-Shafer based decision fusion for distillation-unit case study

DST	1	2	3	4	5	6	7	8	9	10
1	602	0	2	1	0	0	0	2	2	0
2	0	378	0	0	1	0	1	0	0	0
3	0	0	237	0	0	0	0	0	0	0
4	0	0	2	155	2	0	0	2	7	0
5	0	0	0	0	208	0	0	0	0	0
6	0	0	31	0	0	173	0	0	1	0
7	0	0	1	0	0	1	110	0	1	1
8	0	0	7	9	7	0	0	148	0	0
9	0	213	0	0	213	0	213	0	561	215
10	0	0	0	0	0	0	0	0	0	268

Table 8-21: Summary of FDI results by various decision fusion strategy

	Voting		Bayesian		Dempster-Shafer	
DST #	**Time fault detected**	**Time fault diagnosed**	**Time fault detected**	**Time fault diagnosed**	**Time fault detected**	**Time fault diagnosed**
1	3	3	2	2	2	3
2	2	2	2	2	2	2
3	365	365	360	360	360	360
4	357	358	357	357	357	357
5	427	427	426	426	426	426
6	351	354	351	354	351	354
7	347	347	346	346	346	347
8	348	350	348	350	348	350
9	2	2	2	3	2	2
10	300	300	203	203	203	203

Table 8-22: Performance of disturbance diagnosis based on heterogeneous FDI approaches

Combination Strategy	DSTs Identified	Recognition Rate	Avg Detection Delay	Avg Diagnosis Delay
Voting	10	98.36	31.3	31.9
Bayesian	10	97.78	20.9	21.5
Dempster-Shafer	10	75.23	20.9	21.6

The agreements among inter-classifiers are quantified through Cohen's Kappa statistic, K^p, (see Table 8-23). All measured K^p are positive ($K^p > 0$) indicating that no systematic disagreement exist between the diagnostic classifiers. Contrary to previous case study, the observed K^p between classifiers A^d_{SOM}, and A^d_{PCA} is observed to be 0.2391 ∓ 0.005, signifying only a fair agreement between them. Nonetheless, the

disagreements between A^d_{SOM} and A^d_{PCA} classifiers have been effectively resolved by the decision fusion methods.

Table 8-23: Kappa statistic observed among heterogeneous fault classifiers

Classifier I	Classifier II	Kappa K^p Value	Standard Error σ_{K^P}	Agreement Level
A^d_{KDE}	A^d_{NN}	0.4495	0.0085	Moderate
A^d_{NN}	A^d_{SOM}	0.6658	0.0090	Substantial
A^d_{SOM}	A^d_{KDE}	0.2391	0.0053	Fair

8.7 Summary

Three popular methods for combining heterogeneous diagnostic classifiers, namely, voting, Bayesian, and Dempster-Shafer techniques are studied and analyzed in this chapter. Significant improvement in recognition rate has been observed when decision fusion techniques are applied, in which total recognition rate increases from 87.15% to 98.36% is observed for distillation column startup case study. Also, application of decision fusion techniques (particularly Bayesian and Dempster-Shafer) shows that combination of heterogeneous classifiers improve overall system performance by reducing both fault detection and diagnosis time, with improvements of 21.1% based on Bayesian fusion for the Tennessee Eastman Challenge problem, and a minimum improvement of 37.9% for the distillation unit startup case study.

In general, methods that utilize additional information of a classifier, *i.e.*, Bayesian and Dempster-Shafer methods show better performance. Bayesian combination strategy uses class specific historical information for combining heterogeneous classifiers and has been found to perform better than Dempster-Shafer and Voting methods in terms of speed of fault detection and diagnosis. Dempster-Shafer method uses less precise information by quantifying only the recognition and substitution rate from a diagnostic classifier. The performance of Dempster-Shafer method can be

improved by incorporating class-specific information as for the Bayesian approach, *i.e.*, utilizing the Confusion Matrix to generate basic probability assignment. Though the Voting method performs poorly compared to Bayesian and Dempster-Shafer method in term of speed of fault detection and diagnosis, voting method shows highest recognition rate among the three decision fusion methods as the plurality voting implemented rejects wrong predictions by seeking agreement from a plurality of classifiers. Nevertheless, its average speed of fault diagnosis has been improved considerably compared to application based on any single FDI approach. In addition, all three decision fusion algorithms are able to improve fault detection and diagnosis performance over single FDI method under conflicting scenarios, as shown in the distillation-unit startup case study when agreement level among FDI agents is low (*i.e.*, when $K^p = 0.24$).

Nomenclature

Indices

i , j fault/ class ID

r classifier

Parameters

J total number of fault classes known to a fault classifier

R total number of fault classifiers used

N total number of samples in the training data

Variables

σ_{K^P} Standard error for Cohen's Kappa statistic for large sample

Λ the superset containing all of the known fault classes A^d a diagnostic agent

A^m a monitoring agent

C_j the j^{th} class

CM^r confusion matrix for rth classifier

$E(x)$ combined prediction results for sample x

$e(x_i)$ classification results for sample x_i

m basic probability assignment

$\emptyset$ an empty set

$P(\text{A}|\text{B})$ conditional probability of event A occurring based on evidence B

P_E combined probability for the occurrence of an event

F_V^{opt} final prediction results obtained through Voting

F_{DS}^{opt} final prediction results obtained through Dempster-Shafer

F_B^{opt} final prediction results obtained through Bayesian combination

s_j similarity between a sample x when compared to j^{th} class

K^p the Kappa statistic between two classifiers

x^R the reference data used for training of models

p_c proportion of agreement by chance

p_o proportion of agreement observed

x_i a multivariate sample collected at time i

κ_r the r^{th} classifier

Chapter 9 Summary and Recommendations for Future Work

9.1 Research Summary

Fault diagnosis in the process industries deals with online identification of process states, detection of abnormal process behavior during operations, and diagnosis of possible root causes of the abnormal behavior. Since most plants increasingly emphasize the domain of agile manufacturing to increase their competitiveness globally, conventional steady-state monitoring techniques are no longer adequate. Additionally, recent regulatory changes in the FDA cGMP guidelines for pharmaceutical processes also drive tremendous interest in monitoring and diagnosing faults in batch and fed-batch operations (US FDA, 2004). As noted in Chapter 1, modeling and monitoring of transient operations are difficult and characterized by various challenges, *i.e.*, highly non-linear processes, multivariate multi-scale operations, inadequacy of regulatory control, etc. In this thesis, an attempt was made to design a suitable architecture to monitor and diagnose faults in transient operations. Multiple new fault detection and diagnostic methods were developed and a multi-agent based diagnostic system has been proposed to integrate the strengths of these various FDI methods. The thesis was divided into five major sections covering the various developments.

First, a new self-organizing map-based methodology was proposed for (i) visualizing temporal, high dimensional operating data, and (ii) diagnosing faults in transient and steady-state operations. The proposed methodology successfully modeled transient operations with a SOM. The SOM is able to represent any state of a process.

Further clustering allows it to accommodate process noise and run-to-run variations. Process measurements recorded in real-time can be SOM projected and the online data abstracted into a univariate state-signature, generated based on the sequence observed in the cluster space after SOM projection of the online data. Different transitions are usually uniquely represented on the SOM, as well as in the signature space. So abnormality during operations are observed as deviations from normal state-sequences. A syntactic pattern recognition approach for disturbance identification was also proposed that compares the similarity between the online state-signature and the state-signature of known disturbances stored in a knowledge-base.

Second, an Adjoined Dynamic Principal Components Analysis (ADPCA) approach was proposed that overcomes the drawbacks encountered in conventional PCA-based monitoring of transient operations. Single PCA models suffer from extensive Type-I and Type-II errors when a process is in a transient state. To overcome this, multiple overlapping PCA models were proposed. The proposed method enforces continuity in data modeling by allowing neighboring models to overlap with each other, and hence avoids discontinuity in modeling transient types of operations. A strategy to select the optimal PCA model in real-time for monitoring was also proposed in accordance with the ADPCA method.

Third, a pattern recognition approach for fault diagnosis was proposed based on a two-layer architecture, namely, principal components analysis and kernel density estimation. Conventional pattern recognition approach for PCA applications have been overly dependent on Hotelling's T^2 statistic. By substituting the statistic with a non-parametric technique (specifically KDE), the boundaries separating the fault classes can be improved and result in higher accuracy of fault classification. Since KDE

operates on a two-dimensional platform, the proposed index is formulated as multivariable classification through a binomial combination of bi-variate KDE models.

Given these extensive developments in fault diagnosis methodologies for transient operations, the various FDI approaches can be integrated to further improve the diagnostic efficiency. In the fourth part of this thesis (Chapter 7), a multi-agent architecture called CAMEO, Collaborative Agents for Managing Efficient Operations, was proposed to facilitate the integrated management of transient processes. In the proposed architecture, the whole operation is managed through a society of interacting agents, where each agent specializes in a certain area and addresses a special operational issue within the plant. The efficacy of the multi-agent approach was illustrated through combination of heterogeneous fault classifiers. The rationale for such an integration is based on the precept that the strengths of various approaches can be combined while their shortcomings overcome through collaboration.

When heterogeneous fault classifiers are combined, there is a need to resolve possible conflicts among the diagnostic classifiers. Various means of integrating decisions from the multi-agent architecture was thoroughly studied and compared in the final part of this thesis (Chapter 8). Three approaches of integrating decisions, namely voting, Bayesian theory, and Dempster-Shafer theory were studied. With the decision fusion algorithm in place, the fault-diagnosis module of the multi-agent architecture can be scaled-up with ease through addition of monitoring and diagnostic agents.

9.2 Future Recommendations

This thesis thus offers a new multi-agent formalism for monitoring and diagnosis during transient operations. Next, some suggestions for future research are recommended.

9.2.1 Improvement to Diagnostic Methods

The proposed multi-agent architecture serves as a foundation for integrating FDI classifiers. In general, the performance of the multi-agent fault classifier increases with addition of new FDI methods. The gain in FDI performance is more obvious when classifiers utilizing distinct features of the process are added. Throughout this thesis, the main emphasis has been placed on data-based FDI approaches for disturbance isolation. The diagnostic algorithms used mainly focus on the area of pattern matching or similarity search with a fault database. These approaches of diagnosing faults inherit the common drawbacks of all data-driven approaches, such as inability to diagnose novel faults, and inadequate performance in situations where a single fault maps to many patterns in the feature space. To overcome these shortcomings, methods based on other reasoning methodologies, e.g.: signed-digraphs that reason based on causality, expert systems that solve uncertainties based on human experiences, and first-principle models, can be integrated.

Also, there should be a continuous drive to develop better diagnostic methods for FDI of transient operations. A multi-agent architecture consisting of poor diagnostic methods cannot be expected to do miracles.

9.2.2 Transition Automation and Fault Tolerant Control

Throughout this thesis, the focus has been on the domain of fault detection and identification. There is however a strong motivation for achieving higher level of transition control through automation of transition operations. The use of data-driven approaches to automate transition has been previously proposed by Özkan *et al.* (2003), and by Banerjee and Arkun (1998). The proposed modeling and transition representation methods in this thesis, *i.e.*, self-organizing map, or kernel density estimator techniques, can be well extended to drive a transition automatically based on

schemes such as Model Predictive Control. As noted in Chapter 3 & 4, self-organizing maps give a robust representation of process operations and the transition trajectory produced on SOM allows different states of transitions to be quantified and compared. Fault tolerant control can also be integrated into the proposed multi-agent framework to improve the robustness of an operation. Such integration allows the plant to continue to operate at least at a suboptimal level in the event of a fault, instead of failing abruptly.

9.2.3 Integration of Multi-agent System with Planning Mechanism

Multi-agent system can be considered more robust as agent-based method has been shown to be more tolerant to failure (Wooldridge, 2002). Since each agent has been made autonomous, failure in some agents (under the condition that there exist other agents to undertake the task) will not cause an abrupt failure but rather a graceful degradation in system performance. Agent researchers have also reported that agents can be programmed to change their behavior, and attend to unforeseen circumstances through means-end analysis or belief-desire-intention models (Jennings and Wooldridge 1996; Wooldridge, 1997; Fatima *et al.*, 2004). While an individual agent can learn to change its performance, a society of agents can evolve to find a parato optimal configuration for a certain task and environment. This is a form of emergent behavior which would make multi-agent systems an attractive alternative to current approaches in the domain of fault detection and diagnosis in the process industries.

9.2.4 Integration with Other Plant Operations

The concept of integrating fault diagnosis with other part of plant operations is not new. It was first conceptualized by Mylaraswamy (1996) who stressed the need to integrate fault diagnosis with supervisory control, operator notification system, and fault rectification system. Since the concept of integrating plant operations is deemed

very important, it is reiterated here. A graphical illustration of a possible integrated framework for process operations is shown in Figure 9-1. It is believed that a unified framework which could accurately segregate the plant operations into modes and transitions, and intelligently optimize both areas would be handy for engineers and operators. Some developments have been achieved in this area through the years (see Srinivasan *et al.*, 2004; Srinivasan *et al.*, 2005a,b,c; and Ng *et al.*, 2006), but further developments in this area are needed before these methods can be put into practice.

There is also a growing consensus among both manufacturers and researchers that productivity of plant engineers and operators can be improved by better management of information flow within a company. It is believed that integrating additional information such as plant scheduling information, simulation studies for change of operating parameters, data mining from historical operations, etc, would be useful to plant engineers. However, such integration is generally difficult as plant engineers use different tools for different purposes. For instance, it is not difficult to locate a chemical plant which uses software from SAP for scheduling, software from ASPEN for simulation studies, systems from Emerson or Yokogawa for control and automation, and a dozen other systems provided by different vendors for a variety of other functions. It is hence crucial to develop standards for allowing distributed information to be collected, integrated, and unified. Defining such standards for plant operations, or OPC for real-time monitoring and control purposes (Liu *et al.*, 2005) is also considered a practicable area worth studying.

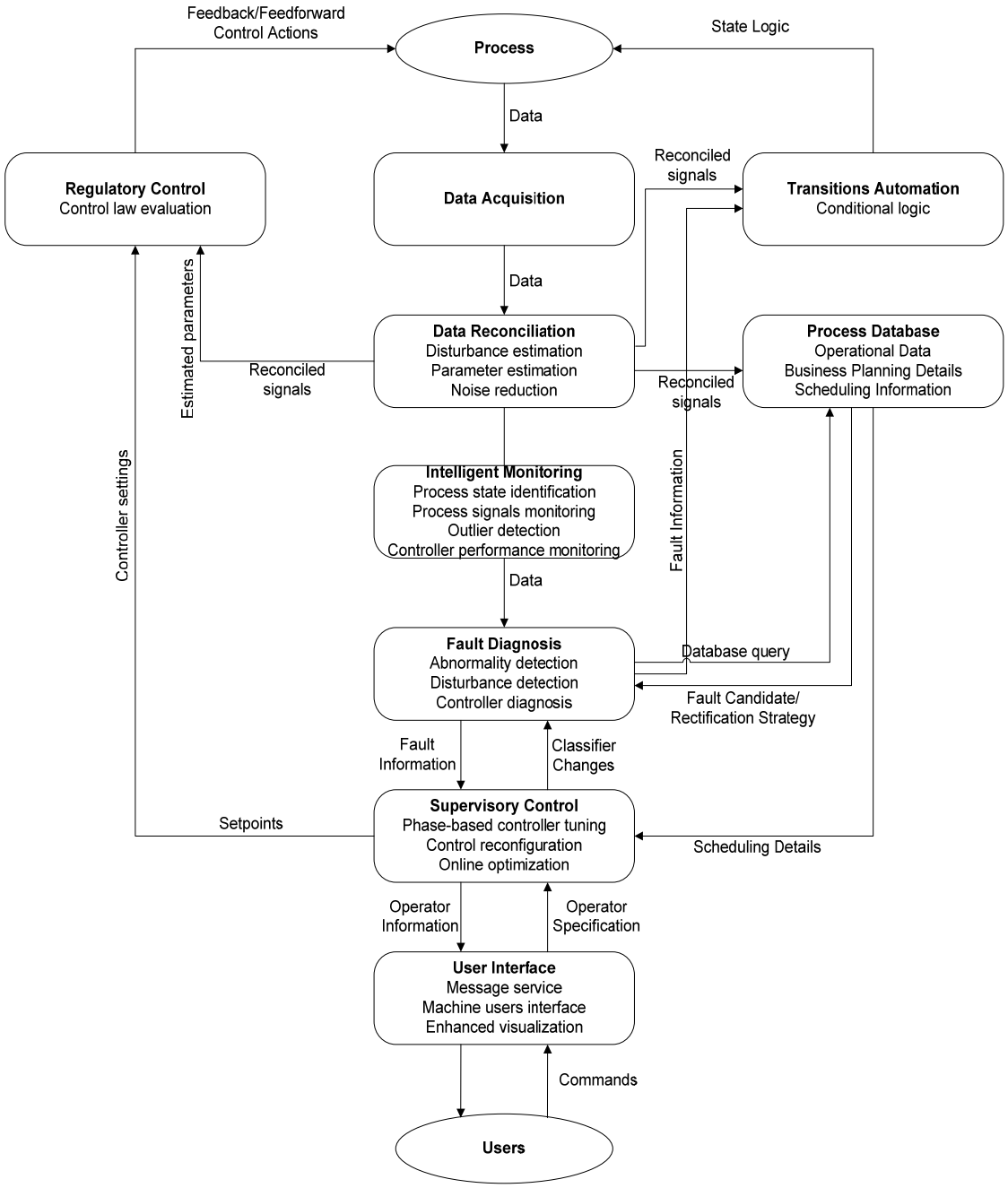


Figure 9-1: Framework for integrating diagnosis with other parts of process operations

Bibliography

Abonyi, J., Nemeth, S., Vincze., C., Arva, P., (2003). Process analysis and product quality estimation by self-organizing maps with an application to polyethylene production. *Computers in Industry 52*, pp. 221-234.

Al-Ghoneim, K., and Kumar, B.V.K.V., (1998). Unified decision combination framework, *Pattern Recognition 31(12)*, pp. 2077-2089.

Albazzaz, H., Wang, X.Z., Marhoon, F., (2005). Multidimensional visualization for process historical data analysis: a comparative study with multivariate statistical process control, *Journal of Process Control 15*, pp. 285-294.

Allsopp, D.N., Beautement, P., Kirton, M., Bradshaw, J.M., Suri, N., Durfee, E.H., Knoblock, C.A., Tate, A., Thompson, C.W., (2002). Coalition agents experiment: Multiagent cooperation in international coalitions, *IEEE Intelligent Systems, Coalition Operations, May/June 2002 Issue*, pp. 26-35.

Andrews, D., (1972). Plots of high dimensional data, *Biometrics 28*, pp. 125-136.

Azimzadeh, F., Galan, O., Romagnoli, J.A., (2001). On-line optimal trajectory control for a fermentation process using multi-linear models, *Computers & Chemical Engineering 25*, pp. 15-26.

Bajpai, R., and Reuss, M., (1980). A mechanistic model for penicillin production, *Journal of Chemical Technology and Biotechnology 30*, pp. 330-344.

Banerjee, A., and Arkun, Y., (1998). Model predictive control of plant transitions using a new identification technique for interpolating nonlinear models, *Journal of Process Control*, 8(5-6), pp. 441-457.

Basir, O., and Yuan, X., (2005). Engine fault diagnosis based on multi-sensor information fusion using Dempster-Shafer evidence theory, *Information Fusion* (In Press).

Benouhiba, T., and Nigro, J-M., (2006). An evidential cooperative multi-agent system, *Expert Systems with Applications 30*, pp. 255-264.

Bezdek, J. C., (1974). Numerical taxonomy with fuzzy sets, *Journal of Mathematical Biology 1*, pp. 57-71.

Bhagwat, A., Srinivasan, R., Krishnaswamy, P.R., (2003a). Fault detection during process transitions: a model-based approach, *Chemical Engineering Science 58*, pp. 309–325.

Bhagwat, A., Srinivasan, R., Krishnaswamy, P.R., (2003b). Multi-linear model-based fault detection during process transitions, *Chemical Engineering Science 58*, pp. 1649–1670.

Birol, G., Ündey, C., Çinar, A., (2002). A modular simulation package for fed-batch fermentation: penicillin production, *Computers and Chemical Engineering 26*, pp. 1553-1565.

Böling, J.M., Häggblom, K.E., Nyström, R.H., (2004). Multivariable uncertainty estimation based-on multi-model output matching, *Journal of Process Control 14*, pp. 293-304.

Bowman, A.W., (1984). An alternative method of cross-validation for the smoothing of density estimates, *Biometrika 71*, pp. 353-360.

Brams, S.J., and Fishburn, P.C., (1983). Approval voting, *Birkhauser*, Boston.

Bureau of Labor Statistics (1998). Occupational injuries and illness in The United States by industry, *Government Printing Office*, Washington, DC.

Chan, A.D.C., Englehart, K.B., Hudgins, B., Lovely, D.F., (2006). Multiexpert automatic speech recognition using acoustic and myoelectric signals, *IEEE Transactions on Biomedical Engineering 53(4)*, April, pp. 676-685.

Chan, C.W., Jin, H., Cheung, K.C., Zhang, H.Y., (2001). Fault detection of systems with redundant sensors using constrained Kohonen networks, *Auomatica 37*, pp. 1671-1676.

Chen, J., and Liu, K.C., (2002). On-line batch process monitoring using dynamic PCA and dynamic PLS models, *Chemical Engineering Science 57*, pp. 63-75.

Chen, Q., Kruger, U., Meronk, M., Leung, A.Y.T., (2004). Synthesis of T^2 and Q statistics for process monitoring, *Control Engineering Practice 12*, pp. 745-555.

Chen, Q., Wynne, R.J., Goulding, P., Sandoz, D., (2000). The application of principal component analysis and kernel density estimation to enhance process monitoring, *Control Engineering Practice 8*, pp. 531-543.

Chernoff, H., (1973). Using faces to represent points in k-dimensional space graphically, *Journal of American Statistical Association 68*, pp. 361-368.

Cheung, J.T., and Stephanopoulos, G., (1990). Representation of process trends part I. A formal representation framework, *Computers and Chemical Engineering 14* (4-5), pp. 495-510.

Chiang, L.H. and Braatz, R.D., (2003). Process monitoring using causal map and multivariate statistics: fault detection and identification, *Chemometrics and Intelligent Laboratory Systems 65*, pp. 159 – 178.

Chiang, L.H., Russell, E.L., Braatz, R.D., (2001). Fault detection and diagnosis in industrial systems, *Springer-Verlag London Limited*, Briton.

Cho, K-J., Ahn, S-J., and Chung, J-W., (2003). A study on the classified model and the agent collaboration model for network configuration fault management, *Knowledge-Based Systems 16*, pp. 177-190.

Clemen, R.T., (1989). Combining forecasts: A review and annotated bibliography, *International Journal of Forecasting 5*, pp. 559-583.

Cohen, J., (1960). A coefficient of agreement for nominal scales, *Educational and Psychological Measurement 20*, pp. 37-46.

Cohen, J., (1968). Weighted Kappa: Nominal scale agreement with provision for scaled disagreement or partial credit, *Psychological Bulletin 70(4)*, pp. 213-220.

Craig, P., Kennedy, J., Cumming, A., (2005). Animated interval scatter-plot views for the exploratory analysis of large-scale microarray time-course data, *Information Visualization 4*, pp. 149-163.

Dash, S., Rengaswamy, R., Venkatasubramanian, V., (2003). Fuzzy-logic based trend classification for fault diagnosis of chemical processes, Computers *and Chemical Engineering 27*, pp. 347-362.

Dempster, A.P., (1968). A generalization of Bayesian inference, *Journal of the Royal Statistical Society, Series B(30)*, pp. 205-247.

Deventer, J.S.J.V., Moolman, D.W., Aldrich, C., (1996). Visualization of plant disturbances using self-organizing maps, *Computers and Chemical Engineering 20*, pp.1095-1100.

Doan, X-T, and Srinivasan, R., (2007). Online monitoring of multi-phase batch processes using phase-based multivariate statistical process control, *Computers and Chemical Engineering*, doi:10.1010/j.compchemeng.2007.05.010.

Downs, J.J., and Vogel., E.F., (1993). A plant-wide industrial process control problem, *Computers and Chemical Engineering 17(3)*, 245-255.

Doymaz, F., Chen, J., Romagnoli, J.A., Palazoglu, A., (2001). A robust strategy for real-time process monitoring, *Journal of Process Control 11*, pp. 343-359.

DSTO (Defence Science and Technology Organization), (1994). Data Fusion Special Interest Group, Data fusion lexicon, *Department of Defence*, Australia, pp., 21 September.

Dunia, R., and Qin, J., (1998). Subspace approach to multidimensional fault identification and reconstruction, *AIChE Journal* 44(8), pp.1813-1831.

Engel, J., Herrmann, E., Gasser, T., (1994). An iterative bandwidth selector for kernel estimation of densities and their derivatives, *Journal of Nonparametric statistics 4*, pp. 21-34.

Epanechnikov, V.K., (1969). Non-parametric estimation of a multivariate probability density, *Theory of Probability and its Application 14*, pp. 153-158.

Fabro, J.A., Arruda, L.V.R., Neves Jr, F., (2005). Startup of a distillation column using intelligent control techniques, *Computers & Chemical Engineering 30*, pp. 309-320.

Fatima, S., Wooldridge, M., Jennings, N.R., (2004). Optimal negotiation of multiple issues in incomplete information settings, Proceedings of the Third International Joint Conference on Autonomous agents and Multiagent Systems, AAMAS, pp. 869-903, New York, USA.

Fix, E., and Hodges, J.L., Jr., (1951). Nonparametric Discrimination: Consistency Properties, Report Number 4, *USAF School of Aviation Medicine*, Randolph Field, Texas.

Foggia, P., Sansone, C., Tortorella, F., Vento, M., (1999). Multiclassification : reject criteria for the Bayesian combiner, *Pattern Recognition 32*, pp. 1435-1447.

Fourie, S.H., and de Vaal, P., (2000). Advanced process monitoring using an on-line non-linear multiscale principal component analysis methodology, *Computers & Chemical Engineering 24*, 755-760.

García-Flores, R., Wang, X.Z., Goltz, G.E., (2000). Agent-based information flow for process industries' supply chain modeling, *Computers and Chemical Engineering 24*, pp. 1135-1141.

Giarratano, J., and Riley, G., (1998). Expert Systems: Principles and Programming, *PWS Publishing Company*, Boston, MA, USA.

Gollmer, K., Posten, C., (1996). Supervision of bioprocess using a dynamic time warping algorithm, *Control Engineering Practice* 4(9), pp. 1287 – 1295.

Goulding, P.R., Lennox, B., Chen, Q., Sandoz, D.J., (2000). Robust inferential control using kernel density methods, Computers & Chemical Engineering 24, pp. 835-840.

Gusfield, D., (1997). Algorithms on strings, trees, and sequences: Computer Science and Computational Biology, *Cambridge University Press*, Cambridge, England.

Haddad, A.H., (2006). Probabilistic systems and random signals, *Pearson Education Inc*, Upper Saddle River, N.J., USA.

Hagan, M.T., Demuth, H.B., Beale, M., (1996). Neural-network design, *PWS Publishing Company*, Boston, MA.

Hall, D.L., (1992). Mathematical techniques in multi-sensor data fusion, *Artech House, Inc*, Norwood, MA.

Hathaway, R. J., and Bezdek, J.C., (1988). Recent convergence for the fuzzy c-means clustering algorithm, *Journal of Classification 5*, pp. 237-247.

Himmelblau, D.M., (1978). Fault detection and diagnosis in chemical and petrochemical processes, *Elsevier Scientific Publishing Company*, Amsterdam, The Netherlands.

Honda, H., and Kobayashi, T., (2000). Fuzzy control of bioprocess, *Journal of Bioscience and Bioengineering 89*(5), pp. 401-408.

Howell, D.C., (2001). Statistical methods for psychology (Fifth edition), *Duxbury, Thomson Learning Academic Resource Centre*, Pacific Grove, C.A., USA.

Hwang, D.H., and Han, C, (1999). Real-time monitoring for a process with multiple operating modes, *Control Engineering Practice 7*, pp. 891-902.

Inselberg, A., (1985). Intelligent instrumentation and process control, *In Proceedings of the 2nd Conference on Artificial Intelligence*, IEEE Computer Society Press, Washington, pp. 302-307.

Inselberg, A., Chomut, T., Reif, M., (1987). Convexity algorithms in parallel coordinates, *Journal of the Association for Computing Machinery 34(4)*, pp. 765-801.

Inselberg, A., (2002). Visualization and data mining of high-dimensional data, *Chemometrics and Intelligent Laboratory Systems 60*, pp. 147-159.

Jackson, J.E., (1991). A user's guide to principal components, *John Wiley & Sons*, New York.

Jackson, J.E., and Mudholkar, G.S., (1979). Control procedures for residuals associated with principal component analysis, *Technometrics 21 (3)*, pp. 341-349.

Jämsä-Jounela, S.L., Vermasvuori, M., Endén, P., Haavisto, S., (2003). A process monitoring system based on the Kohonen self-organizing maps, *Control Engineering Practice 11*, pp. 83-92.

Jennings, N., and Wooldridge, M., (1996). Software agents, *IEE Review January 42(1)*, pp. 17-20.

Jokinen, P.A., (1994). Visualization of multivariate processes using principal component analysis and nonlinear inverse modeling, *Decision Support Systems 11*, pp. 53-65.

Julka, N., Srinivasan, R., Karimi, I., (2002a). Agent-based supply chain management - 1: framework, *Computers and Chemical Engineering 26*, pp. 1755-1769.

Julka, N, Karimi, I., Srinivasan, R., (2002b). Agent-based supply chain management - 2: a refinery application, *Computers and Chemical Engineering 26*, pp. 1771-1781.

Kapil, G.G., Mehra, S., Gomes, J., (2005). On-line adaptation of neural networks for bioprocess control, *Computers and Chemical Engineering 29*, pp. 1047-1057.

Kaski, S., Kangas, J., Kohonen, T., (1998). Bibliography of self-organizing map (SOM) papers: 1981-1997, *Neural Computing Surveys* 1, pp. 102-350.

Kassidas, A., MacGregor, J.F., Taylor, P.A., (1998a). Synchronization of batch trajectories using dynamic time warping, *American Institute of Chemical Engineers Journal 44*, No. 4, pp. 864-875.

Kassidas, A., Taylor, P.A., MacGregor, J.F., (1998b). Off-line diagnosis of deterministic faults in continuous dynamic multivariable processes using speech recognition methods, *Journal of Process Control* 8(5-6), 1998, pp. 381-393.

Kavuri, S.N., and Venkatasubramanian, V., (1993). Representing bounded fault classes using neural networks with ellipsoidal functions, *Computers and Chemical Engineering 17(2)*, pp. 139-163.

Kepner, J., and Ahalt, S., (2004). MatlabMPI, *Journal of Parallel and Distributed Computing*, *8(64),* pp. 997-1005.

Kohonen, T., (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics* 43, pp. 59-69.

Kohonen, T., (1993). Things you haven't heard about the self-organizing map, *IEEE International Conference,* Neural Networks, pp. 1147 – 1156.

Kohonen, T., (2000). Self-Organizing Maps, *Springer Series in Information Sciences*, Springer, Berlin, Germany.

Kolehmainen, M., Rönkkö, P., Raatikainen, O., (2003). Monitoring of yeast fermentation by ion mobility spectrometry measurement and data visualization with self-organizing maps, *Analytica Chemica Acta 484*, pp. 93 – 100.

Kosanovich, K.A., Charboneau, J.G., and Piovoso, M.J., (1997). Operating regime-based controller stategy for multi-product processes, *Journal of Process Control 7(1)*, pp 43-56.

Kourti, T., (2002). Process analysis and abnormal situation detection: From theory to practice, *IEEE Control Systems 22(5)*, pp. 10-25.

Krzanowski, W.J., (1979). Between-groups comparison of principal components, *Journal of the American Statistical Association 74(367),* pp.703-707.

Ku, W., Storer, R.H., Georgakis, C., (1995). Disturbance detection and isolation by dynamic principal component analysis, *Chemometrics and Intelligent Laboratory Systems 30*, pp. 179-196.

Lam, L., and Suen., C.Y., (1997). Application of majority voting to pattern recognition: An analysis of the behavior and performance, *IEEE Trans. Systems Man Cybernet 27(5)*, pp. 553-567.

Landis, J.R., and Koch, G.G., (1977). The measurement of observer agreement for categorical data, *Biometrics 33(1)*, pp. 159-174.

Laser, M., (2000). Recent safety and environmental legislation. *Trans IchemE 78(B)*, pp. 419-422.

Lee, J.L., Yoo, C.K., Lee, I.B., (2004a). Statistical process monitoring with independent component analysis, *Journal of Process Control 14,* pp. 467-485.

Lee, J.M., Yoo, C.K., Lee, I.B., (2004b). Enhanced process monitoring of fed-batch penicillin cultivation using time-varying and multivariate statistical analysis, *Journal of Biotechnology 110*, pp. 119-136.

Lim, C.P., and Harrison, R., (2003). Online pattern classification with multiple neural-network systems: An experimental study, *IEEE Transactions on Systems, Man, and Cybernatics – Part C: Applications and Reviews 33(2)*, May, pp. 235-247.

Lin, X., Yacoub, S., Burns, J. Simske, S., (2003). Performance analysis of pattern classifier combinatino by plurality voting, *Pattern Recognition Letters 24*, pp. 1959-1969.

Liu, J., Lim., K.W., Ho., W.K., Tan, K.C., Srinivasan, R., (2005). Using the OPC standard for real-time process monitoring and control, *IEEE Software 22(6)*, pp. 54-59.

Ljung, L., and Glad, T., (1994). Modeling of dynamic systems, *Prentice Hall*, Englewood Cliffs.

Lopes, J.A., and Menezes, J.C., (2004). Multivariate monitoring of fermentation processes with non-linear modeling methods, *Analytica Chimica Acta 515*, pp. 101-108.

López-Rubio,E., Muñoz-Pérez, J., Gómez-Ruiz, J.A., (2003). A principle component analysis self-organizing map. *Neural Networks 17(2)*, March, pp. 261-270.

Lu, N., and Gao, F., (2005). Stage-based process analysis and quality prediction for batch processes, *Industrial and Engineering Chemistry Research 44,* 3547-3555.

Lyman, P.R., and Georgakis, C., (1995). Plant-wide control of the Tennessee Eastman problem, *Computers and Chemical Engineering 19(3)*, pp. 321-331.

MacGregor, J.F. and Kourti, T., (1995). Statistical process control of multivariate processes, *Control Engineering Practice 3 (3*), pp. 403 – 414.

Mandenius, C-F., Hagman, A., Dunås, F., Sundgren, H., Lundström, I., (1998). A multisensor array for visualizing continuous state transitions in biopharmaceutical processes using principal component analysis, *Biosensors & Bioelectrics 13(2)*, pp. 193-199.

Mangina, E.E., McArthur, S.D.J., McDonald, J.R., Moyes, A., (2001). A multi agent system for monitoring industrial gas turbine start-up sequences*, IEEE Transactions on Power systems 16(3)*, pp. 396-401.

Martin, E.B., and Morris, A.J., (1996). Non-parametric confidence bounds for process performance monitoring charts*, Journal of Process Control 6*, pp. 349-358.

Maturana, F.P., Tichý, P., Šlechta, P., Discenzo, F., Staron, R.J., Hall., K., (2004). Distributed multi-agent architecture for automation systems, *Expert Systems with Applications 26*, pp. 49-56.

McArthur, S.D.J., Strachan, S.M., Jahn, G., (2004). The design of a multi-agent transformer condition monitoring system, *IEEE Transactions on Power Systems 19(4)*, November, pp. 1845-1852.

McGraw-Hill Economics (1985). Survey of investment in employee safety and health, *McGraw-Hill Publishing Company*, New York.

Muthuswamy, K. and Srinivasan, R., (2003). Phase-based supervisory control for fermentation process development*, Journal of Process Control 13*, pp. 367-382.

Mylaraswamy, D., (1996). Dkit: A blackboard-based, distributed, multi-expert environment for abnormal situation management, *PhD Thesis,* Purdue University*,* Department of Chemical Engineering, West Lafayette, USA.

Mylaraswamy, D., and Venkatasubramanian, V., (1997). A hybrid framework for large scale process fault diagnosis, *Computers and Chemical Engineering 21*, pp. 935-940.

National Safety Council (1999). Injury facts 1999 edition, *National Safety Council*, Chicago, USA.

Needleman, S.B., and Wunsch, C.D., (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology 48*, pp. 443-453.

Ng, Y. S. and R. Srinivasan, (2004a). Monitoring of Distillation Column Operation through Self-organizing Map, Presented in *7th International Symposium on Dynamics and Control of Process System (DYCOPS),* Massachusetts, USA, July 5 – 7, 2004.

Ng, Y.S, and Srinivasan, R., (2004b). Transitions in the Process Industries: Opportunities and Prospective Solution, *IEEE International Symposium on Intelligent Control*, (Invited Session), Taipei, Taiwan.

Ng, Y.S., and Srinivasan, R., (2005). Distillation unit case study homepage, iACE-Laboratory, Singapore, http://www.iace.eng.nus.edu.sg/research/Distillation_column/index.htm, National University of Singapore.

Ng, Y.S., Yu, W., Srinivasan, R., (2006). Transition classification and performance analysis: A study on industrial hydro-cracker, *International Conference on Industrial Technology ICIT 2006*, Dec 15-17, Mumbai, India, (submitted).

Nimmo, I., (1995). Adequately address abnormal operations, *Chemical Engineering Progress*, September 1995.

Nomikos, P., and MacGregor, J.F., (1994). Monitoring batch processes using multiway principal component analysis, *AIChE Journal 40*, pp. 1361-1375.

Nomikos, P., and MacGregor, J.F., (1995). Multivariate SPC charts for monitoring batch processes, *Technometrics 37(1*), pp. 41-59.

Özkan, L., Kothare, M.V., Georgakis, C., (2003). Control of a solution copolymerization reactor using multi-model predictive control, *Chemical Engineering Science 58*, pp. 1207-1221.

Özyurt, B., and Kandel, A., (1996). A hybrid hierarchical neural network-fuzzy expert system approach to chemical process fault diagnosis, *Fuzzy Sets and Systems 83*, pp. 11-25.

Palma, F.D., and Magni, L, (2004). A multi-model structure for model predictive control, *Annual Reviews in Control 28*, pp. 47-52.

Qin, S. J., (2003). Statistical process monitoring: basics and beyond, *Journal of Chemometrics 17*, pp. 480-502.

Rahman, A.F.R., and Fairhurst, M.C., (1999). Enhancing multiple expert decision combination strategies through exploitation of a priori information sources, *IEE Proc-Vis. Image Signal Processing 146(1),* February, pp.40-49.

Raich, A., and Çinar, A., (1997). Diagnosis of process disturbances by statistical distance and angle measures, *Computers and Chemical Engineering 21*, pp. 661-673.

Raich, A.C., and Çinar, A., (1996). A statistical process monitoring and disturbance diagnosis in multivariate continuous processes, *AIChE Journal 42*, pp. 995-1009.

Rännar S., MacGregor, J.F., Wold, S., (1998). Adaptive batch monitoring using hierarchical PCA, *Chemometrics and Intelligent Laboratory Systems 41*, pp. 73-81.

Rauwendaal, C., (1993). SPC-statistical process control in extrusion, *Hanser Publishers*, Munich; New York.

Rengaswamy, R., and Venkatasubramanian, V., (2000). A fast training neural network and its updation for incipient fault detection and diagnosis. *Computers and Chemical Engineering 24 (27)*, pp. 431-437.

Ringuest, J.L., and Tang, K., (1987). Simple rules for combining forecasts: some empirical results, *Socio-Economic Planning Sciences 21(4)*, pp. 239-243.

Rosenblatt, M., (1956). Remarks on some nonparametric estimates of a density function, *Annals of Mathematics Statistics 27*, pp. 832-837.

Rudemo, M., (1982). Empirical choice of histograms and kernel density estimators, *Scandinavian Journal of Statistics 9*, pp. 65-78.

Saari, D.G., (1994). Geometry of voting, volume 3 of Studies in Economic Theory, *Springer-Verlag*, New York.

Scott, D.W., (1979). On optimal and data-based histograms, *Biometrika 66*, pp. 605-610.

Scott, D.W., Tapia, R.A., Thompson, J.R., (1977). Kernel density-estimation revisited, *Nonlinear Analysis, Theory, Methods and Applications 1*, pp. 339-372.

Seber, G.A.F., (2004). Multivariate observation, *Wiley-Interscience*, Hoboken, N.J.

Shafer, G, (1990). Perspectives on the theory and practice of belief functions, *International Journal of Approximate Reasoning 3*, pp. 1-40.

Shafer, G., (1976). A Mathematical Theory of Evidence, *Princeton University Press*.

Sheather, S.J., and Jones, M.C., (1991). A reliable data-based bandwidth-selection method for kernel density estimation, *Journal of the Royal Statistical Society series B(53)* , pp. 683-690.

Sheremetov, L.B., Contreras, M., Valencia, C., (2004). Intelligent multi-agent support for the contingency management system, *Expert Systems with Applications 26*, pp. 57-71.

Shi, D., and Tsung, F., (2003). Modeling and diagnosis of feedback-controlled processes using dynamic PCA and neural networks, *International Journal of Production Research 41(2)*, pp. 365-379.

Silverman, B.W., (1978). Density Ratios, Empirical Likelihood and Cot Death, *Applied Statistics 27*, pp. 26-33.

Singhal, A., and Seborg, D.E., (2002). Pattern matching in historical batch data using PCA, *IEEE control Systems Magazine*, October Issue, pp. 53-63.

Smets, P., and Kennes, R., (1994). The transferable belief model, *Artificial Intelligence 66*, pp. 191-243.

Smith, T.F., and Waterman, M.S., (1981). Identification of common molecular subsequences, *Journal of Molecular Biology 147*, pp. 195-197.

Srinivasan, R., and Qian, M., (2005). Offline temporal signal comparison using singular points augmented time warping, *Industrial & Engineering Chemistry Research 44*, pp. 4697-4716.

Srinivasan, R., and Qian, M., (2006). Online fault diagnosis and state identification using dynamic locus analysis, *Chemical Engineering Science 61(18)*, pp. 6109-6132.

Srinivasan, R., and Qian, M., (2007). Online temporal signal comparison using singular points augmented time warping, *Industrial & Engineering Chemistry Research 46(13)*, 4531-4548.

Srinivasan, R., and Gopal, S., (2002). Extracting information from high-dimensional operations data using visualization techniques, *AIChe meeting*, Indianapolis, #271c.

Srinivasan, R., Wang, C., Ho, W. K., Lim, K. W., (2004). Dynamic PCA based methodology for clustering process states in agile chemical plants, *Industrial and Engineering Chemistry Research 43*, 2123 – 2139.

Srinivasan, R., Wang. C., Ho, W.K., and Lim, K.W., (2005a). Neural network systems for multi-dimensional temporal pattern classification, *Computers & Chemical Engineering 29*, pp. 965-981.

Srinivasan, R., Wang, C., Ho, W.K., and Lim, K.W., (2005b). Context-based recognition of process states using neural networks, *Chemical Engineering Science 60*, pp. 935-949.

Srinivasan, R., Viswanathan, P., Vedam, H., Nochur, A., (2005c). A framework for managing transitions in chemical plants, *Computers and Chemical Engineering 29*, pp. 305-322.

SSH Communication Security (2006), http://www.ssh.com/

Suen, C.Y., Nadal, C., Mai, T.A., Legault, R., Lam, L., (1990). Recognition of totally unconstrained handwritten numerals based on the concept of multiple experts, Frontiers in Handwriting Recognition, Suen, C.Y., (Eds), *Proc. Int. Workshop on Frontiers in Handwriting Recognition*, Montreal, Canada, Apr 2-3, pp. 131-143.

Sundarraman, A., and Srinivasan, R., (2003). Monitoring transitions in chemical plants using enhanced trend analysis, *Computers & Chemical Engineering 27*, pp. 1455 – 1472.

SVU webpage, (2006). http://www.nus.edu.sg/comcen/svu/hardware_b.html, Computer Centre, National University of Singapore.

Tabuada, P., and Pappas, G.J., (2005). Motion feasibility of multi-agent formations, *IEEE Transactions on Robotics 21(3),* 387-392.

Theil, H., Kidwai, S.A., Yellé, K.A., (1982). Normal distribution plots and the estimation of distribution tails, *Economics Letters 10*, pp. 299-303.

Tso, S.K., Lau, H., Ho, J.K.L., (2000). Coordination and monitoring in an intelligent global manufacturing service system, *Computers In Industries 43*, pp. 83-95.

Ultsch, A., Siemon, H.P., (1990). Kohonen's self organizing feature maps for exploratory data analysis, *Proceedings of International Neural Network Conference (INNC'90)*, Kluwer academic Publishers, Dordrecht, pp. 305-308.

US FDA, (2004). Guidance for Industry, PAT- A framework for innovative pharmaceutical development manufacturing, and quality assurance, U.S. Department of Health and Human Services, Food and Drug Administration, Center for Drug Evaluation and Research (CDER), Center for Veterinary Medicine (CVM), and Office of Regulatory Affairs (ORA).

Vedam, H., and Venkatasubramanian, V., (1999). PCA-SDG based process monitoring and fault diagnosis, *Control Engineering Practice 7*, pp. 903-917.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N., (2003a). A review of process fault detection and diagnosis Part I: Quantitative model-based methods, *Computers and Chemical Engineering 27*, pp. 293 – 311.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., (2003b). A review of process fault detection and diagnosis Part II: Qualitative models and search strategies, *Computers and Chemical Engineering 27*, pp. 313 – 326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., Yin, K., (2003c). A review of process fault detection and diagnosis Part III: Process history based methods, *Computers and Chemical Engineering 27*, pp. 327 – 346.

Vesanto, J., (2002). Data exploration process based on the self-organizing map, phD Thesis, Helsinki University of Technology, Department of Computer Science & Engineering, Finland.

Viera, A.J., and Garrett, J.M., (2005). Understanding interobserver agreement: The Kappa statistic, *Family Medicine 37(5)*, pp. 360-363.

Wahba, G., (1971). A polynomial algorithm for density estimation, *Annals of Mathematics Statistics 42*, pp. 1870-1886.

Wand, M.P., and Jones, M. C., (1994). Kernel Smoothing, *Chapman & Hall*, London.

Wang, X.Z., Medasani, S., Marhoon, F., Albazzaz, H., (2004). Multidimensional visualization of principal component scores for process historical data analysis, *Industrial and Engineering Chemistry Research 43*, pp. 7036-7048.

Watson, G.S., and Leadbetter, M.R., (1963). On the estimation of the probability density I, *Annals of Mathematics Statistics 34*, pp. 480-491.

Wise, B.M., and Gallagher, B., (1996). The process chemometrics approach to process monitoring and fault detection, *Journal of Process Control 6(6)*, pp. 329-349.

Wise, B.M., Ricker, N.L., Veltkamp, D.J., Kowalski, B.R., (1990). A theoretical basis for the use of principal components model for monitoring multivariate processes, *Process Control and Quality 1(1), pp.* 41-51.

Wold, S, (1976). Pattern recognition by means of adjoined principal components models, *Pattern Recognition 8*, pp.127-139.

Wold, S., Kettaneh-Wold, N., and Skagerberg, S., (1989). Non-linear PLS modeling, *Chemmometrics and Intelligent Laboratory Systems 7*, pp. 53-65.

Wooldridge, M., (1997). Agent-based software engineering, *IEE Proceedings on Software Engineering 144*, pp. 26-37.

Wooldridge, M., (2002). An introduction to multiagent systems, *John Wiley & Sons Ltd*, West Sussex, England.

Wooldridge, M., and N.R. Jennings, (1995). Intelligent agents: theory and practice, *The Knowledge Engineering Review10(2),* pp. 115-152.

Xiao, L., Wang, K., Teng, Y., Zhang, J., (2003). Component plane presentation integrated self-organizing map for microarray data analysis. *FEBS Letters 538*. pp. 117-124.

Xu, L., Krzyżak, A., Suen, C.Y., (1992). Methods of combining multiple classifiers and their applications to handwriting recognition, *IEEE Transactions on Systems, Man, and Cybernatics 22(3)*, pp. 418-435.

Yager, R.R., (2001). Dempster-Shafer belief structures with interval valued focal weights, *International Journal of Intelligent Systems 16*, pp. 497-512.

Yang, B.S., and Kim, K.J., (2006). Application of Dempster-Shafer theory in fault diagnosis of induction motors using vibration and current signals, *Mechanical Systems and Signal Processing 20*, pp. 403-420.

Yue, H., and Qin, S.J., (2001). Reconstruction based fault identification using a combined index, *Industrial and Engineering Chemistry Research 40,* pp. 4403-4414.

Zhang, L., Tang, C., Song, Y., Zhang, A., (2003). VizCluster and its application on classifying gene expression data, *Distributed and Parallel Databases 13*, pp. 73-97.

Zheng, M.M., Krishnan, S.M., Tjoa, M.P., (2005). A fusion-based clinical decision support for disease diagnosis from endoscopic images, *Computers in Biology and Medicine 35*, pp. 259-274.

Appendix A: Back-propagation Neural-network

Back-propagation neural-network has been a popular method for context and disturbance identification (Rengaswamy and Venkatasubramanian, 2000; Srinivasan *et al.*, 2005a,b). Neural-networks develop non-linear input-output model and are well suited for analysis of high-dimensional data. The neural-network achieves non-linear class separation through combination of individual neurons. Each neuron has its corresponding activity o_m governing by a transfer function f_m through (Hagan *et al.*, 1996):

$$o_m = f_m\left[(\sum_{i=1}^{r} w_{mi} \cdot p_i) + b\right], \qquad \text{(Eq A-1)}$$

where w_{mi} are the weights from neurons from previous layer, p_i is the inputs from previous layer, and b is a connection bias. During learning, the training data (from normal and abnormal operations), $X \in [X^R, F_1, F_2, ...F_J]$ can be used in conjunction with their class information, C to train a neural-network by adjusting the weights w_{mi} and bias b of all neurons. C is of similar length to the total rows of X and is often represented in binary form:

$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \qquad \text{(Eq A-2)}$$

The learning of neural-networks usually involves iteration to adjust w_{mi} and bias b of all neurons until the sum of squared errors between C and predicted values $\tilde{C}$ drops below a prespecified threshold thr, *i.e.*, $d(C,\tilde{C}) = (\sum_{i=1}^{I}\sum_{j=1}^{J}(C_{ij} - \tilde{C}_{ij})^2) < thr^t$. Different training algorithms such as conjugate gradient algorithms or quasi-Newton

algorithms can be used for such purpose. The fully trained neural-network model M^{NN} can then be used for class prediction of new samples X_i. A threshold thr^C can be placed on the predicted $\tilde{C}$ to define the class acceptance criteria of M^{NN}:

$$\tilde{C}(X_i) = \begin{cases} j, & if \parallel \tilde{C}_{ij} - 1 \parallel < thr^C \\ J+1, & otherwise \end{cases}, \ \forall j \in [1, J]. \qquad \text{(Eq A-3)}$$

The notation $\tilde{C}(X_i) = j$ indicates that the sample X_i is exhibiting abnormal pattern of j^{th} fault class, while $\tilde{C}(X_i) = J+1$ indicates that X_i is rejected by M^{NN} (existence of novel fault).

Appendix B: Multiway-PCA and Dynamic-PCA

PCA has been widely used for monitoring continuous operations (Kourti, 2002). However, there exist some limitations of the PCA approaches when used for monitoring batch processes (Nomikos and MacGregor, 1994). In practice, batch data are usually stored in a three dimensional data matrix. An extension of PCA called Multiway-PCA (Nomikos and MacGregor, 1994) was proposed by for batch data analysis. MPCA organizes the batch data into time-ordered blocks by unfolding the three dimensional array into a large two dimensional matrix before they are decomposed into their corresponding principal components. In general, there exist three different ways that a 3-Dimensional array X can be unfolded into two dimensional matrices Lee *et al.* (2004b):

- Batches x variables at each specific time (time-wise unfolding)
- Variables x time for each specific batch (batch-wise unfolding)
- Batches x times for each specific variables (variable-wise unfolding)

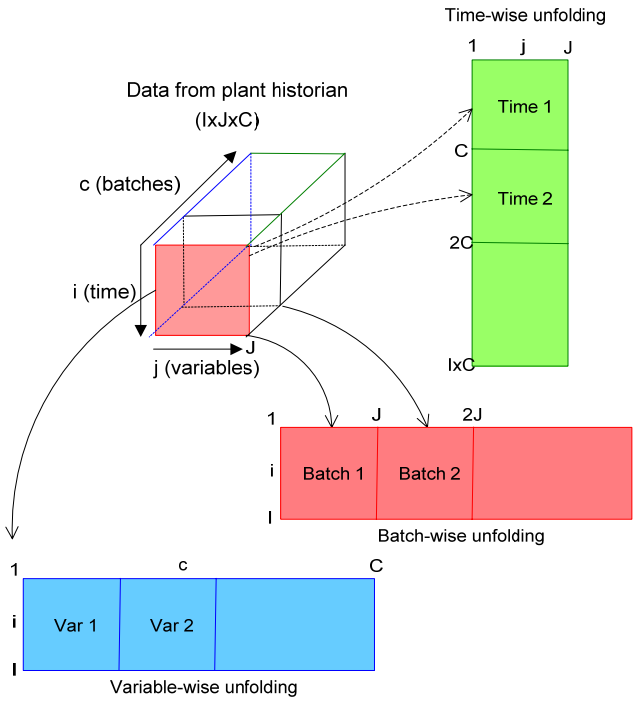


Figure B-1 : Different means of data unfolding based on MPCA approach

Each of these provides the flexibility to analyze a different type of variability in the batch-data set. Time-wise unfolding analyzes variability among samples across different batches at a specific time point, batch-wise unfolding identify abnormal batches from the 3-dimensional batch dataset, and variable-wise unfolding analyzes variability among the samples within a batch. Among the three unfolding methods, the batch-wise unfolding and variable-wise unfolding are more commonly used for batch processes monitoring. A major shortcoming of batch-wise unfolding is the need of complete batch dataset, which often limits their direct application for online monitoring. In this work, unless otherwise noted, the multiway-PCA technique adopted is based-on the variable-wise unfolding technique. Such a way of unfolding allows abnormal samples to be identified from a given batch trajectory.

PCA generates a linear static model of the data matrix *X*. When the data contains dynamic information, as in the case with data from batch processes and transitions, applying PCA/MPCA on the data does not capture the actual correlations between the variables, but only a linear static approximation. Even though there have been examples where static PCA have been applied to isolate disturbances in a dynamic system, the latent variables generated (scores) will be auto-correlated or cross-correlated. This can lead to misleading results: both false positives and false negatives. In such scenarios, a dynamic-PCA is more appropriate (Ku *et al.*,1995). The PCA assumes that all samples taken at different time instants are statistically independent. For non-stationary systems, the current values of process variables will depend on the past values due to time-lag behavior of the chemical processes. *X(t)* can be augmented with previous observations. The dynamic PCA correlation matrix is constructed by performing PCA projection to the vectors of current measurements stacked with time-lagged information, $X_D(t)$, where

$X_D(t) = [X(t) \quad X(t-1) \quad \ldots \quad X(t-l)]$, and l is the number of previous observations that are correlated to the current sample. In the general case,

$$X_D(t) = \begin{bmatrix} X(1) & X(0) & \cdots & X(1-l) \\ X(2) & X(1) & \cdots & X(2-l) \\ \vdots & \vdots & \ddots & \vdots \\ X(t) & X(t-1) & \cdots & X(t-l) \end{bmatrix}, \qquad \text{(Eq B-1)}$$

where $X(t)$ is the two dimensional observation vector in the training dataset at time t. The extracted dynamic model is implicitly multivariate autoregressive (AR) (Ljung and Glad, 1994) if process inputs are included (Ku *et al.*, 1995). The use of DPCA for fault diagnosis in feedback-controlled processes was reported by Shi and Tsung, (2003). Dynamic PCA was also shown to be able to cluster time varying states more effectively than conventional PCA approaches by Srinivasan *et al.* (2004). Chen and Liu (2002) integrated MPCA with DPCA to capture the correlation among different runs of a discontinuous batch-process. Though DPCA improves the performance of PCA models by incorporating process dynamics, its modeling approach based on a single model is still unable to capture the dynamics of non-stationary processes as in the case of transient operations.

Appendix C: PCA Similarity

The similarity between two PCA models can be measured using the S_{PCA}. S_{PCA} measures the similarity between two PCA models based on the angles between the spaces of the first *k* PCs (Krzanowski, 1979). Let A and B be two groups with *n* variables. The similarity between the two groups is quantified by comparing their principal components subspaces *L* and *M*, which are the eigenvector matrices corresponding to the first *k* PCs (Krzanowski, 1979):

$$S_{PCA}(A,B) = \frac{1}{k}\sum_{i=1}^{l}\sum_{j=1}^{l}\cos^2\theta_{ij} = \frac{trace(L'MM'L)}{k}, \qquad \text{(Eq C-1)}$$

where θ_{ij} is the angle between the i^{th} PC of *L* and the j^{th} PC of *M*. A modified form of S_{PCA} is given by Singhal and Seborg (2002) by normalizing the similarity factor with variances:

$$S^{\lambda}_{PCA}(A,B) = \frac{\sum_{i=1}^{l}\sum_{j=1}^{l}\lambda_i^A\lambda_j^B\cos^2\theta_{ij}}{\sum_{i}^{l}\lambda_i^A\lambda_i^B}. \qquad \text{(Eq C-2)}$$

S_{PCA} is in the range of 0 to 1. A smaller values of S_{PCA} indicate low similarity between models whilst large value signifies high similarity.

Appendix D: Bandwidth Selection for Kernel Density Estimator

The shape of the constructed kernel density is normally determined by the choice of kernels while the kernel width is controlled by a bandwidth matrix, *H*. In practice, the choice of kernels has only minimal impact over the estimated density, $\hat{f}$. The more critical issue lies in the value of bandwidth selector used, *H* or sometimes being referred to as smoothing parameters. A number of measures to estimate *H* can be found in Wand and Jones (1994). The appropriate choice for *H* should be dependent on the purpose for which the kernel-density estimate technique is to be used upon. Some commonly used approaches for determining *H* include: normal scale rule, least squares cross-validation, plug-in type bandwidth selector, etc. Most of the proposed techniques used standard error criteria when estimating $\hat{f}$. Two of the most popularly used error criteria include mean squared error (MSE):

$$MSE = E(\hat{f}(x,H) - f(x)), \qquad \text{(Eq D-1)}$$

and mean integrated squared error (MISE):

$$MISE = E\int \{\hat{f}(x,H) - f(x)\}^2 dx, \qquad \text{(Eq D-2)}$$

Here, $\hat{f}(x,H)$ is the estimated density function, and $f(x)$ is the real density function of *X*.

The *normal scale rule* computes the bandwidth matrix, H_{AMISE} by calculating the optimal value of asymptotic-MISE (AMISE) of the data density, which is the approximation to MISE. H_{AMISE} is given as (Wand and Jones, 1994):

$$H_{AMISE} = \left[\frac{R(K)}{\mu_2(K)^2 R(f'')I}\right]^{1/5}, \qquad \text{(Eq D-3)}$$

where $\mu_2(K) = \int x^2 K(x)dx$, $R(K) = \int K(x)^2 dx$, $R(f'') = \int f''(x)^2 dx$. The computation of H_{AMISE} is less computational intensive but often produces over-smoothed density estimate, thus the use of H_{AMISE} might capture additional region in the subspace of training data. A tighter bound can be created through *Least squares cross-validation* (LSCV). LSCV was developed by Rudemo (1982) by expanding the MISE criteria of Eq D-2 to:

$$MISE = E\int \hat{f}(x,H)^2 dx - 2E\int \hat{f}(x,H) f(x)dx + \int f(x)^2 dx . \qquad \text{(Eq D-4)}$$

The solution to minimization of Eq D-4 is given by Rudemo (1982), and Bowman (1984) as:

$$LSCV(H) = \int \hat{f}(x,H)^2 dx - 2I^{-1}\sum_{i=1}^{n} \hat{f}_{-1}(X_i,H) \qquad \text{(Eq D-5)}$$

where $\hat{f}_{-i}(x,H) = (I-1)^{-1}\sum_{j\neq i}^{I} K_h(x - X_j)$ is equivalent to the density estimate of X with X_j deleted. The optimal density estimate can then be obtained from the H that minimizes Eq D-1:

$$H_{LSCV} = \arg\min\left(LSCV(H)\right). \qquad \text{(Eq D-6)}$$

On the other hand, *plug-in type* of bandwidth estimators normally suggest means to estimate the term $R(f'')$ in Eq D-3, thus making it directly solvable. Some examples of plug-in rules can be obtained from Sheather and Jones, (1991), Scott *et al.* (1977), and Engel *et al.* (1995).