Adaptive Process Monitoring using Principal Component Analysis and Gaussian Mixture Models

by

Prince Addo

Thesis presented in partial fulfilment
of the requirements for the Degree

of

MASTER OF ENGINEERING
(EXTRACTIVE METALLURGICAL ENGINEERING)

in the Faculty of Engineering
at Stellenbosch University

Supervisor
Dr. L. Auret

Co-Supervisor
Dr. R. S. Kroon

April 2019

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

PLAGIARISM DECLARATION

1. Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.
2. I agree that plagiarism is a punishable offence because it constitutes theft.
3. I also understand that direct translations are plagiarism.
4. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
5. I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Student number: ……..

Initials and surname: ……..

Signature: ……..

Date: ……..

ABSTRACT

Principal component analysis (PCA) is a well-known technique used in combination with monitoring statistics for fault detection. Moving window PCA and recursive PCA are adaptive extensions of PCA that operate by periodically updating the monitoring model to incorporate new observations. This allows the monitoring model to cope with process behaviours that change slowly over time such as equipment aging, catalyst deactivation, and reaction kinetics drift and thereby improving monitoring performance.

Recent demands and advancements in process industries, however, may result in multimodal operations, where distinct clusters are present in measurement data. The performance of the aforementioned PCA-based monitoring techniques is hindered due to the violation of the implicit assumption that all the observed process data belong to the same Gaussian distribution. To improve monitoring performance, multimodal techniques are required. The Gaussian mixture model (GMM) is a probabilistic model that can account for the observed modes in the process data and therefore be used in the monitoring of multimode processes. However, multimodal processes also exhibit behaviours that change slowly over time, which is challenging.

This work develops a monitoring approach that extends adaptive PCA techniques to GMM, which effectively addresses the aforementioned challenge. This is done by continuously refreshing the model parameters and monitoring statistics for the PCA and GMM. Other key areas that the work focuses on are in improving the specifications for adaptive PCA protocol (taking into consideration the various model update methods) and Gaussian mixture model methods (taking into consideration the monitoring model types and data types). Also, the performance of unimodal and multimodal process monitoring approaches was assessed.

The performance of the developed approach and the improved implementations of the pre-existing methods were assessed using various case studies including unimodal and multimodal processes both with and without drift as well as various fault types. The Tennessee Eastman process and the non-isothermal continuously stirred tank reactor process are the two main simulators considered.

Results for the considered cases show improved performance for the developed approach (adaptive PCA-based GMM) as compared to PCA, adaptive PCA, and traditional GMM, in fault detection. The GMM, as expected, performed better for multimodal cases than the PCA approaches. Also, the adaptive PCA approach performed better than PCA when there is process drift.

ABSTRAK

Hoofkomponentanalise (PCA) is 'n welbekende tegniek wat gebruik word saam met moniteringstatistiek vir foutdeteksie. Bewegende venster PCA en rekursiewe PCA is aanpassende uitbreidings van PCA wat bedryf word deur die moniteringsmodel periodies op te dateer om nuwe waarnemings te inkorporeer. Dit laat die moniteringsmodel toe om by te hou met die proses gedrag wat geleidelik verander, soos toerusting wat verouder, katalis deaktivering, en reaksie kinetika dwaal, en sodoende die monitering werkverrigting verbeter.

Onlangse eise en vooruitgang in proses industrieë mag egter multimodale bedrywe tot gevolg hê, waar duidelike groepe in metingsdata teenwoordig is. Die werkverrigting van voorafgenoemde PCA-gebaseerde moniteringstegnieke word belemmer as gevolg van die oortreding van die implisiete aanname dat al die prosesdata waargeneem, aan dieselfde Gauss-verdeling behoort. Om monitering werkverrigting te verbeter, word multimodale tegnieke benodig. Die Gauss-mengselmodel (GMM) is 'n waarkynlikheidsmodel wat rekening kan hou met die waargenome modusse in die prosesdata en kan daarom gebruik word in die monitering van multimodale prosesse. Multimodale prosesse vertoon egter gedrag wat stadig met tyd verander, wat uitdagend is.

Hierdie werk ontwikkel 'n monitering benadering wat aanpassende PCA-tegnieke na GMM uitbrei, wat die voorafgenoemde uitdaging doeltreffend aanspreek. Dit word gedoen deur deurlopend die model parameters en moniteringstatistiek vir die PCA en GMM te verfris. Ander sleutelareas waarop die werk fokus is om die spesifikasies vir aanpassende PCA protokol te verbeter (die verskillende model opdatering metodes word in ag geneem) en Gauss-mengselmodelmetodes (die monitering model tipes en data tipes word in ag geneem). Die werkverrigting van unimodale en mulitmodale proses monitering benaderings is ook geassesseer.

Die werksverrigting van die ontwikkelde benadering en die verbeterde implementasies van die voorafbestaande metodes is geassesseer deur verskillende gevallestudies te gebruik, insluitend unimodale en multimodale prosesse beide met en sonder dwaal sowel as verskillende fout tipes. Die Tennessee Eastman-proses en die nie-isotermiese kontinu geroerde tenk reaktor proses is die twee hoof simulators wat oorweeg is.

ACKNOWLEDGEMENTS

I wish to thank everybody who made this project a success and particularly:

- Dr. Lidia Auret for your supervision and kind support throughout my study.
- Dr. Steve Kroon for your continuous support and supervision.
- Dr. John McCoy, for your role as my unnamed supervisor (mentor).
- Dr. Roelof Coetzer for your support and contributions.
- The process monitoring and systems research group colleagues, for your support.
- D Attah-Kyei and Patricia K Bisika for your friendship and encouragements.
- My family, for your unending support.
- The Center for Artificial Intelligence Research (CAIR) of the CSIR and Sasol for your financial support of this project.

TABLE OF CONTENTS

ABBREVIATIONS AND SYMBOLS

Acronym	Description
AIC	Akaike's Information Criterion
APCA	Adaptive Principal Component Analysis
BA	Best Approximation
BIC	Bayesian Information Criterion
CA	Closest Approximation
CPCA	Conventional Principal Component
CSTR	Continuous Stirred Tank Reactor
CUSUM	Cumulative Sum
DD	Detection Delay
DPCA	Dynamic Principal Component Analysis
EM	Expectation–Maximization
EWMA	Exponentially Weighted Moving Average
FAR	False Alarm Rate
FJ	Figueiredo-Jain
GEM	Greedy Expectation–Maximization
GMM	Gaussian Mixture Model
ICA	Independent Component Analysis
LCL	Lower Control Limit
mAB	Minimum AIC-BIC criterion
MAR	Missing (Missed) Alarm Rate
MPCA	Multiple Principal Component Analysis
MSPC	Multivariate Statistical Process Control
MWPCA	Moving Window Principal Component Analysis
NLPDF	Negative Logarithm of Probability Density Function
NOC	Normal Operating Conditions
PC	Principal Component
PCA	Principal Component Analysis
PDF	Probability Density Function
RePCA	Recursive Principal Component Analysis
ROC	Receiver Operator Characteristic
SPC	Statistical Process Control
SPE	Squared Prediction Error
SVD	Singular Value Decomposition
TAR	True Alarm Rate
UM	Update Method
UCL	Upper Control Limit

Symbol	Description
$(.)_k$	Current value
$(.)_{k+1}$	Next (or updated) value
$(.)_{k-1}$	Previous (or down-dated) value
$(\hat{t}^2)_\alpha$	Modified Hotelling T^2 statistic critical value
$\hat{\boldsymbol{t}}^2(\hat{t}^2)$	Modified Hotelling T^2 statistic vector (scalar)
δ_{ij}	Responsibility of cluster j for observation i
$\mathbf{NLPDF}_{\boldsymbol{\alpha}}$	Negative-logarithm PDF critical value vector
$F_\alpha(\upsilon, n-\upsilon)$	The upper 100α % critical point of the F-distribution
$\widehat{\boldsymbol{P}}$	Retained principal components matrix
$\widehat{\boldsymbol{T}}$	Retained score matrix
$\widehat{\boldsymbol{X}}(\hat{\boldsymbol{x}})$	Reconstructed data matrix (vector)
$\hat{\boldsymbol{\lambda}}$	Retained eigenvalues vector
$\boldsymbol{\theta}^*$	Maximum likelihood estimates for GMM parameters
$\boldsymbol{\theta}^t$	Initial GMM parameter estimates
$\boldsymbol{C}$	Correlation matrix
$\boldsymbol{D}(\boldsymbol{d})$	Input data matrix (vector)
d_1	Oldest observation of in input matrix
$\boldsymbol{E}(\mathbf{e})$	Reconstruction error matrix (vector)
γ	Number of parameters in the covariance matrix
g_j	Gaussian distribution of cluster j
$L(.)$	Likelihood function
L_υ	Cumulative variance explained by υ principal components
m	Number of process variables
n	Number of observations
$\dot{n}$	Number of observations for validation data
$\boldsymbol{P}$	Matrix of eigenvectors
$p(.)$	Probability density function
$\boldsymbol{q}(q)$	SPE statistic vector (scalar)
r	Number of modes
$r_{\max}$	Maximum plausible number of modes
$\boldsymbol{S}$	Covariance matrix
$\boldsymbol{T}$	PCA score matrix
$\boldsymbol{X}(\boldsymbol{x})$	Normalized data matrix (vector)
$\hat{\boldsymbol{Z}}(\hat{\boldsymbol{z}})$	Latent variables matrix (vector)
z_α	Normal deviate corresponding to the $(1-\alpha)$ percentile
α	Significance level of probability distribution
γ	Number of parameters of the means and mixing proportions
$\boldsymbol{\lambda}$	Vector of eigenvalues
$\boldsymbol{\mu}(\mu)$	Mean vector(scalar)
$\boldsymbol{\sigma}$	Standard deviation vector
τ	Tolerance level for EM
υ	Number of retained principal components
z	Consecutive number of observations required to trigger an alarm
$\boldsymbol{\Sigma}$	Diagonal matrix of $\boldsymbol{\sigma}$
ϱ	Mixing weight of a Gaussian cluster

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Chapter 1 introduces the thesis by providing a brief description of process monitoring after which the motivation for the study is highlighted. The aims and objectives of the project are provided as well as the scope. The thesis layout is then provided.

1.1 Process monitoring and control in industry

Process engineering industries are characterized by highly complex and integrated equipment operations and process measurements in the quest to get products from raw materials. Monitoring and control of operations are therefore critical for maintaining desired product specifications and overall plant safety in order to avoid losses. The basic monitoring framework can, therefore, be described as detecting situations that represent suboptimal process conditions which are called faults. The basic form of fault detection involves physical inspection of process units to find faults.

Advancements and increased complexity in industrial processes, however, makes the aforementioned monitoring process a challenge. This is as a result of an increase in the number of simultaneous process operations and the resulting increase in the number of process units. The shortcomings of the physical inspection approach to monitoring and control have resulted in a significant amount of negative safety and environmental impacts (Xia, Chu, and Geng, 2013).

However, with advanced instrumentation in recent years, modern process industries are able to collect a large amount of frequently measured process data. The need for process monitoring consequently shifted from physical plant inspections to analyzing process data (by monitoring process variables in order to detect faults). This led to a growth in the use of statistical control charts which use some form of distance measure to judge whether an observation is normal or abnormal.

Early monitoring approaches made use of univariate control charts such as Shewhart charts. This involves comparing observations of a single variable against control limits using univariate control charts developed by Shewhart (Shewhart, 1925) in the 1920s (Ipek, Ankara and Ozdag, 1999). However, process variables are interrelated which could result in the propagation of faults to other variables, compromising the monitoring strategy. Also, the use of the univariate control charts tends to be problematic due to the fact that many variables have to be simultaneously monitored which is an inefficient and tedious task. A monitoring approach that considers the interaction of process variables (i.e. multivariate monitoring techniques such as PCA and partial least squares) therefore became necessary.

The last three decades have thus seen a dramatic rise in the use of multivariate monitoring and control techniques, as opposed to the traditional univariate charts. Hotelling's T^2 statistic (Hotelling, 1931) and squared prediction error (SPE) (Balakrishnama and Ganapathiraju, 1998) are two common multivariate statistical process control (MSPC) measures that enable all process variables to be monitored using a single metric. However, due to the correlation in process variables as a result of energy and material balance overlaps, a single detail could be reflected in more than one variable which leads to redundancy. Also, with an increase in the number process variables, a large amount of resources are required to adequately explain the process data. Feature extraction methods (Hira and Gillies, 2015) are therefore required to extract relevant features and reduce data dimension for better implementation results.

Principal component analysis (PCA) (Jolliffe, 2002) is one of the common feature extraction methods that has gained popularity. PCA combined with Hotelling's T^2 statistic and SPE has proved to be efficient and therefore successfully applied as a better alternative to the univariate monitoring methods (Kourti, 2002). The improved performance is achieved by compensating for the correlation between measured variables and also monitoring of an information rich reduced space (that combines all the variables) rather than a number of individual charts.

The standard (conventional) PCA-based monitoring procedure involves building a model using historical data, which is used to make judgements on the process condition of future observations. Although this approach seems to provide good results, the monitoring model fails to learn new operation properties and leading to deterioration in monitoring performance (Li et al., 2000).

Recursive PCA (RePCA) (Li et al., 2000) and moving window PCA (MWPCA) (Wang, Kruger and Irwin, 2005) are adaptive extensions of PCA that were popularised in the early- to mid-2000s to address the aforementioned deficiencies of conventional PCA. These adaptive methods operate by periodically updating the monitoring model to incorporate new observations (and possibly discard old observations). This makes the monitoring model more aware of the current process operation conditions and allowing it to detect faults more accurately (Wang, Kruger and Irwin, 2005).

Although the adaptive PCA methods seem to achieve better results in coping with new process changes, recent demands of process industries to meet market specifications (i.e. different product types for consumers) result in multimodal operations (Choi, Park and Lee, 2004). Process industries currently employ the same process setup to manufacture different product specifications (e.g. different concentrations) to meet different target groups (e.g. age, weight).

This results in operating in unique regimes (modes); these regimes could be described as multimodal. The performance of the aforementioned adaptive PCA techniques is thus hindered. This is due to the fact that the monitoring model and statistics employed in the conventional and adaptive PCA techniques are based on the implicit assumption that all observed process data belong to the same Gaussian distribution. However, this assumption is, violated severely for multimodal data.

The Gaussian mixture model (GMM) (Choi, Park and Lee, 2004) is a popular statistical model used for handling clusters in data. The GMM is a probabilistic model that can account for multiple observable modes in process data. This ability helps the model achieve better results than PCA techniques in multimodal cases (Choi, Park and Lee, 2004). Although the GMM serves to provide a suitable monitoring approach in the multimodal case, it can be sensitive to outliers in the data (Peng et al., 2017). In addition, high-dimensional data make the clustering approach difficult. These challenges are effectively handled by PCA in the ability to provide a reduced-dimensional space.

The idea of PCA-based GMM (Choi, Park and Lee, 2004) was formulated to combine the two monitoring techniques: PCA serves as data pre-processor for the GMM process. This enables the extension of PCA to incorporate GMM to produce effective monitoring models that can deal with some of the problems caused by outliers and high-dimensional data.

Motivation

Although there has been extensive work in PCA, adaptive PCA, and GMM techniques, we are not aware of any work on the extension of adaptive PCA methods to GMM. The combination of adaptive PCA and GMM may produce a monitoring technique that can cope with multimodal monitoring while incorporating new observations to facilitate better monitoring performance.

Furthermore, the robustness of the adaptive methods is an issue. (Robustness of a model, in this case, have to do with the ability to deal with various fault types and still be able to adapt to process changes). Many implementations of adaptive PCA techniques focused on the size of the data window to maintain and how to efficiently update the existing model parameters with minimal computational cost. Two areas with little attention are the robustness with respect to how and when to update a monitoring model.

On when to update, some standard adaptive PCA techniques update existing models when no alarm is triggered. Model update using this approach risks the inclusion of fault data for cases

where there are intermittent faults. And since the absence of an alarm trigger does not mean an observation is in control, this could lead to missing alarms.

On how to update, adaptive PCA methods compute monitoring statistics of an observation (under investigation) using the pre-existing model parameters (means and variances). The monitoring model is then updated if an alarm is not triggered. This approach would prove unsuitable in instances where momentary observations which do not warrant an alarm trigger but are extreme, are added to the model.

Finally, little in-depth comparison of the aforementioned techniques is available for analysis to inform decision making. A comprehensive case study would, therefore, be useful for future research.

1.2 Aim and objectives of the project

The overall aim of this project is, therefore, to contribute to the development and improvement of process monitoring methods by investigating techniques to enhance available adaptive and multimodal monitoring approaches. The objectives identified to achieve the overall aim are:

1. Design and implementation of algorithms based on linear projection for the monitoring of processes exhibiting behaviours that change slowly over time, specifically, continuously updating the models and modification of the implementation to deal with data from faults.
2. Design and implementation of algorithms based on linear projection for the monitoring of processes exhibiting multimodal operation properties, specifically, GMM.
3. Design and implementation of algorithms based on linear projection for the monitoring of process changes in the forms of both multimodal operation and also exhibiting behaviours that change slowly over time, specifically, combining adaptive PCA and GMM.
4. Comparative analysis of the performance of fault detection models for multimode behaviour and behaviours that change slowly over time with simulated case studies.

1.3 Scope of the project

1) Recursive PCA and MWPCA (with modifications to the standard model update mechanisms) will be applied to fault detection and their performance will be analysed.
2) GMM (with modifications to the standard implementation techniques) will be applied to fault detection and the performance will be analysed.

3) Extension of adaptive PCA methods to GMM will be applied to fault detection and the performance will be analysed.
4) Case studies to be considered are simulated unimodal and multimodal processes. Also, processes that involve process drift would be considered for unimodal and multimodal cases. The performance of adaptive and standard GMM would be compared with each other on a non-stationary multimodal process as well. The work focuses on fault detection only and does not involve fault diagnosis for any case studies.

1.4 Thesis layout and contributions

The thesis is organized as follows: Chapter 2 presents a literature review on statistical process control in which univariate and multivariate statistical process control methods are considered with a focus on PCA and GMM. Chapter 3 follows with a background to the methodology by providing the relevant mathematical derivations for the approaches to be used with PCA, RePCA, MWPCA, and GMM. A novel model selection criterion for GMM as well as a novel model update method for adaptive PCA are provided as well. In Chapter 4, the way the employed approaches are applied is provided and adaptive PCA-based GMM, which is a novel technique is introduced. Chapter 5 then presents and discusses the fault detection results of the various approaches for the case studies considered. The thesis then concludes with Chapter 6 which summarizes the work and presents conclusions and recommendations.

CHAPTER 2: LITERATURE REVIEW

This chapter provides information on various approaches to fault detection. An overview of statistical process control is presented in Section 2.1. Next, PCA is considered as a feature extraction method in Section 2.2. Adaptive PCA (APCA) and also multimodal process monitoring and their respective limitations are presented in Sections 2.4 and 2.5.

2.1 Statistical process control

Statistical process control (SPC) involves the application of statistical tools and techniques principally for measuring and analysing process data[1]. Process operations can be described by the means (which are the averages of process measurements), variances (the spread of process measurements from the averages) and other relevant statistical descriptions of the process data. It is therefore paramount to monitor the variation in a pre-existing process to determine the presence of significant changes.

The traditional tool associated with SPC is the use of Shewhart charts introduced by Shewhart (1925). These charts are used for determining quantitative variations that occur in a process over a certain time frame. According to chart principles, if random factors play a role in the operation of a process, the process data can be described by a normal distribution (Ipek, Ankara and Ozdag, 1999). It is therefore relevant to monitor the region defined by a normal distribution in order to make judgements. The approach of using control charts for process monitoring is presented in the subsequent section.

2.1.1 Univariate statistical process control

The use of Shewhart charts involves monitoring a single process variable at a time and can, therefore, be considered a univariate method. As shown in Figure 2.1, a process variable is monitored by measuring the deviation of measurements from the known operating regime, which is defined by the operating centre (mean) and bounded by an upper control limit (UCL) and lower control limit (LCL). The control limits are calculated using suitable percentiles of the normal distribution. Other commonly used univariate charts are the cumulative sums (CUSUM) charts for change detection (Woodall and Adams, 1993) and exponential weighted moving average (EWMA) charts (Wold, 1994), which are useful in monitoring processes with small shifts in the process. These methods were introduced to account for time-varying

[1]SPC enhances early detection of faults and does not involve automated controls which focus on the correction of faults after they have occurred.

behaviours. Failure of univariate charts (both stationary and time-varying) to account for correlations between process data variables is a challenge. For example, an increase in reaction volume is expected to affect other process variables such as pressure and temperature of the reactor. Also, there is difficulty in monitoring (keeping track) of each variable simultaneously as it is inefficient. A monitoring approach that can combine all the process variables, therefore, became the dominant resulting approach–this is presented in the next section.

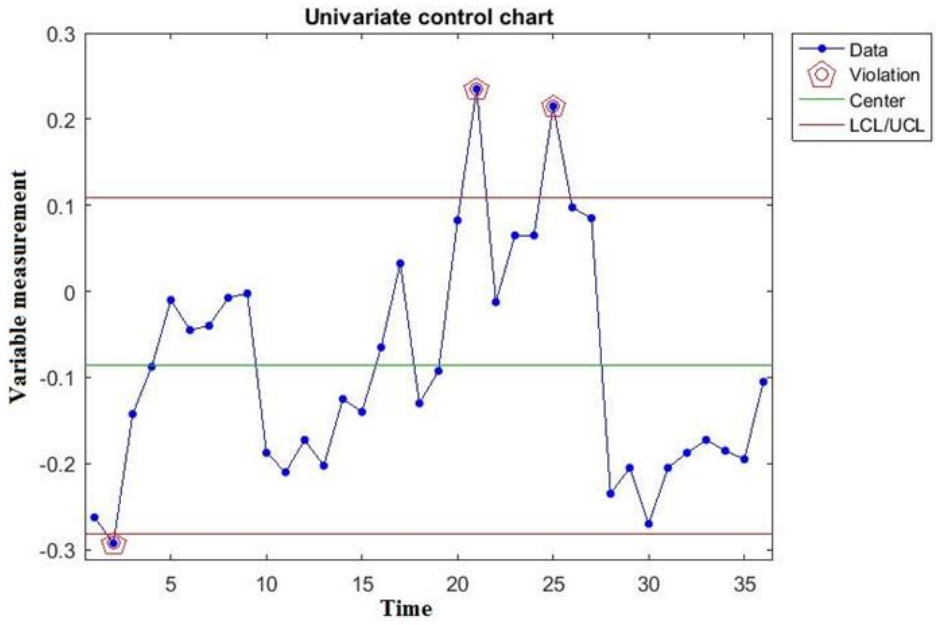


Figure 2.1: Implementation of a univariate statistical control chart. Observations below LCL or above UCL are flagged as faults.

2.1.2 Multivariate statistical process control

Multivariate statistical process control (MSPC) involves monitoring multiple variables at a time. The aforementioned univariate challenges are addressed by allowing the simultaneous monitoring of process variables (with a single monitoring chart) as well as accounting for correlations between variables. Although many multivariate statistical control procedures are reported in literature, Hotelling's T^2 statistic (Hotelling, 1947) is one of the most popular metrics and can be viewed as the multivariate equivalent of the Shewhart chart (Mason and Young, 2002). Multivariate EWMA charts (Morais et al., 2008) is another noteworthy MSPC technique.

A simple implementation of a multivariate control chart is depicted in Figure 2.2. As shown in the figure, the univariate control charts are able to detect observations beyond the individual chart (highlighted black). The observations highlighted orange, however, are not detected by the univariate charts because they fall within all the individual chart limits. The multivariate

chart with its elliptical threshold, however, can detect unusual combinations of variables and appropriately flag the observations as faults.

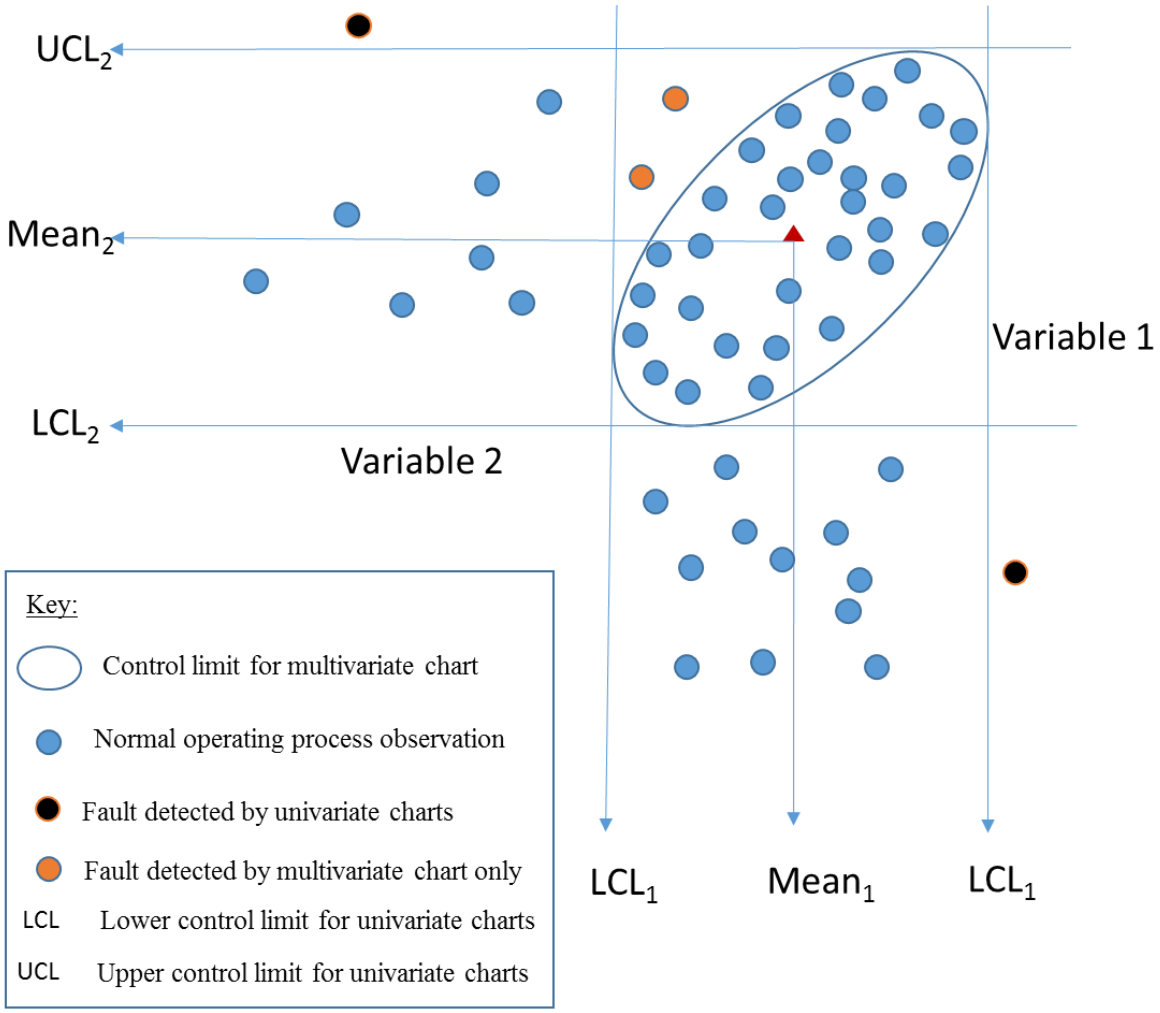


Figure 2.2: Implementation of MSPC for data with two variables. Instead of separate bounds on each process variable, an ellipsoidal limit surface is used for the two variables.

Due to correlations in the process data, a single detail (piece of information) can be reflected in more than one process variable which leads to redundancy. Feature extraction may, therefore, be required to capture the most relevant information in data before applying the relevant MSPC techniques. Feature extraction is presented in the subsequent section.

2.2 Feature extraction

The curse of dimensionality, as described by Bellman (1957), describes the negative effect of sparsity (with associated increasing noise) of data on performance as the dimensionality of data rises. Feature extraction is a general term for methods of constructing derived values (features), which are intended to be informative and sometimes non-redundant, from an initial set of measured data (Vines, 2000; Martínez and Zhu, 2005; Serradilla, Shi and Morris, 2011). The

extracted features enable the monitoring of relevant information space which could improve accurate detections and also computational efficiency. These are key issues in fault detection and machine learning respectively.

Feature extraction methods reported in literature can generally be grouped into non-linear or linear techniques[2], depending on the extraction approach (Hira and Gillies, 2015; Huang et al., 2015). Key among the linear methods is PCA, which is considered in this work. This is chosen due to its wide deployment, computational efficiency, and the potential of easier interpretation as well as the potentially reduced requirement for hyperparameter specification as compared to non-linear methods.

2.3 Principal component analysis

PCA is a dimensionality reduction technique that constructs the most relevant linear features from a large multivariate dataset which is made up of interrelated variables (Owen and Demirkiran, 2014). This is achieved by projecting the original data with high dimension to space with a low dimension. The low-dimensional space is obtained by finding the principal components (PCs), which are linear combinations of the original data variables, to create a new basis for the input space. The goal of the new basis is to provide a suitable way to re-express the data, such that it filters out the noise (unpredictable signals with no useful information), reveals the non-redundant hidden structure in the data and decouples the process variables (by projecting onto the eigenvectors which are a new undefined independent basis) (Owen and Demirkiran, 2014).

Unlike feature selection methods (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzoss, 2013), where specific variables of the dataset are selected as features, PCA allows the combination of all the process variables into features, with the most important linear combinations being selected. A typical feature selection situation would be the decision to monitor only the pressure and level measurements from a process output which consists of the aforementioned variables, along with flow, volume and temperature measurements. The use of PCA, therefore, minimizes the risk of information loss that could be associated with not monitoring the unselected variables (in the feature selection case). The application of PCA provides PCs which are linear combinations of all the process variables along with their

[2] Linear techniques can be differentiated from non-linear techniques by the fact that the former make use of linear functions to transform the data as opposed to non-linear transformations for the latter.

respective eigenvalues (which correspond to explained variances). The eigenvalues describe the significance of the PCs (i.e. most of the process variability is reflected in the PCs with high eigenvalues) and PCs associated with negligible explained variance can often be omitted–this is a design decision made depending on the required model accuracy. Figure 2.3 shows the reduction of normalized two-dimensional data (on the *x*-*y* plane) to a single dimension (x^1) with the retained dimension corresponding to the first principal component.

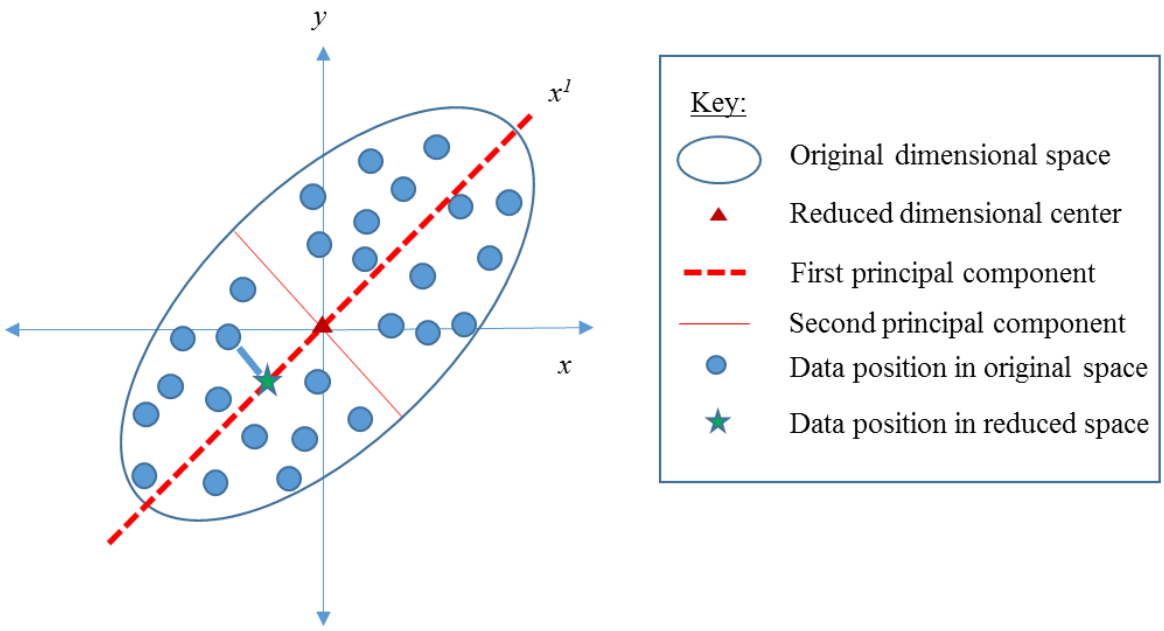


Figure 2.3: Implementation of PCA showing the original and reduced space for some data.

The amount of explained variance retained by the PCs depends on the selected number of PCs. A large number of retained PCs provides a relatively accurate model (He et al., 2006), but which may involve the modelling of noise. Several procedures are reported in literature to select the number of retained PCs (e.g. Cattell, 1966; Xia, Chu and Geng, 2013). Some of these methods include the size of explained variances; a cumulative fraction of total variation; the scree graph; and the log-eigenvalue diagram.

The cumulative fraction of explained variation approach (Voegtlin, 2004; Yu, 2011; Xia, Chu, and Geng, 2013) is one of the widely used criteria for selecting the number of retained PCs. Equation **3-1** shows the computation of the cumulative fraction of variance accounted for by the first υ PCs (L_υ).

$$L_\upsilon = \frac{\sum_{i=1}^{\upsilon} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \qquad \textbf{2-1}$$

Here, λ_i refers to the eigenvalue (explained variance) for each PC i and m refers to the total number of PCs.

Another common method is the scree graph (Cattell, 1966), which involves plotting a graph of the variance (λ_i) explained by each PC in decreasing order. The choice of how many PCs to retain is often made with a visual heuristic, which involves looking for a levelling off or an "elbow" in the plot. Figure 2.4 and Figure 2.5 illustrate the scree plot. An obvious challenge is to find an "elbow" in the plot, which in many cases becomes subjective when there are no clear breaks as is the case for Figure 2.4. (The Pareto charts provide bar charts which describe the variance explained by each PC while the associated line plot at each point provides the cumulative variance of PCs up to the current point.)

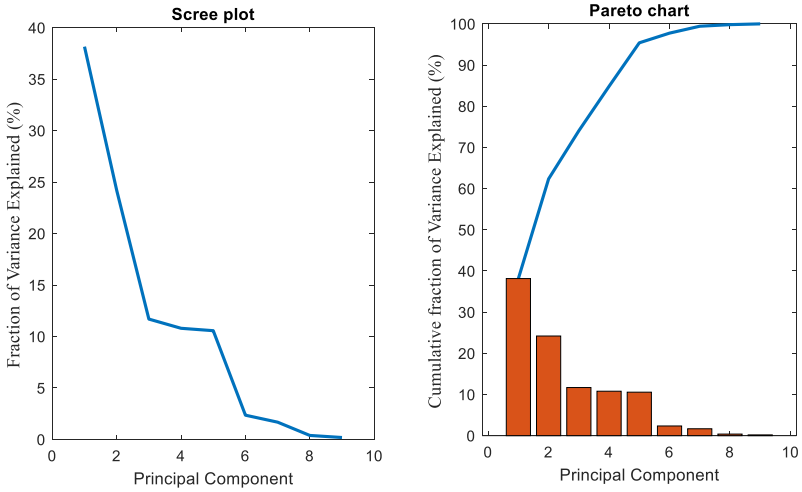


Figure 2.4: Scree plot (left) and a Pareto chart (right) for some data showing more than one 'elbow'.

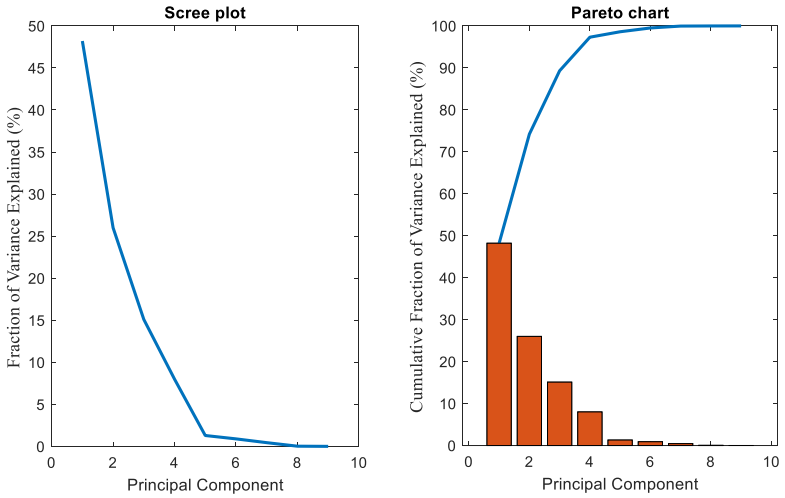


Figure 2.5: Scree plot (left) and a Pareto chart (right) for some data showing a clear 'elbow'.

Once the dimensionality reduction is complete, future observations are monitored using various monitoring statistics. Hotelling's T^2 statistic and the SPE (Q) statistic are the two predominant techniques used in conjunction with PCA in fault detection. Hotelling's T^2 statistic can be defined as the Mahalanobis distance. It provides an indication of whether there is an unusual variability (indicated by comparatively high values) within the known normal feature subspace (Slišković, Grbić and Hocenski, 2012). Intuitively, this is realised by measuring the dotted-blue line as shown in Figure 2.6, which detects that a sample (highlighted green circle) is out of range of normal operating conditions (NOC), under the assumption that the expected correlation structure is still valid. The SPE, on the other hand, checks whether an observation behaves according to the expected correlation structure and effectively characterizes the reconstruction space. Also shown in Figure 2.6 is the conceptual diagram of the SPE statistic to measure the distance (thick-blue line) of a point from its location in the original space (highlighted red) and the new point in the reduced space (highlighted-black square).

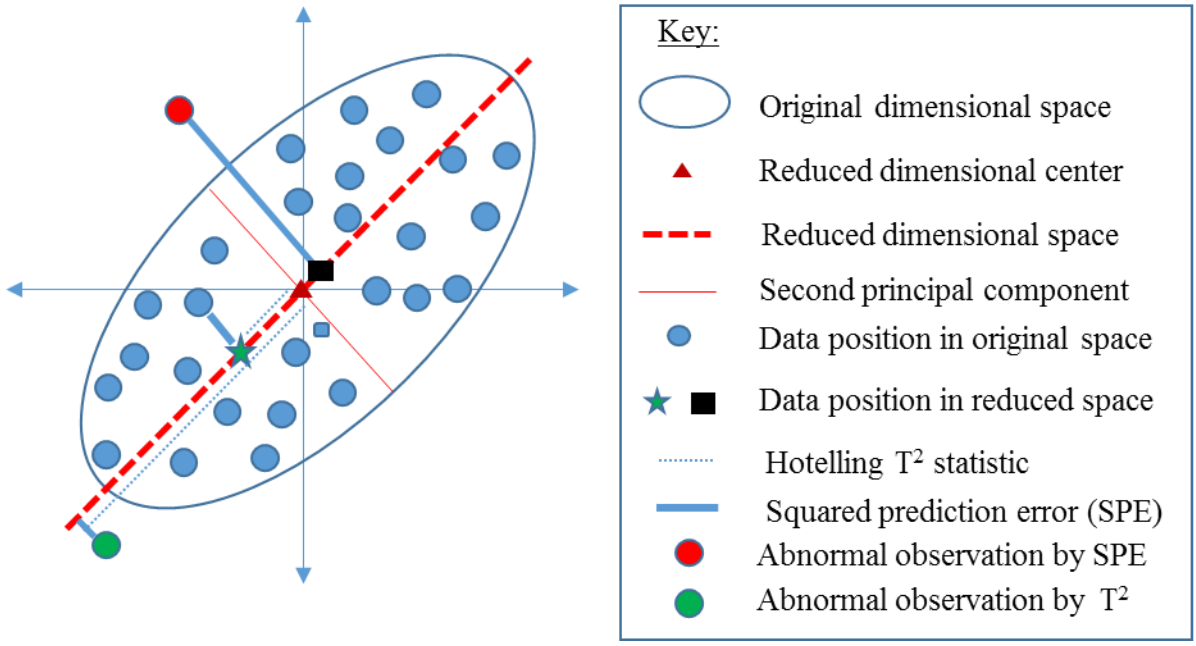


Figure 2.6: Illustration of PCA-based process monitoring.

Since observations from operations are assumed to be described by a normal distribution, the observations are expected to fall within a required distance from the model mean, resulting in a need to define a limit. The limit, also termed the detection threshold, forms the boundary around the NOC region and is used for classifying whether an observation is a fault or not; any observation that falls outside the boundary is then evaluated as abnormal. Selection of a detection threshold (presented in Section 2.3.2) is done to optimize monitoring performance

which is quantified by various performance metrics. We discuss a number of relevant metrics below.

2.3.1 Process monitoring performance metrics

The performance of a monitoring model is evaluated by its ability to correctly distinguish normal and abnormal (faulty) conditions. The instance in which a model fails to identify an anomaly is termed a missing (missed) alarm. The fraction of fault data that is incorrectly identified as NOC data is thus termed as missing (missed) alarms rate (MAR). On the other hand, the fraction of NOC data points which is incorrectly identified as faulty is termed the false alarm rate (FAR). Also, the time period between a fault manifesting and the monitoring model detecting the fault is termed detection delay (DD) for the occurrence of the fault. The FAR, MAR and DD consequently form the key performance metrics used to analyse the performance of a process monitoring model.

Due to the noisy nature of process data, a number z of consecutive observations beyond the monitoring limits is typically required to increase the likelihood of a disturbance being a fault before an alarm is triggered (Russel, Chiang, and Braatz, 2000). Figure 2.7 shows instances for $z = 3$ where one or two observations beyond the threshold for non-faulty data are not flagged as false alarms. The value of z is a tuneable parameter that shows the level of confidence needed to trigger an alarm. Large values of z imply an increase in the confidence that a fault has occurred, but limits the ability to detect a fault quickly.

The ideal detection threshold, independent of the z value, is the optimal point on the receiver operator characteristic (ROC) curve, as presented in the next section.

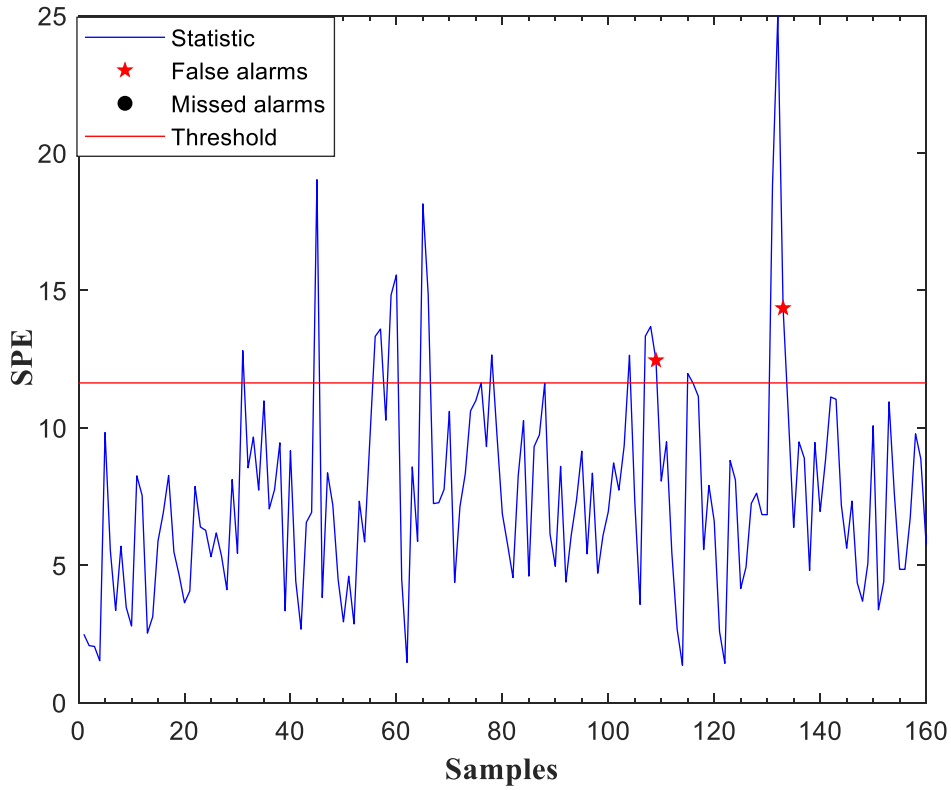


Figure 2.7: Instances of observations beyond the threshold not flagged as false alarms.

2.3.2 Detection threshold from the ROC curve

Similar to the selection of *z*, selection of the detection threshold requires a trade-off–in this case, between MAR and FAR: raising the detection threshold to a higher value will lead to more faults being missed on the one hand, but will reduce the FAR on the other hand. A common way to find the optimal point is to make use of a ROC curve (Pudil, Novovieova and Kittler, 1994). The ROC curve, as shown in Figure 2.8, is a plot of true alarm rate (TAR[3]) against FAR as the detection threshold changes. A high area under the curve indicates a good performance. A random (unbiased) decision as indicated by the blue line on the figure should provide an equal FAR and TAR.

Similar to the scree plot, the ROC curve helps to find the point (threshold value) beyond which an increment would not improve monitoring performance in terms of TAR and FAR. Also, the ROC curve helps compare different monitoring approaches, as each model will have a different curve, and therefore a different area under the curve.

[3] TAR can be defined as 1-MAR, i.e. instances in which anomalies occur and are flagged appropriately.

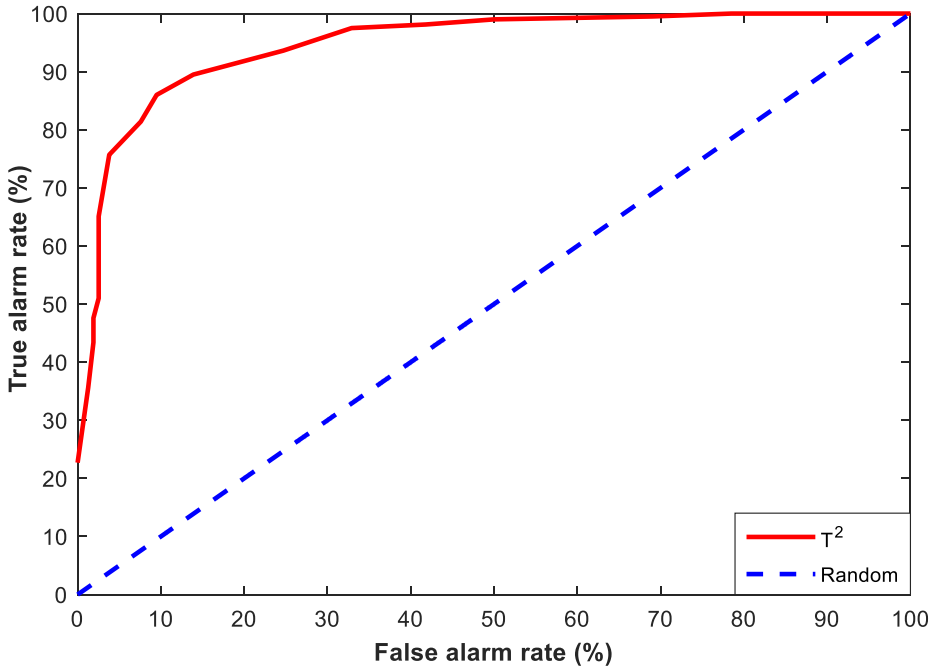


Figure 2.8: A ROC curve for sample data.

It is noteworthy that the ROC curve can only be used to find an optimal threshold if there is fault data in a training set otherwise TAR is undefined. However, since fault data are not always available during initial training of the model, what the ROC curve can be used for is to check the performance of methods against each other; i.e. it is not used for threshold tuning but rather used to compare methods.

The performance of an implemented monitoring technique is therefore adjudged by the alarm rates and detection delays. This is done for a PCA approach by considering its ability to handle time-varying processes in the next section.

2.3.3 Process monitoring using PCA and time invariance

Typically, a model trained on a set of data is used to evaluate whether new observations are normal or faulty. However, processes are typically not completely stationary and can change with time. Process changes such as equipment aging and wearing, catalyst deactivation, sensor aging, and others cause the normal operating conditions region of process data to expand and move beyond the monitoring limits with time (Wood and Hines, 2012). Therefore, normal process drifts can lead to the originally trained model becoming invalid. This in effect compromises the monitoring performance.

The need for models that continuously learn during process operations thus became relevant. These methods are termed adaptive methods due to their ability to adapt to process changes.

RePCA and MWPCA are the two most popular adaptive PCA methods. Dynamic PCA (DPCA) presented by Ku, Storer, and Georgakis (1995), which involves adding the lagged values of the observed variables as an input for PCA estimation, is another noteworthy APCA technique. DPCA is not considered in this work because it serves to correct the assumption of statistical independence in the monitored observations without an actual model update with time (which is desired). DPCA basically introduces autocorrelations into the data (when the variables are lagged) to make the assumed statistical independence in the observations (which is valid for only long sampling intervals) hold for frequent (shorter) sampling intervals (which are desired for quick fault detection) (Russell, Chiang, and Braatz, 2000).

2.4 Adaptive PCA

RePCA and MWPCA are two of the APCA variants with slight differences in the implementation of the model update technique. The difference is reflected in the window size of data the model captures at each time. The general framework for both methods involves the incorporation of information from new observations into the monitoring model. The information incorporated into the model is reflected in the means and variances that describe the monitoring model. Also, while RePCA allows continual incorporation of information from new observations to the model, MWPCA only allows the model to capture a fixed number of observations. This is achieved by removing the effect of the oldest observations from the model as new observations are included.

RePCA, the earlier version of the two methods, was presented by Li et al. (2000), based on the adaptation mechanism of recursive partial least squares (Qin, 1998). The RePCA and recursive partial least squares works were motivated by the work of Dayal and MacGregor (1997) on achieving efficient computation by updating the pre-existing process model rather than completely building a new model from scratch.

Li et al. (2000) presented extensive work on how to iteratively update a pre-existing correlation matrix and determine the retained PCs. This was successfully applied to a rapid thermal annealing process. However, the challenge of the ever-growing data size of the RePCA model is a concern. Two major challenges faced in this regard are the negative impact on processing time as well as the slow response to changes (due to old observations which are unrepresentative of the current process).

To correct the issue of slow response, Choi et al. (2006) proposed the use of a weighting factor (forgetting factor) that gives relatively more relevance to newer observations. Another issue

tackled in the work was the extension of the RePCA work from the sample-wise update approach to a block-wise update approach. The difference between the two is that the sample-wise update is carried out for each observed normal observation, while the block-wise update occurs after a specified number of normal observations are observed.

Similar to the work of Choi et al. (2006), Portnoy et al. (2016) proposed a weighted RePCA, which also involves the application of weights to older observations but with a forgetting factor varying over time. The forgetting factor at each point is computed as the ratio of the current observation index *i* to the sum of indices of all the observations previously incorporated (i.e. 1/50 for a model that is already built on data with 50 observations).

Although the incorporation of the forgetting factor tends to make the RePCA model more representative of the current process, the issues of slow adaptation speed and increased processing time persist. Also, the selection of the forgetting factor or the weights without a priori knowledge of the fault conditions is difficult. All of the works on the RePCA mentioned above were successfully applied to various case studies.

MWPCA was introduced by Wang, Kruger, and Irwin (2005), which was termed 'fast MWPCA' to handle the issues of selecting a forgetting factor and the adaptation speed. The introduced method ensures a fixed number of observations form part of the monitoring model. This was successfully applied to a fluid catalytic cracking unit.

The method of MWPCA implementation, therefore, allows the model to keep current process changes and discard old observations which are unrepresentative of the current process unlike with RePCA. This, therefore, results in a faster response of the MWPCA model to changes in operation as opposed to the slower response in RePCA. Although MWPCA seems to perform better than RePCA in this regard, the risk of incorporating faulty data in the monitoring model is a concern. Maintaining a reasonable window size is key to avoid this problem. An illustration of the difference between RePCA and MWPCA is presented later in Figure 3.5.

He and Yang (2008) then proposed a variable window approach for the MWPCA. Their argument was that processes sometimes change rapidly or slowly. Consequently, in the case of the former, where the window covers too much outdated sample data, the fixed window size MWPCA fails to detect the quick changes. The method of adjusting the window size to track the changes so that an efficient window is decided at each time stamp tends to be quite effective.

In all the works of the MWPCA methods (Wang, Kruger and Irwin, 2005; He and Yang, 2008), the model needs a reasonable amount of data to capture enough process variance in order to

make correct judgements. However, this is problematic when there are only a few observations to build a model. Building the model on data that does not capture enough process variance would be ineffective since it would tend to adapt to all changes (both faults and normal).

To resolve the aforementioned problem, Jeng (2010) worked on the combination of RePCA and MWPCA. The proposed method primarily involves recursively updating the model using the RePCA technique which allows the size of the dataset to grow. Once a defined window size is met, the MWPCA technique is then applied.

While work by Ayech, Chakour and Harkat (2012), took a further look at window size and its impact on monitoring performance, Schmitt et al. (2016) provide a general framework on parameter selection for the forgetting factor, detection thresholds and window size.

Another APCA approach is 'variance sensitive adaptive threshold-based PCA' presented by Alkaya and Eker (2011). Although it does not fully implement a full model update as the RePCA and MWPCA methods do, it involves weighting the thresholds as a function of the variance and mean (i.e. multiplying the threshold at the previous time stamp by a value to create the threshold value at the current time stamp) to mitigate false alarms and is therefore worthy of mention.

Although all the above-mentioned works provided insight into the RePCA and MWPCA methods, robustness to outliers is still an issue. Some standard update procedures for some implementations of APCA methods involve updating the model if no alarm is triggered. However, considering the fact that a consecutive number of observations are required to be beyond the threshold for some update techniques, the absence of an alarm trigger is not enough evidence to update.

Consider three observation cases (*A*, *B* and *C*) as shown in Figure 2.9, for a situation where two consecutive observations are needed to trigger an alarm. In case *A*, some update approaches would update the model after the observation at time t-1 (highlighted orange) is observed because the previous observation was in control. The model would, however, not be updated after the observation at time t (highlighted green) because an alarm would be triggered (due to 2 observations out of control). In case *B*, the model would be updated for both green and orange highlighted observations (because of no alarm trigger). Also, in case *C*, the model would be updated for both green and orange highlighted observations (irrespective of how extreme the orange observation is) because there is no alarm triggered. An oscillation or a large number of

consecutive observations required to trigger an alarm can thus make the monitoring model susceptible to fault inclusions and compromise monitoring performance.

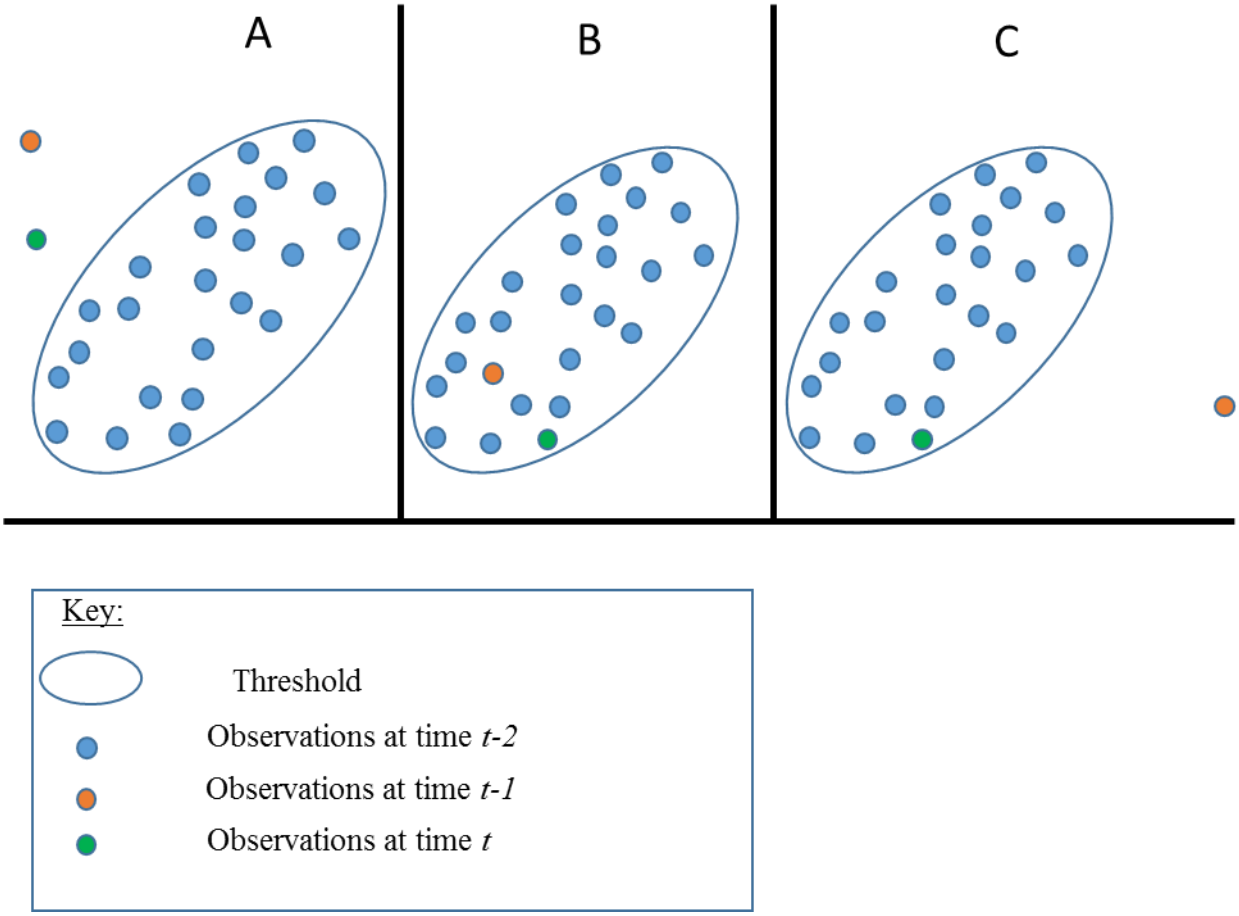


Figure 2.9: Model update conditions and their effects on performance for APCA methods– considering various observation types (*A*, *B*, *C*).

2.5 PCA and APCA limitations

Although PCA and APCA techniques achieve good results for case studies as presented by Jeng (2010), Li et al. (2000), Wang et al. (2005), the implementation method and associated monitoring statistics assume all the observations belong to the same group. This is manifested in the computation of detection thresholds as well. However, process data do not always follow a Gaussian distribution as is typically assumed. In industrial processes, operating condition shifts are normally experienced due to changes in various factors such as feedstock, product specification, set points, and manufacturing strategy (Yu and Qin, 2008). In cases where operating condition changes result in distinct clusters of data, the unimodal data distribution assumption becomes inadequate.

Such operating condition shifts may render conventional PCA and other unimodal MSPC methods inappropriate. As shown in Figure 2.10, it can then be a challenge to fit a sensible detection threshold to the reduced space of the data in order to be able to detect the associated faults since the detection threshold can only be elliptical in this case. The quality of the detection threshold is judged on how well it fits the data. Multimodal MSPC methods are desirable in such cases.

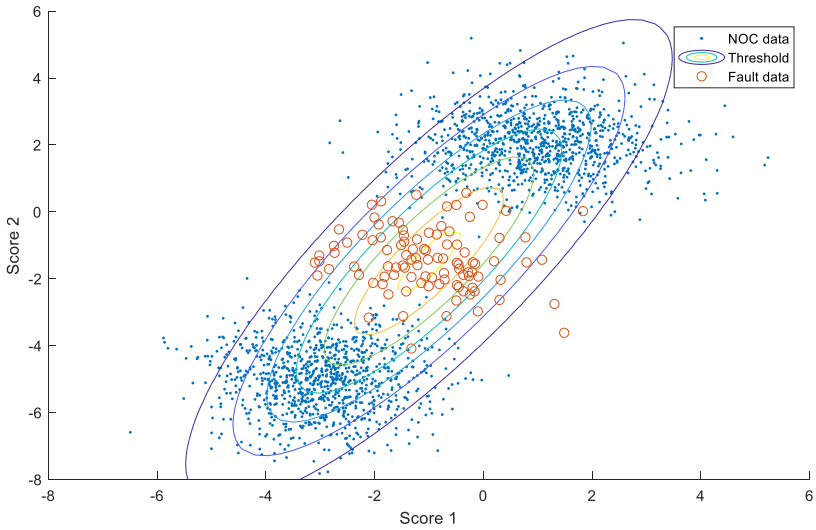


Figure 2.10: Fitted threshold to reduced space of observed data using a unimodal approach. Note how the threshold is fitted over all the clusters and the fault data remains unidentified.

Multiple PCA (MPCA) models (Zhao, Xu and Zhang, 2004) and independent component analysis (ICA) (Cao et al., 2003; Rashid and Yu, 2012) are a few of the many approaches available in literature to monitor multimodal processes. While MPCA allows PCA models for each mode of operation to be built–which is achieved by expertly separating the data into groups and then building models. ICA finds independent latent components from the input process variables, where the derived components are both statistically independent and non-Gaussian (Stone, 2002). MPCA would be more useful if operating condition changes are known and allocation of data to each.

The GMM, on the other hand, is a commonly used pattern recognition technique that is used to build probabilistic models of data (Helfand and Stillinger Jr, 1964; Tresp, 2001). GMM model the unknown distribution of a dataset by identifying clusters and fitting a Gaussian model for each cluster. This approach of finding individual modes is similar to that of MPCA but with the additional advantage of combining all the modes into a single probabilistic model that can be singly monitored. The GMM are fit using the expectation-maximization (EM) algorithm

(Dempster, Laird and Rubin, 1977), which starts with an initial guess of the model parameters and finds the locally optimal model parameters that define a number of multivariate Gaussian distributions (i.e. the covariance matrices, means, and weights). The expectation stage of the EM-algorithm computes the allocations of the data to the clusters given the estimated model parameters, while the maximization step computes the updated model parameters for the generated allocations. Figure 2.11 shows the GMM for sample data, where two clusters are fitted to the data and the faulty data can consequently be detected.

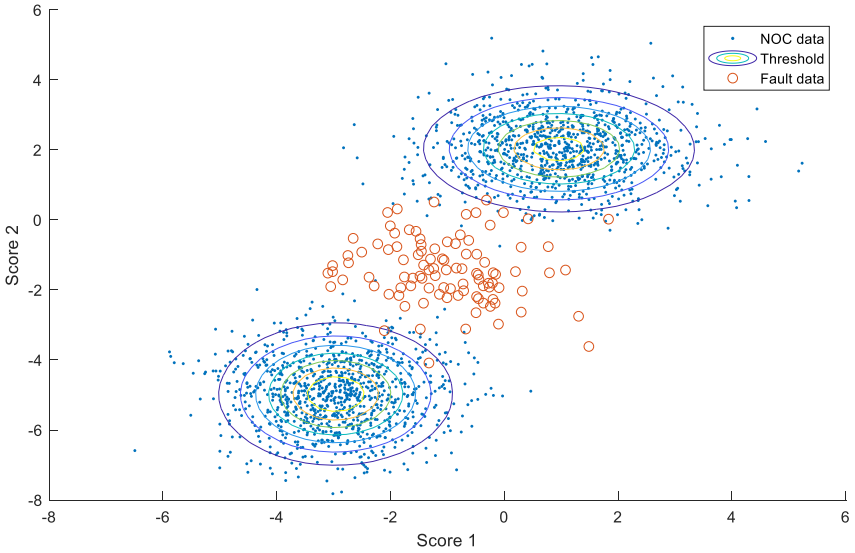


Figure 2.11: Fitted thresholds to reduced space of observed data using a multimodal approach. Note how individual thresholds are fitted over all the identified clusters and the fault data identified.

Although the EM algorithm does well in estimating the unknown Gaussian distributions, it requires the number of modes in the data to be pre-specified. Many approaches available in literature resolve this by using a model selection criteria which tells how well the developed model fits the data by considering the trade-off between goodness-of-fit and complexity. The Bayesian information criterion (BIC) (Schwarz, 1978) and Akaike's information criterion (AIC) (Akaike, 1973) are the two most popular general model selection criteria. The BIC penalizes for complexity more severely and therefore tends to select simpler models that might underfit the data; the AIC, on the other hand, penalizes less for complexity and tends to select more complex models that might over fit the data (Erar, 2011). The complexity of a GMM is determined by the number of different modes as well as the covariance matrix type to be fitted

to the data. Increasing the number of modes directly implies more parameters to be estimated and thus more complexity.

Another two important algorithms associated with EM processes to aid in model selection are the Figueiredo-Jain (FJ) (Figueiredo and Jain, 2002) and the Greedy EM (GEM) (Vlassis and Likas, 2002) algorithms. Both methods aim to find the number of modes for an unsupervised clustering approach. While the FJ algorithm does this by starting off the EM process with a very large number of modes and removing modes that tend to become singular, the GEM algorithm, on the other hand, starts with a single mode and then adds more modes into the mixture model one after the other. Once a good model is fit to the data, the model is deployed in monitoring new observations. The EM combined with the AIC/BIC outperforms the FJ and GEM in classification if only the number of determined clusters is close to the true number; the FJ outperforms the GEM in classification accuracy and also has a worse failure (crash) rate than the GEM (Paalanen, 2004).

2.5.1 Process monitoring using GMM

Process monitoring using GMM involves using the probability values of the observations. Similar to the T^2 and SPE statistics, the probability values provide an idea of whether the observed data is novel or known to the model. The posterior probability can be used to monitor how an observation fits a particular mode by adjudging observations with values greater than 0.5 for a specific mode as part of that mode. To minimise the risk of misclassification for observations that almost belongs to two modes (i.e. around 55% for mode one and 45% for mode two), the individual posterior probabilities are weighted (by their mixing weights) to provide a unified (global) probability index as employed in the works of Xie and Shi (2012) and Yu and Qin (2008).

2.5.2 Limitations of process monitoring using GMM

Although GMM tends to achieve good results for multimodal processes (Choi, Park and Lee, 2004; Yu and Qin, 2008), clustering or classification tend not to do well for raw data (Choi, Park and Lee, 2004; Paalanen, 2004). This is because raw process data usually contain many outliers and noise and also could be sparse (Peng et al., 2017). Also, high dimensional data is undesirable and negatively impacts clustering performance (in terms of stability) (Yu, 2011).

PCA-based GMM (Choi, Park and Lee, 2004; Yu, 2011) combines PCA and GMM by using PCA as a preprocessoring step for the GMM clustering process. This enhances monitoring in the feature space which is almost always desired due to the unpredictable nature of data (in terms of noise) as well as the potential of fitting a rather parsimonious GMM model.

The basic PCA-based GMM technique involves computing the retained PCs and projecting the data into the reduced subspace to produce the scores. The scores are then used as inputs into the GMM process which clusters the data in the reduced space.

The PCA-based GMM was successfully implemented by Choi et al. (2004) on a non-isothermal CSTR process (Yoon and MacGregor, 2001). The implementation, however, made use of the GMM in building local PCA models and monitoring each mode by their respective Hotelling's T^2 and SPE statistics. This approach looks similar to MPCA but with the use of GMM to find the modes instead of a supervised way. However, Yu (2011a) implemented the PCA-based GMM with the unified PDF inference which was successfully applied to a semiconductor manufacturing process.

Similar to conventional PCA, PCA-based GMM and GMM monitoring methods are time-invariant. This presents the GMM processes with the same challenges faced by PCA in learning to update the monitoring model with time.

Adaptive GMM was investigated by Xie and Shi (2012) for the purpose of addressing the aforementioned limitation by retraining the GMM model at each timestamp. This is done by using a moving data window to update the Gaussian component (mode) to which the current observation is assigned in a sample-wise manner (i.e. for each observation). Although this method was successfully applied, the techniques face the same challenges of clustering performance as that mentioned for GMM (i.e. raw data space and high dimensions and poor clustering results) as well as GMM model component update at each instance.

In summary, the chapter provided a literature review on statistical process control by introducing the univariate and multivariate approaches. Feature extraction with focus on PCA was considered. Also, the limitation of PCA to cope with process drift introduced APCA methods of which the two key approaches, recursive and MWPCA were considered. PCA and APCA generalization of a single Gaussian for every process make a multimodal monitoring a challenge; multimodal approaches were then considered with a focus on GMM. Finally, the limitation of GMM in its inability to handle multimodal processes which exhibit drift properties was also discussed.

The next chapter provides a background to the methodology, in which mathematical derivations for PCA, recursive, MWPCA, GMM, and PCA-based GMM are provided.

CHAPTER 3: BACKGROUND TO METHODOLOGY

This chapter proposes the techniques identified to address each stated objective. The methodology is divided into two sections. The first section describes PCA and APCA approaches. In here, the novel model update condition for APCA is presented as well. The second section describes GMM and PCA-based GMM approaches.

3.1 Overview

The APCA approaches are suitable for monitoring of process exhibiting behaviours that change slowly over time (for which PCA is deficient). These approaches seek to maintain a monitoring model with detection thresholds that are representative of the current process state. This is done by periodically incorporating new NOC data.

The PCA-based GMM method addresses the objective of monitoring of multimodal processes. The approach follows the approach of PCA but builds more than one monitoring model to monitor each observation by the model that best describes it. This avoids generalizing of detection thresholds over all modes, which proves to be problematic in the PCA and APCA methods.

Figure 3.1 presents a simple flow diagram showing how a monitoring approach is made adaptive by adding new observations to the model data which builds the monitoring model (when deployed online). Figure 3.2 presents a simple flow diagram showing the individual monitoring models of GMM and PCA.

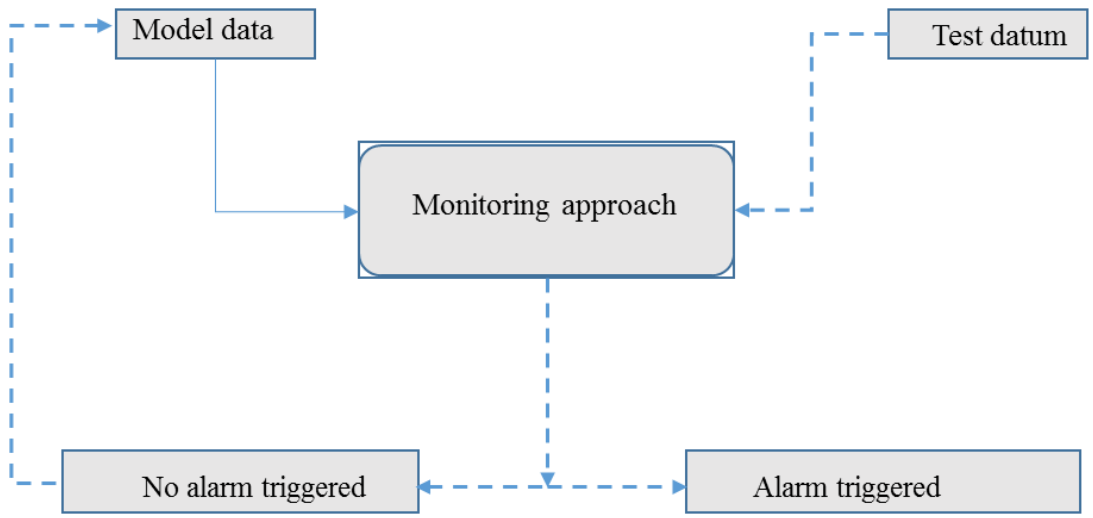


Figure 3.1: Adaptation of the monitoring model (for an approach) by the addition of new observations test datum) to the model data.

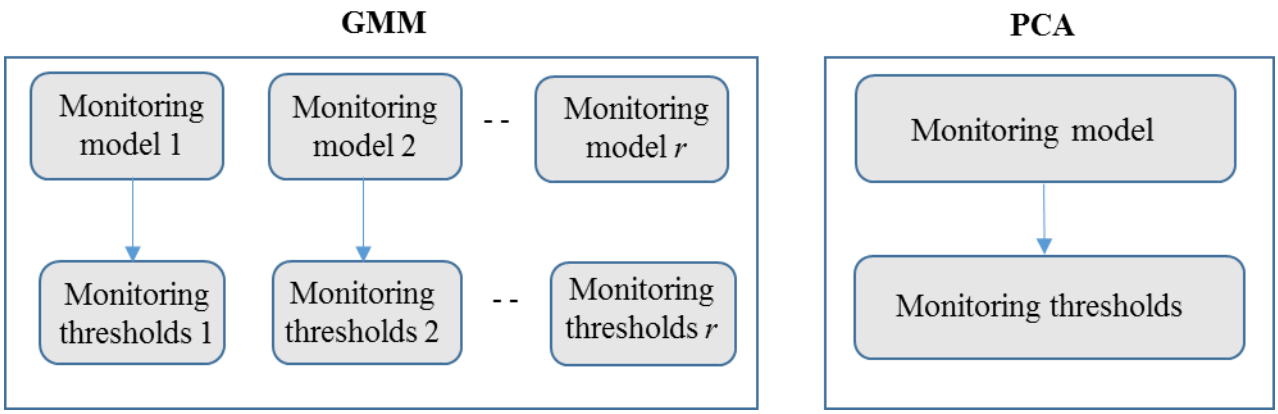


Figure 3.2: Monitoring approaches for GMM (left) (for r number of modes) and PCA (right) showing the individual monitoring models.

3.2 PCA

3.2.1 Overview

PCA (Jolliffe, 2002) as a framework for fault detection allows the monitoring of observations in a feature space of reduced dimension (Russel, Chiang, and Braatz, 2000; Kourti, 2002; Shlens, 2009; Kruger et al., 2012).

The overall implementation involves splitting a NOC data into a training and validation data. The training data is used for model development and the validation data is then used to test the generalisability of the derived model parameters and monitoring statistics and tuning the hyperparameters. The hyperparameter tuning is better with test data if it is available to assess how the model performs in presence of specific faults. The developed model is then deployed online for monitoring new observations. The conceptual diagram for process monitoring using PCA is illustrated in Figure 3.3.

The monitoring strategies involve using the scores and the reconstructed data. This is done via the modified Hotelling's T^2 statistic and SPE respectively. Figure 3.4 provides a view of how the monitoring statistics are applied in combination with PCA for a simple two-dimensional example. The figure shows how the normalized data[4] in the original input space is transformed into the lower dimensional feature space by finding the PCs and retaining the ones with the highest explained variance (which is highlighted by the dashed-red line in this case). The derivation of the model parameters and relevant monitoring statistics are presented in the next subsection.

[4] Normalized data as used in this work defines data where the variables have been centered and scaled such that the resulting data has a mean of zero and standard deviation of one.

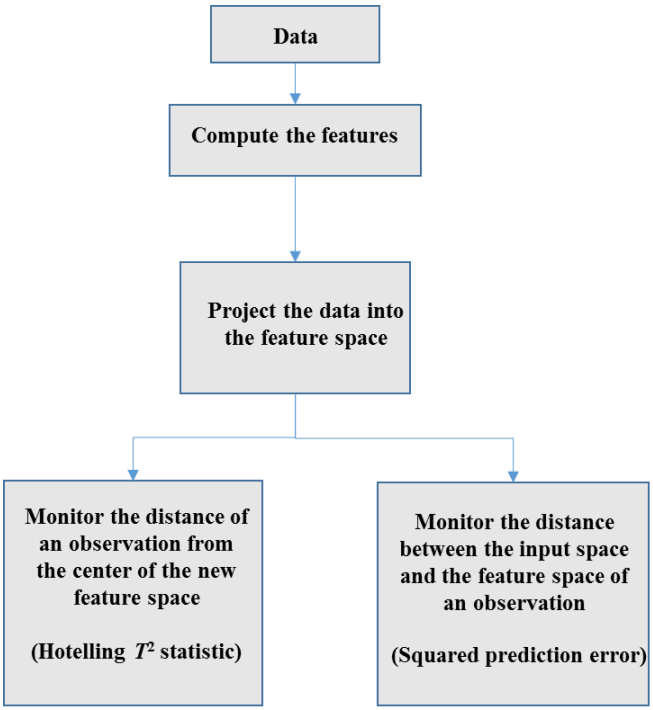


Figure 3.3: Conceptual diagram for fault detection using PCA.

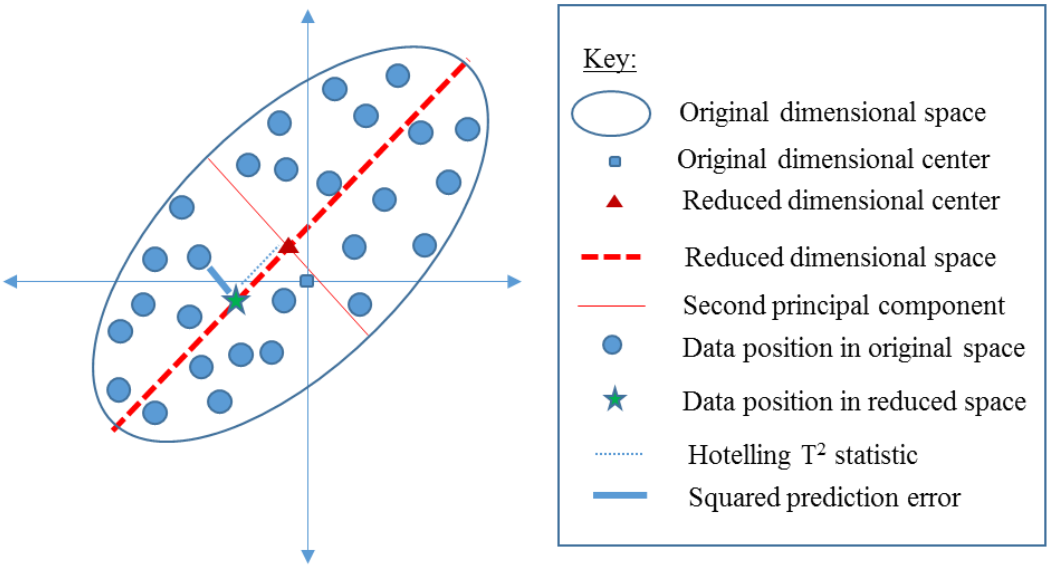


Figure 3.4: Pictorial view of dimension reduction of normalized 2-D data to 1-D by projection onto the first PC. A sample original observation which is highlighted as a blue circle transforms to a green star in the reduced space.

3.2.2 Algorithm

The derivation of the model parameters for PCA involves the calculation of the PCs and their respective variances. These can be obtained by or from the singular value decomposition (SVD) of the normalized training data or via the eigendecomposition of the correlation matrix.

The eigendecomposition of the correlation matrix is used in this work and presented below.

The raw data $n \times m$ matrix, $\boldsymbol{D} \in R^{n \times m}$, is normalized and decomposed as:

Normalizing the input

1. For the input data matrix $\boldsymbol{D}$, with n observations and each of the m columns representing a measured variable, let μ_j denote the mean of the j-th variable:

$$\mu_j = \frac{\sum_{i=1}^{n} d_{i,j}}{n} \qquad \textbf{3-1}$$

2. Calculate the standard deviation of the j-th variable, σ_j:

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n} \left(d_{i,j} - \mu_j\right)^2}{n}} \qquad \textbf{3-2}$$

3. Each variable is normalized. The variable value for each observation is centred by subtracting the variable's mean μ_j and scaling the result by dividing it by its standard deviation σ_j:

$$x_{i,j} = \frac{d_{i,j} - \mu_j}{\sigma_j} \qquad \textbf{3-3}$$

Here $x_{i,j}$ and $d_{i,j}$ are respectively the normalized and original i^{th} sample of the j^{th} variable. The result of normalizing data matrix $\boldsymbol{D}$ is data matrix $\boldsymbol{X} \in R^{n \times m}$.

Computing the correlation matrix

The correlation matrix, $\boldsymbol{C} \in R^{m \times m}$, for the normalized data, $\boldsymbol{X}$, is computed as:

$$\boldsymbol{C} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} \qquad \textbf{3-4}$$

Computing the loading vectors by using eigen-decomposition

Next eigendecomposition of the correlation matrix is performed by solving the eigenvalue equation:

$$\boldsymbol{CP} = \boldsymbol{\lambda P} \qquad \textbf{3-5}$$

$\boldsymbol{P} \in R^{m \times m}$ and $\boldsymbol{\lambda} \in R^{1 \times m}$ respectively represent the matrix of eigenvectors and the vector of eigenvalues produced as solutions.

Retaining first υ loading vectors with largest eigenvalues

The fraction of variance in the normalized data matrix accounted for by a PC is computed as shown in Equation **3-7**.

$$\frac{\lambda_i}{\Sigma \lambda_i}. \qquad \textbf{3-7}$$

Thus using the υ PCs corresponding to the υ largest eigenvalues collectively account for some fraction L_υ:

$$L_\upsilon = \frac{\sum_{i=1}^{\upsilon} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \qquad \textbf{3-8}$$

$\widehat{\boldsymbol{P}} \in R^{m \times \upsilon}$ and $\hat{\boldsymbol{\lambda}} \in R^{1 \times \upsilon}$ respectively represent the PCs and variances retained once υ is decided.

Computing the retained scores

The score matrix $\widehat{\boldsymbol{T}}$ is next computed as:

$$\widehat{\boldsymbol{T}} = \boldsymbol{X}\widehat{\boldsymbol{P}} \qquad \textbf{3-6}$$

Reconstruction

The transformation of the score matrix back into the original dimensional observational space can now be computed as in Equation **3-9**.

$$\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{T}} \left(\widehat{\boldsymbol{P}}\right)^T \qquad \textbf{3-9}$$

Here $\widehat{\boldsymbol{X}} \in R^{n \times m}$ represents this reconstructed data generated from the scores. Note the error introduced by discarding the eigenvectors.

Reconstruction error

The difference between the normalized input data and the reconstructed data is the reconstruction error and denoted by $\boldsymbol{E}$:

$$\boldsymbol{E} = \boldsymbol{X} - \widehat{\boldsymbol{X}} \qquad \textbf{3-10}$$

3.2.3 Monitoring statistics and control limits

Monitoring using PCA involves the use of the SPE (Q) and Hotelling's T^2 statistics for the retained loadings, which will be called the modified Hotelling's T^2 statistic and denoted by $\hat{\boldsymbol{t}}^2$. Let $\boldsymbol{q}$ denote the vector of SPE statistics for all observations. The SPE for the observation i is then q_i and computed by:

$$q_i = \sum_{j=1}^{m} \left(x_{i,j} - \hat{x}_{i,j}\right)^2, \qquad \textbf{3-11}$$

while the computation of $\hat{\boldsymbol{t}}^2$ for the first υ scores of observation i is given by:

$$(\hat{t}^2)_i = \sum_{j=1}^{\upsilon} \frac{\left(\hat{t}_j\right)^2}{\lambda_j} \qquad \textbf{3-12}$$

where $\hat{t}_j$ is the j-th column entry of the score matrix $\hat{\boldsymbol{T}}$.

Equation **3-11** can intuitively be seen as the difference between an observation x_i and its reconstructed value $\hat{x}_i$. The q_i and $(\hat{t}^2)_i$ are the test statistics calculated for comparison to the control limits, which follow in Equations **3-13** and **3-14**.

The modified Hotelling's T^2 statistic control limit, $(\hat{t}^2)_\alpha$, is calculated from the critical value of an F-distribution (Russell, Chiang and Braatz, 2000):

$$(\hat{t}^2)_\alpha = \frac{\upsilon(n-1)(n+1)}{n(n-\upsilon)} F_\alpha(\upsilon, n-\upsilon) \qquad \textbf{3-13}$$

Here, $F_\alpha(\upsilon, n-\upsilon)$ represents the upper 100α % critical point of the F-distribution with υ and n - υ degrees of freedom, with n being the number of observations. *The degrees of freedom are impacted by the number of retained components υ and observations n.*

The detection limit for Q statistic with a significance level α as approximated by Jackson and Mudholkar (1979) is:

$$q_\alpha = \varphi_1 \left[\frac{h_o z_\alpha \sqrt{2\varphi_2}}{\varphi_1} + 1 + \frac{\varphi_2 h_o (h_o - 1)}{\varphi_1^2} \right]^{\frac{1}{h_o}} \qquad \textbf{3-14}$$

where $\varphi_i = \sum_{j=\upsilon+1}^{n} \sigma_j^{2i}$, $h_o = 1 - \frac{2\varphi_1\varphi_3}{3\varphi_2^2}$ and z_α is the normal deviate corresponding to the $(1 - \alpha)$ percentile.

3.2.4 Fault detection method

Fault detection involves determining whether a process measurement exhibits normal or abnormal behaviour. This is done by normalizing using the means and standard deviations of each input dimension and then projecting using the retained PCs identified during training.

The modified Hotelling's T^2 and SPE statistics for the new point are computed as shown in Equations **3-12** and **3-11** respectively and checked against the calculated thresholds, $(t_A^2)_\alpha$ and q_α. The observation is deemed to be abnormal if it is beyond the detection thresholds for one or both of the statistics.

Due to the random nature of observations, a consecutive number of observations must be over the threshold to increase the confidence in a fault occurring before an alarm is triggered, usually three observations (Choi, Park and Lee, 2004; Choi et al., 2005; Ayech, Chakour and Harkat, 2012).

The next section considers RePCA which is an APCA approach.

3.3 RePCA

3.3.1 Overview

RePCA (Li et al., 2000) and MWPCA (Wang, Kruger and Irwin, 2005) (presented in Section 3.4) are the APCA methods considered in this work. RePCA has a similar algorithm as conventional PCA but updates the retained model each time an observation becomes available. In doing so, the method attempts to capture the most recent data variation to adapt to normal process changes and thereby reduce false alarms. RePCA seeks to address the issue of slow changes that vary over time that occur in process industries, which cannot be predicted or accounted for during the development of the monitoring model.

The assumption is that most of the new observations from slow drift are NOC data.

3.3.2 Algorithm

The RePCA methodology involves augmenting the initial model with new observations. This changes the correlation matrix and the subsequent parameters derived from the correlation matrix. For the data window initially with *n* observations, the model parameters are computed following the same procedure of PCA as listed in Section 3.2.2. Adding a new observation to the data window increases the number of observations to $n + 1$ and the correlation matrix is updated as presented in the next section.

3.3.2.1 Adaptation of the correlation matrix

As the data window (initially set at the outset) is augmented with a new observation, changes occur in the means and standard deviations of the process variables which impact the correlation matrix of the data window. The changes in the properties as a result of the augmentation is described as follows:

Computing new mean, variance and correlation matrix

For some variable *w*, let its previous (initial), current and updated (next) values be denoted by w_{k-1},w_k, and w_{k+1} respectively. Also, let the diagonal matrix of some mean $\boldsymbol{\sigma}$ be denoted by $\boldsymbol{\Sigma}$. For the data window, the initial mean $\boldsymbol{\mu}_k$, initial variance $\boldsymbol{\sigma}_k^2$ and the initial correlation matrix $\boldsymbol{C}_k$ are updated for a new observation $\boldsymbol{d}_{k+1}$ as:

$$\boldsymbol{\mu}_{k+1} = \frac{n(\boldsymbol{\mu}_k) + \boldsymbol{d}_{k+1}}{n+1} \quad \textbf{3-15}$$

$$\boldsymbol{\sigma}_{k+1}^2 = \frac{n}{n+1}\boldsymbol{\sigma}_k^2 + \frac{1}{n+1}(\boldsymbol{d}_{k+1} - \boldsymbol{\mu}_{k+1})^2 + (\boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k)^2 \quad \textbf{3-16}$$

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{d}_{k+1} - \boldsymbol{\mu}_{k+1}}{\boldsymbol{\sigma}_{k+1}} \quad \textbf{3-17}$$

$$\boldsymbol{C}_{k+1} = \frac{n}{n+1}\boldsymbol{\Sigma}_{k+1}^{-1}\boldsymbol{\Sigma}_k\boldsymbol{C}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_{k+1}^{-1} + \boldsymbol{\Sigma}_{k+1}^{-1}(\Delta\boldsymbol{\mu})^T(\Delta\boldsymbol{\mu})\,\boldsymbol{\Sigma}_{k+1}^{-1} + \frac{1}{n+1}\boldsymbol{x}_{k+1}^T\boldsymbol{x}_{k+1} \quad \textbf{3-18}$$

where $\Delta\boldsymbol{\mu} = \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k$.

The updated correlation matrix $\boldsymbol{C}_{k+1}$ provides a new basis to compute new model parameters and also monitoring statistics and their critical values. The decomposition of the new correlation matrix follows the method outlined in conventional PCA.

3.3.3 Monitoring statistics and adaptation of control limits

RePCA implements the same monitoring statistics as that of conventional PCA. The computation of monitoring statistics for the updated window follows the same method as that outlined for conventional PCA in Section 3.2.3 but with updated model parameters.

3.3.4 Fault detection method

RePCA follows the same fault detection procedure as in the conventional PCA (see Section 3.2.4). The difference, in this case, is that the threshold changes at each time interval as the model updates. In order for an observation to exhibit NOC, the monitoring statistics at each

interval is checked against their corresponding thresholds to make sure they are not beyond their respective thresholds.

3.3.5 Conditions to update model

The decision to update a model depends on some heuristics to establish if the observation under analysis is worth a model update. The three common update techniques available are presented as follows:

The first update method (UM-1) updates the model if no alarm is triggered for a consecutive number of observations. For example, irrespective of the value of *z*, if at least one observation of the *z* observations has both the SPE and T^2 statistic below their respective thresholds, the model is updated (Zhao, Xu, and Zhang, 2004; Jeng, 2010).

The second update method (UM-2) prevents the update of the model if any of the current observation's statistics (SPE and T^2 in this case) are out of control. That is, it is independent of the *z* value (Xia, Chu, and Geng, 2013).

The third update method (UM-3) requires that *z* observations for both of the monitoring statistics (SPE and T^2) must be in control before an update can occur. To put things in perspective, UM-3 is a special case of UM-2 with $z = 1$ (Tien, 2005; Zhou et al., 2016).

The next section presents the MWPCA which is the other APCA approach considered.

3.4 MWPCA

3.4.1 Overview

In contrast to RePCA, MWPCA adapts to new observations while using a fixed window size of data. This is achieved by dropping old observations as new ones are added. Wang, Kruger, and Irwin (2005) first developed MWPCA which ensures constant adaptation speed and quick response to changes and called it fast MWPCA. Figure 3.5 shows how the training data window gets updated with new observations for the two approaches.

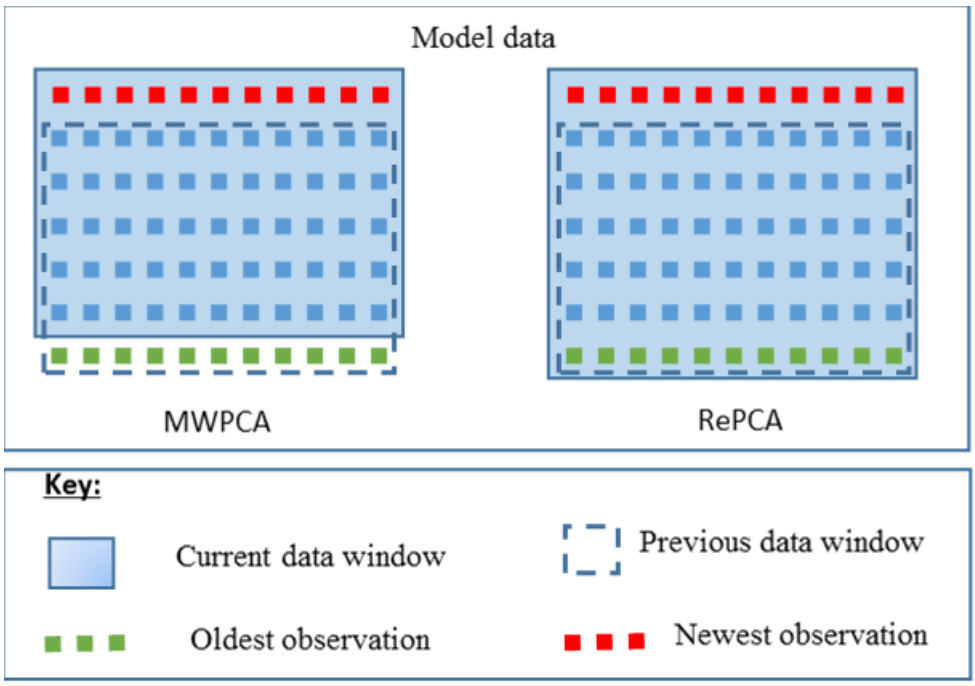


Figure 3.5: Differences in model update methods for RePCA and MWPCA. The different blocks of data (i.e. red, blue and green) have the same number of variables and add no extra information apart from specifying the times of observations (i.e. newest or oldest).

3.4.2 Algorithm

The MWPCA algorithm follows the approach described for RePCA but with a fixed window size of data. The updated window follows an eigendecomposition method to derive the updated model parameters.

The modification here is the removal of the oldest observation from the current data window to create a down-dated data window before adding the newest observation to create an updated window. The changes in the correlation matrix as a result of updating the window is described as follows:

Removing the oldest observation (down-dating)

Removing the oldest observation $\boldsymbol{d}_1$ creates a down-dated (previous) mean, variance and correlation matrix. As stated previously, let the diagonal matrix of some mean $\boldsymbol{\sigma}$ be denoted by $\boldsymbol{\Sigma}$. Also for some variable w, let its previous (initial), current and updated (next) values be denoted by w_{k-1},w_k, and w_{k+1} respectively. The down-dated mean $\boldsymbol{\mu}_{k-1}$, variance $\boldsymbol{\sigma}_{k-1}^2$ and correlation matrix $\boldsymbol{C}_{k-1}$ are computed as:

$$\boldsymbol{\mu}_{k-1} = \frac{n(\boldsymbol{\mu}_k) - \boldsymbol{d}_1}{n-1} \qquad \textbf{3-19}$$

$$\boldsymbol{\sigma}_{k-1}^2 = \frac{n}{n-1}\boldsymbol{\sigma}_k^2 - \frac{n}{n-1}\Delta\boldsymbol{\mu}_1{}^2 - \frac{1}{n-1}(\boldsymbol{d}_1 - \boldsymbol{\mu}_k)^2 \quad \textbf{3-20}$$

$$\boldsymbol{C}_{k-1} = \frac{n}{n-1}\boldsymbol{\Sigma}_{k+1}^{-1}\boldsymbol{\Sigma}_k(\boldsymbol{C}_k - \boldsymbol{\Sigma}_k^{-1})\,(\Delta\boldsymbol{\mu}_1)^T(\Delta\boldsymbol{\mu}_1)\boldsymbol{\Sigma}_k^{-1} - \frac{1}{n}\boldsymbol{x}_{k-1}^T\boldsymbol{x}_{k-1}\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_{k+1}^{-1} \quad \textbf{3-21}$$

where $\Delta\boldsymbol{\mu}_1 = \boldsymbol{\mu}_{k-1} - \boldsymbol{\mu}_k$.

Adding a new observation (updating)

Adding a new observation $\boldsymbol{d}_{k+1}$ creates an updated mean $\boldsymbol{\mu}_{k+1}$, variance $\boldsymbol{\sigma}_{k+1}^2$ and correlation matrix $\boldsymbol{C}_{k+1}$ which are computed as:

$$\boldsymbol{\mu}_{k+1} = \frac{(n-1)(\boldsymbol{\mu}_{k-1}) + \boldsymbol{d}_{k+1}}{n} \quad \textbf{3-22}$$

$$\boldsymbol{\sigma}_{k+1}^2 = \frac{n-1}{n}\boldsymbol{\sigma}_{k-1}^2 + (\Delta\boldsymbol{\mu}_2)^2 + \frac{1}{n}(\boldsymbol{d}_{k+1} - \Delta\boldsymbol{\mu}_2)^2 \quad \textbf{3-23}$$

$$\boldsymbol{C}_{k+1} = \frac{n-1}{n}\boldsymbol{\Sigma}_{k+1}^{-1}\boldsymbol{\Sigma}_{k-1}\boldsymbol{C}_{k-1}\boldsymbol{\Sigma}_{k-1}\boldsymbol{\Sigma}_{k+1}^{-1} + \boldsymbol{\Sigma}_{k+1}^{-1}\,(\Delta\boldsymbol{\mu}_2)^T(\Delta\boldsymbol{\mu}_2)\boldsymbol{\Sigma}_{k+1}^{-1} + \frac{1}{n}\boldsymbol{x}_{k+1}^T\boldsymbol{x}_{k+1} \quad \textbf{3-24}$$

where $\Delta\boldsymbol{\mu}_2 = \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_{k-1}$.

Same as RePCA, the updated correlation matrix $\boldsymbol{C}_{k+1}$ provides a new basis to compute new model parameters and also monitoring statistics and their critical values.

3.4.3 Monitoring statistics and adaptation of control limits

MWPCA implements the same monitoring statistics as that of the conventional and RePCA. The computation of monitoring statistics for the updated window follows the same method as outlined for conventional PCA in Section 3.2.3 but with changing retained PCs υ over time.

3.4.4 Fault detection method

MWPCA follows the same fault detection procedure as RePCA in Section 3.5.4.

The next section considers the GMM which is a multimodal approach.

3.5 GMM

3.5.1 Overview

GMM (Yan, Hyewon and Soohyun, 2008; Yu, 2012) is a common machine intelligence technique used for modelling data. GMM describes complex process data as a mixture of a number of local Gaussian models and learns the underlying distributions in data. Such learned models may help account for nonlinearity and multimodal features as may be experienced in process industries.

The overall GMM procedure we use involves splitting NOC data into a training and a validation dataset. While the training data is used for the development of the model, the validation data serves to help in the selection of the hyperparameters of the derived model and its monitoring statistics. The developed model is then deployed online for monitoring of new observations.

Monitoring using GMM is a multimodal approach (which finds multiple clusters) in contrast to the PCA approach which assumes there is a single cluster in the training data (and therefore a unimodal approach is used). The monitoring statistic employed in GMM is the probability value of an observation. This specifies how closely an observation follows the model created by the GMM training data.

The procedure of learning the model parameters and determining relevant monitoring statistics is presented in the next sections.

3.5.2 Algorithm

For a given dataset $\boldsymbol{D} \in R^{n \times m}$ with m process variables, the observations are assumed to come from some number r of possible operating conditions. The value of r specifies the expected number of clusters in the data. Assuming the observations are independent and identically distributed, the probability density function (PDF) for an observation $\boldsymbol{d}$ denoted by $p(\boldsymbol{d})$ is a weighted sum of the Gaussian PDFs $g_1, g_2, \ldots, g_r$ and it is computed as:

$$p(\boldsymbol{d}) = \sum_{j=1}^{r} \varrho_j g_j(\boldsymbol{d}|\boldsymbol{\mu}_j, \boldsymbol{S}_j) \qquad \textbf{3-25}$$

Here, $\boldsymbol{S}_j$ and $\boldsymbol{\mu}_j$ are respectively the covariance matrix and the mean of the j^{th} mixture component.

For a normal distribution, the parameter list $\boldsymbol{\theta}$ that defines the Gaussian mixture density consists of the cluster means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_r$, the cluster covariance matrices $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_r$ and the cluster weights $\varrho_1, \varrho_2, \ldots, \varrho_r$, as shown in Equation **3-26**.

$$\boldsymbol{\theta} = (\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}_1, \ldots, \varrho_r, \boldsymbol{\mu}_r, \boldsymbol{S}_r) \qquad \textbf{3-26}$$

The mixture weights of the j^{th} component are ϱ_j, whereby $0 \leq \varrho_j < 1$ is true for all components, and $\sum_{j=1}^{r} \varrho_j = 1$. The mixture weight ϱ_j represents the probability that a new observation belongs to the cluster j. Figure 3.6 shows a fitted Gaussian mixture density for some data with two modes.

The individual component densities are described by normal distribution PDFs: g_j given by Equation **3-27**.

$$g_j(\boldsymbol{d}|\boldsymbol{\mu}_j, \boldsymbol{S}_j) = |2\pi \boldsymbol{S}_j|^{-0.5} \times \exp[-0.5(\boldsymbol{d} - \boldsymbol{\mu}_j)^T S_j^{-1}(\boldsymbol{d} - \boldsymbol{\mu}_j)] \qquad \textbf{3-27}$$

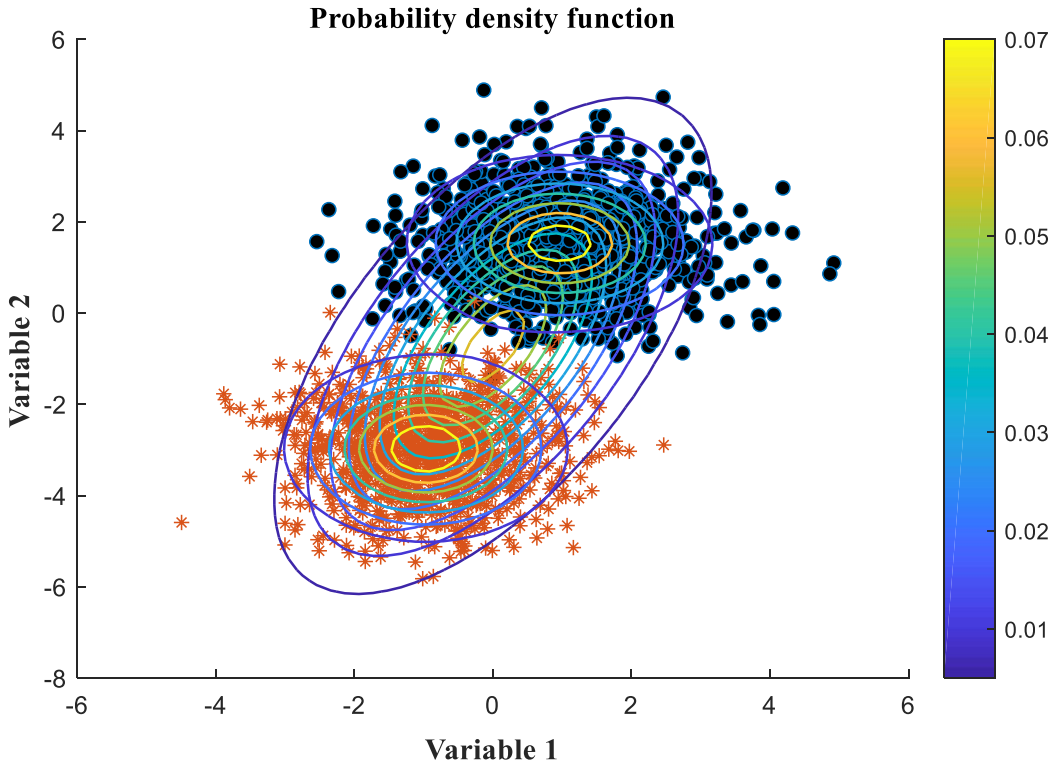


Figure 3.6: An example of Gaussian mixture density fitted to t data with two clusters.

Learning the values of $\boldsymbol{\theta}$ (with the size dependent on the number of clusters) that describe $\boldsymbol{D}$ involves estimation of the parameters that best fit the data. The procedure for estimating the Gaussian mixture density is presented next.

3.5.2.1 Estimation of Gaussian mixture density

Estimation involves evaluation of how well the postulated distribution with estimated parameters represents the data (or how confident are we that the postulated distribution generated the data). The measure of how well the estimated parameters fit the data is termed

the likelihood. The likelihood of the derived parameters given the data $L(\boldsymbol{\theta}|\boldsymbol{D})$ is defined as a product of conditional probability density functions and is formulated as shown in Equation **3-28**.

$$L(\boldsymbol{\theta}|\boldsymbol{D}) = \prod_{i=1}^{n} p(\boldsymbol{d}_i|\boldsymbol{\theta}) \qquad \textbf{3-28}$$

The aim of the estimation process is to find a value for $\boldsymbol{\theta}$ that maximizes the likelihood function, denoted as $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\boldsymbol{D}) \qquad \textbf{3-29}$$

Taking the log of the likelihood function in Equation **3-28**, yields Equation **3-30** transforming the product of potentially small likelihoods into a sum of logs, which is easier to distinguish from 0 in computation. The Equation in **3-30** is therefore maximized instead of the likelihood function in Equation **3-28** because it is computationally easier to handle.

$$\log L(\boldsymbol{\theta}|\boldsymbol{D}) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{r} \varrho_j g_j(\boldsymbol{d}_i|\boldsymbol{\mu}_j, \boldsymbol{S}_j)\right) \qquad \textbf{3-30}$$

Finding $\boldsymbol{\theta}^*$cannot be analytically solved by taking the derivative of this log-likelihood function and setting it to zero. This is because the approach has no closed form solution and is intractable. The log likelihood function is rather numerically optimized using the expectation maximization (EM) algorithm, which is an iterative procedure that moves from an initial guess of the parameter estimates $\boldsymbol{\theta}^t$ to locally optimal parameter estimates $\boldsymbol{\theta}^*$.

The EM algorithm is presented in the next section.

3.5.2.2 EM Algorithm

The EM algorithm (Dempster, Laird and Rubin, 1977) is an iterative method for finding the maximum parameter estimates for the likelihood distribution of incomplete data. It is used in maximum likelihood estimation of the GMM where an analytical approach is not possible.

The EM algorithm introduces hidden/latent variables $\hat{\boldsymbol{z}}$ for each observation such that knowledge of the latent variables would simplify the maximization of the likelihood. Consequently, the known data $\boldsymbol{D}$ is interpreted as incomplete data. The missing part $\hat{\boldsymbol{Z}}$ provides knowledge of which cluster produced each observation $\boldsymbol{d}$. As a result, for each observation $\boldsymbol{d}$, there is a one-hot binary vector $\hat{\boldsymbol{z}} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_r]$, where $\hat{z}_j = 1$, if the observation was produced

by cluster j, or $\hat{z}_j = 0$ if not. The complete data (i.e. the observed data $\boldsymbol{D}$ and the hidden data $\hat{\boldsymbol{Z}}$) log-likelihood is subsequently formulated as shown in Equation **3-31**.

$$\log L\left(\boldsymbol{\theta}\middle|\boldsymbol{D},\hat{\mathrm{Z}}\right) = \sum_{i=1}^{n}\sum_{j=1}^{r}\hat{z}_{ij}\log\{\varrho_j g_j(\boldsymbol{d}_i|\boldsymbol{\mu}_j,\boldsymbol{S}_j)\} \qquad \textbf{3-31}$$

The iterative process of EM algorithm consists of two procedures, which is the expectation step and the maximization step. The expectation step (E-step) computes the distribution of the latent variables given the current parameter estimates and the data. Let $\eth_{ij}$ denote the expectation of observation $\boldsymbol{d}_i$ belonging to the j^{th} cluster for the current model parameter estimates. The $\eth_{ij}$ called "responsibilities" is computed as:

$$\eth_{ij} = \frac{\varrho_j g_j(\boldsymbol{d}_i|\boldsymbol{\mu}_j,\boldsymbol{S}_j)}{\sum_{p=1}^{r}\varrho_p g_p(\boldsymbol{d}_i|\boldsymbol{\mu}_p,\boldsymbol{S}_p)} \qquad \textbf{3-32}$$

The maximization step (M-step) computes the updated values of the parameter estimates given the current estimated posterior probabilities.

$$\varrho_j{}^{t+1} = \frac{1}{n}\sum_{i=1}^{n}\eth_{ij} \qquad \textbf{3-33}$$

$$\mu_j{}^{t+1} = \frac{\sum_{i=1}^{n}\eth_{ij}\,\boldsymbol{d}_i}{\sum_{i=1}^{n}\eth_{ij}} \qquad \textbf{3-34}$$

$$\boldsymbol{S}_j{}^{t+1} = \frac{\sum_{i=1}^{n}\eth_{ij}\,(\boldsymbol{d}_i-\mu_j{}^{t+1})(\boldsymbol{d}_i-\mu_j{}^{t+1})^T}{\sum_{i=1}^{n}\eth_{ij}} \qquad \textbf{3-35}$$

Equation **3-33** can be interpreted as updating the mixture weights ϱ_j of cluster j by computing the proportion of the observations that belong to that cluster. This is obtained by computing the cluster PDF with the previous estimates of the parameters (see Equation **3-27**) and then calculating the average of the posterior probabilities of each sample point belonging to the component j (see Equation **3-32**). Equation **3-34** can be interpreted as updating the mean μ_j of a cluster by weighting the observations by their probability of being part of that cluster. Equation **3-35** can likewise be interpreted as updating the covariance matrix $\boldsymbol{S}_j$ of a cluster by weighting the observations by their probabilities of being part of that cluster.

Initiation of the EM algorithm requires the number of clusters r and the initial parameter estimates ($\boldsymbol{\theta}^t$) to be specified. Determining the number of clusters objectively involves fitting the data to a plausible number of components and thereafter selecting the best fitting model. The maximum number of clusters $r_{\max}$ to be considered so as to bound the search space is determined by an empirical relationship given by Bozdooan (1994) as:

$$r_{max} = n^{0.3} \qquad \textbf{3-36}$$

Here, n refers to the number of observations.

Initialization and convergence of EM

Since the EM algorithm iterates between finding the clusters and responsibilities for each observation, the EM can therefore either be initialized using the responsibilities or cluster assignments from which the initial parameter estimates ($\boldsymbol{\theta}^t$) can be deduced. The initial cluster assignments can be done by randomly assigning observations to the clusters or by using k-means clustering (Hartigan and Wong, 1979) algorithm (among other approaches) which is a more effective approach. The k-means clustering algorithm basically groups *n* observations into *k* clusters (which is *r* clusters in this case) in which each observation belongs to the cluster with the closest mean. (Generally, good estimates for the covariance matrices would be the within-cluster covariances, and that for the mixing weights would be the fractions of data points belonging to each cluster.)

The EM algorithm is basically said to converge (locally) when there is no change in the previous and current iteration values for the parameters estimates. Since this is not always achieved, a tolerance level (τ) is defined such that any difference of the previous and current estimates of the parameters are deemed not significant when they are below the value of τ. Convergence can, therefore, be said to be achieved in such case.

Apart from specifying the number of clusters, the covariance structures are a major concern in adequately describing the data to ensure a good fit. The covariance structures considered are discussed next.

3.5.2.3 Covariance structure model

As with the number of clusters, the covariance structures of the GMM component can take a number of different shapes, volumes (defined by the eigenvalues of the covariance matrix) and orientations (defined by the eigenvectors of the covariance matrix) as shown in the earlier work of Bozdooan (1994) and subsequent investigation by (Celeux and Govaert, 1995). The general

covariance structure type is the full covariance structure which allows the variation of the ellipsoids in terms of all the axes as well as the volume and orientations. For a more parsimonious model, the diagonal covariance matrices are desired. In contrast to full covariance matrices which indicate that the variables are correlated, diagonal covariance matrices allow for uncorrelated variables. The correlation, therefore, places no restriction on the elliptical orientations of the full covariance matrices, while the major and minor axes of the ellipsoids of the diagonal covariance matrices have parallel or perpendicular axes (for e.g. the *x* and *y* axes in a 2-D case). Accordingly, the diagonal covariance matrices are more parsimonious than the full covariance matrices.

The restricted and unrestricted covariance types are the two generalizations of the orientations and volumes taken by the covariance matrices. While the restricted covariance matrices indicate that all cluster components are identical, the unrestricted covariance matrices may be unidentical. That is to say, the covariance matrices of the restricted case may be the same, as opposed to the unrestricted case, where they might differ. Figure 3.7 shows the various orientations and volumes for the diagonal and full matrices for cases in which they are unrestricted and restricted (for a two-dimensional case).

The diagonal and full covariance matrices are considered in this work. This raises the number of plausible covariance matrix types to four. For each covariance model mixture type, the number of parameters to be estimated is denoted by ɦ. The formulation of ɦ is as shown in Equation **3-37**.

$$ɦ = \gamma + ɣ \qquad \textbf{3-37}$$

γ is the number of parameters of the means $\boldsymbol{\mu}$ and mixing proportions ϱ whereas ɣ is the number of parameters in the covariance matrix. While the γ is same and equal to $rm + r - 1$ for all covariance types, ɣ values differ for the various model types and are presented in Table 1.

Table 1: The various covariance shapes and the respective estimated number of parameters.

Covariance	**Parameters** ($\boldsymbol{\theta}$)	ɣ
Diagonal-restricted	$\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}, \varrho_2, \boldsymbol{\mu}_2, \boldsymbol{S}, \dots, \varrho_r, \boldsymbol{\mu}_r, \boldsymbol{S}$	m
Diagonal-unrestricted	$\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}_1, \varrho_2, \boldsymbol{\mu}_2, \boldsymbol{S}_2, \dots, \varrho_r, \boldsymbol{\mu}_r, \boldsymbol{S}_r$	rm
Full-restricted	$\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}, \varrho_2, \boldsymbol{\mu}_2, \boldsymbol{S}, \dots, \varrho_r, \boldsymbol{\mu}_r, \boldsymbol{S}$	$m(m+1)/2$
Full-unrestricted	$\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}_1, \varrho_2, \boldsymbol{\mu}_2, \boldsymbol{S}_2, \dots, \varrho_r, \boldsymbol{\mu}_r, \boldsymbol{S}_r$	$rm(m+1)/2$

Taking the covariance structure types into consideration extends the clustering problem from the number of clusters to include the covariance structures as well. Therefore, a model selection criterion is required to select the best model for all the number of clusters and the respective plausible covariance models. Selection of the best model that describes the data is presented in the next section.

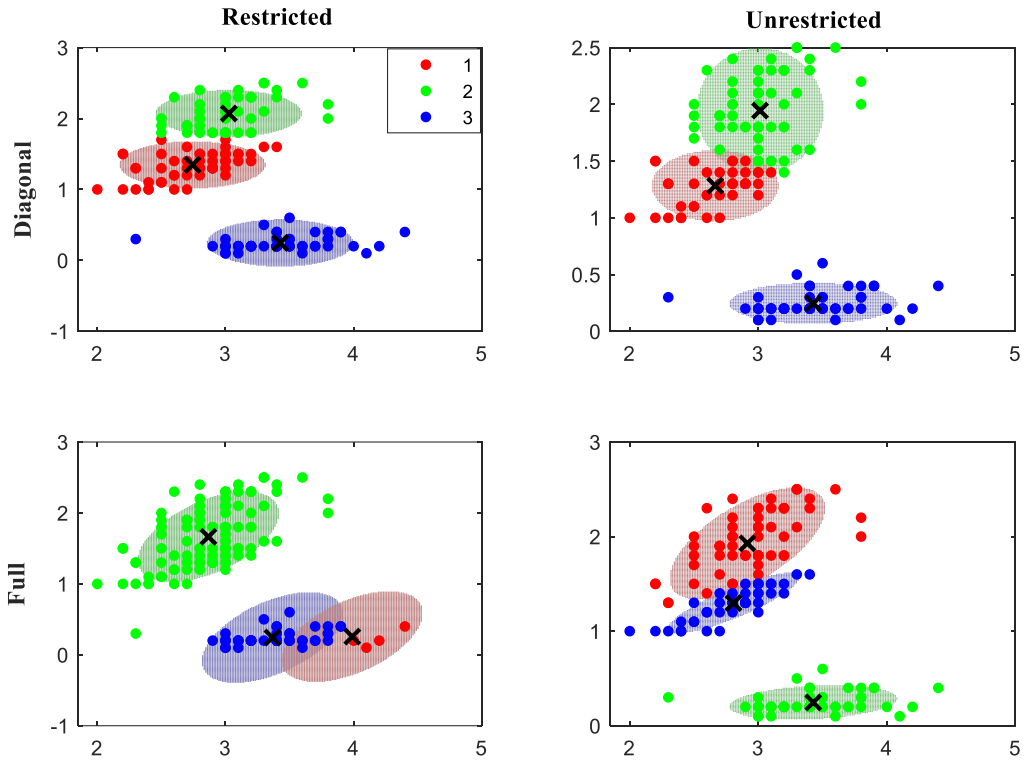


Figure 3.7: An example of different clustering results for the listed covariance structures.

3.5.2.4 Model Selection

In GMM and other clustering approaches, the number of clusters is generally not known. Selection of the most appropriate model then requires identifying and comparing suitable criteria. The aim of using such criteria is usually to provide a balance between goodness of fit of the data to the model and parsimony of the selected model.

Bayesian information criteria (BIC) (Schwarz, 1978) and Akaike's information criteria (AIC) (Akaike, 1973) are two popular model selection criteria that are used for model selection (Bozdogan, 2000).

The formulae for the AIC and BIC are given in Equations **3-38** and **3-39** respectively.

$$\mathrm{AIC} = -2\log L(\boldsymbol{\theta}^{*}|\boldsymbol{D}) + 2\mathfrak{h} \qquad \textbf{3-38}$$

$$\text{BIC} = -2\log L(\boldsymbol{\theta}^*|\boldsymbol{D}) + \text{ɦ}\log(n) \quad \textbf{3-39}$$

In the formulae, n is the number of observations, ɦ is the number of independent parameters to be estimated (see Equation **3-37**) and $\boldsymbol{\theta}^*$ denotes the EM approximates of the maximum likelihood estimate of the parameters.

As shown from the AIC and BIC formulations, both methods penalize the log-likelihood in the same manner, but the BIC penalizes model complexity more severely than the AIC. Consequently, the BIC tends to select simpler models that might underfit the data. The AIC, on the other hand, selects more complex models that might overfit the data (Posada and Buckley, 2004).

As mentioned by Yu (2011), deciding on the best criterion involves using prior experimental work which considers the prediction of the 'true' number of clusters by both methods. Accordingly, prior experimental work produced alternating results between the BIC and AIC as the best model selection criterion (more for BIC). This led to the introduction of the minimal best AIC and BIC criterion denoted by mAB in this work and determined next.

Let BIC_b represent the model with the lowest (best) BIC score. The corresponding AIC score for the same model is denoted as AIC_c. Accordingly, that for the lowest (best) AIC score and the corresponding BIC score are respectively denoted as AIC_b and BIC_c. The absolute distance between AIC_c and BIC_b is denoted by h_1 while that between AIC_b and BIC_c is denoted by h_2. The formulae for h_1 and h_2 are respectively shown in Equations **3-40** and **3-41**.

$$h_1 = |BIC_b - AIC_c| \quad \textbf{3-40}$$

$$h_2 = |AIC_b - BIC_c| \quad \textbf{3-41}$$

The best model selected thereafter is then the best model producing the minimum value of h_1 and h_2 values. That is if h_1 is the minimum value when compared to h_2, the model with the lowest BIC score (BIC_b) is selected.

3.5.3 Monitoring statistics and control limits

Monitoring using GMM involves the use of the estimated PDF (see Equation **3-25**). The PDF indicates how close an observation follows the obtained GMM from the training data. The negative logarithm of the PDF (NLPDF) values is used rather than the PDF values. This consequently transforms the product of potentially small PDF values (close to zero) into a sum

of logarithms, the result of which is more reliably distinguishable from zero in computation. An observation which comes from the same input space as the training data is therefore expected to have a lower NLPDF value as compared to a novel observation.

For every identified cluster, let $\mathbf{NLPDF}_{\alpha}$ denote the vector of critical values [$NLPDF_{\alpha 1}$, $NLPDF_{\alpha 2}$, …, $NLPDF_{\alpha r}$]. The critical values for each cluster are computed by taking an appropriate percentile of the training data monitoring statistics that belong to each cluster. The observations are assigned to the clusters using the responsibilities (i.e. the cluster with the highest responsibility value). (For example, taking the 99^{th} percentile for all observations assigned to cluster one as the critical value for cluster one.) The $\mathbf{NLPDF}_{\alpha}$ is then employed for local monitoring.

For a global monitoring scheme, $NLPDF_{\alpha}$ represents a single value which is taken as the critical value for all the observations (ignoring the cluster assignments). The implementation of local monitoring thresholds prevents the limitations associated with an approximation of thresholds over all clusters while the global limits the risk of misclassification.

3.5.4 Fault detection method

GMM fault detection procedure involves checking if an observation is unusual for the GMM model obtained by the training data. This is described by the probability of the observation given the model parameters.

The best GMM is selected using the mAB model selection criterion (see Section 3.5.2.4) from a number of plausible GMM models obtained by the training data for the various covariance structures options and a number of clusters. The monitoring statistics and critical values are computed from fit on training data. For a new observation, the NLPDF is computed for the observation and it is checked against its respective threshold (as determined by the highest responsibility). The observation is thereafter deemed to be abnormal if it is beyond the detection threshold. Like PCA, a *z* consecutive number of observations beyond the threshold warrants the trigger of an alarm.

The next section considers the PCA-based GMM which is a combination of PCA and GMM.

3.6 PCA-based GMM

3.6.1 Overview

PCA combined with GMM (Choi, Park and Lee, 2004; Yu, 2011) involves the use of PC scores $\boldsymbol{T}$ (see Section 3.2.2) as inputs for the GMM. The use of the scores combines the monitoring in a lower dimensional feature space (as implemented for PCA) and multimodal monitoring. This

work would illustrate that clustering using the scores can achieve better clustering results and reduce the risk of misclassification which is the main concern in employing local monitoring schemes.

The computation of all relevant statistics and model parameters for PCA-based GMM follow the same techniques as described for the GMM. The relevant Equations are updated for the PCA-based GMM such that the raw dataset (***D***) is replaced with the scores (***T***). For example, the probability function of the raw data (shown in Equation **3-25**) is updated from the raw data formulation to the scores formulation as shown in Equation **3-42**.

$$p(\boldsymbol{t}) = \sum_{j=1}^{r} \varrho_j g_j(\boldsymbol{t}|\boldsymbol{\mu}_j, \boldsymbol{S}_j) \qquad 3\text{-}42$$

3.6.2 Algorithm

The PCA-based GMM approach follows the algorithm described for the GMM but using the score as input data instead of the original observation vectors. The process of computing the scores from the raw data is described in PCA (in Section 3.2.2).

Once the scores are computed, the decided retained scores (or all the scores) are used as input for the GMM development following the subsequent procedure as that followed by the raw data in the GMM process shown in Section 3.5.2.

3.6.3 Monitoring statistics and control limits

The PCA-based GMM employs the same monitoring statistics and detection thresholds as shown for GMM in Section 3.5.3.

3.6.4 Fault detection

PCA-based GMM uses the same fault detection method as shown for GMM in Section 3.5.4.

In summary, this chapter presented the mathematical derivations for unimodal and adaptive unimodal monitoring approaches by considering PCA, RePCA, and MWPCA. The multimodal approaches are subsequently presented by considering GMM and combination of PCA with GMM.

The next chapter, therefore, considers the way the above-mentioned approaches are applied in this work as well as the presentation of APCA-based GMM which is an extension of APCA to GMM developed in this work.

CHAPTER 4: METHODOLOGY

This chapter presents the application of the techniques identified to address each stated objective. The applications of PCA and APCA approaches as presented. Also, that for GMM and PCA-based GMM is presented next. The last section considers a new approach which is APCA-based GMM. This approach involves a combination of the APCA and GMM which seeks to ensure monitoring a process using multiple models that are representative of a current process state.

4.1 Application of PCA

The offline and online application of PCA is shown in the subsequent steps.

Offline (Training)

1. Obtain training data $\boldsymbol{D} \in R^{n \times m}$.
2. Normalize the training data and retain the means and standard deviations of the process variables [see Equations **3-1** to **3-3**].
3. Compute the correlation matrix $\boldsymbol{C}$ of the normalized data $\boldsymbol{X} \in R^{n \times m}$ [see Equation **3-4**].
4. Compute the eigenvectors $\boldsymbol{P} \in R^{m \times m}$ and eigenvalues $\boldsymbol{\lambda} \in R^{1 \times m}$ of $\boldsymbol{C}$ [see Equation **3-5**].
5. Retain first (major) υ PCs $\hat{\boldsymbol{P}}$ and their corresponding variances $\hat{\boldsymbol{\lambda}}$ [see Equation **3-7**].
6. Compute the retained scores, $\hat{\boldsymbol{T}} \in R^{n \times \upsilon}$ [see Equation **3-6**].
7. Reconstruct the data from $\hat{\boldsymbol{T}}$ [see Equation **3-9**].
8. Calculate the squared prediction errors $\boldsymbol{q}$ and its detection threshold q_α of the reconstructed data $\hat{\boldsymbol{X}}$ [see Equations **3-11** and **3-14**].
9. Compute the modified Hotelling's T^2 statistics $\hat{\boldsymbol{t}}^2$ and its detection threshold $(\hat{t}^2)_\alpha$ [see Equations **3-12** and **3-13**].
10. Obtain the validation data $\boldsymbol{D_v} \in R^{\hat{n} \times m}$ (obtained by splitting the original data).
11. Normalize the validation data using the retained means and standard deviations from Step 2.
12. Compute the scores by projecting the normalized data $\boldsymbol{X_v} \in R^{\hat{n} \times m}$ on $\hat{\boldsymbol{P}}$ [$\hat{\boldsymbol{T}}_{\boldsymbol{v}} = \boldsymbol{X_v} \times \hat{\boldsymbol{P}}$].
13. Compute $\hat{\boldsymbol{t}}^2{}_{\boldsymbol{v}}$ using the retained PCs $\hat{\boldsymbol{P}}$ and eigenvalues $\hat{\boldsymbol{\lambda}}$.
14. Reconstruct the data and compute the SPE $\boldsymbol{q_v}$ [see Equations **3-9** and Equations **3-11**].
15. Model selection: Test how the derived model parameters ($\hat{\boldsymbol{P}}$ and $\hat{\boldsymbol{\lambda}}$) and thresholds ($q_\alpha, (\hat{t}^2)_\alpha$) perform for unseen NOC data by analyzing the validation data monitoring statistics. Return to Step 5 and adjust the number of retained PCs and significance

thresholds if needed. If the model performs poorly for the validation data, it implies it will not do well when employed online.

16. Save model for online deployment.

Online (Deployment)

1. Obtain new observation $\boldsymbol{d}$.
2. Normalize the sample using the retained means and standard deviations from the training stage [see Equations **3-1** to **3-3**].
3. Compute the scores by projecting the normalized data $\boldsymbol{x} \in R^{1\times m}$ onto $\widehat{\boldsymbol{P}}$ [$\hat{t} = \boldsymbol{x}\widehat{\boldsymbol{P}}$].
4. Reconstruct the datum [$\widehat{\boldsymbol{x}} = \boldsymbol{x} - \widehat{\boldsymbol{x}}$].
5. Compute the SPE q of the reconstructed datum $\widehat{\boldsymbol{x}}$ [see Equation **3-11**].
6. Compute $\hat{t}^2$ using the retained PCs and eigenvalues [see Equation **3-12**].
7. If a z consecutive number of $\hat{t}^2$ and/or q are beyond their respective detection thresholds, trigger an alarm.

The next section considers the application of RePCA for fault detection.

4.1.1 Application of RePCA

The application involves making an assumption that the current observation is normal and thereby computing updated window parameters, monitoring statistics and critical values.

To assess whether a new observation is normal or not, the effect of how the addition of the new observation perturbs the existing PCA model is then analysed by computing pseudo-updated statistics as listed below:

Let Norm ($b \mid w, \mu, \sigma$) represent the normalization of some variable w using some mean μ and some standard deviation σ produce some variable b as shown below:

$$b = \frac{w - \mu}{\sigma} \qquad \textbf{4-1}$$

Also, let Proj ($b \mid w, a$) denote the projection of some variable w onto some variable a to produce some variable b as shown below:

$$b = w \times a \qquad \textbf{4-2}$$

1. Compute a new normalized datum by Norm $((\boldsymbol{x}_{k+1})_{new} \mid \boldsymbol{d}_{k+1}, \boldsymbol{\mu}_{k+1}, \boldsymbol{\sigma}_{k+1})$
2. Compute an old normalized datum by Norm $((\boldsymbol{x}_{k+1})_{old} \mid \boldsymbol{d}_{k+1}, \boldsymbol{\mu}_{k}, \boldsymbol{\sigma}_{k})$
3. Compute a new score by Proj $(\hat{\boldsymbol{t}}_{new} \mid (\boldsymbol{x}_{k+1})_{new}, (\widehat{\boldsymbol{P}})_{k+1})$

4. Using $(\boldsymbol{x}_{k+1})_{int} = (\boldsymbol{x}_{k+1})_{new}$, compute an intermediate score by Proj $(\hat{\boldsymbol{t}}_{int} | (\boldsymbol{x}_{k+1})_{int}, (\hat{\boldsymbol{P}})_k)$
5. Compute old score by Proj $(\hat{\boldsymbol{t}}_{old} | (\boldsymbol{x}_{k+1})_{old}, (\hat{\boldsymbol{P}})_k)$
6. Compute the reconstructed data for all the scores by Proj $((\hat{\boldsymbol{x}}_{k+1})_{new} | \hat{\boldsymbol{t}}_{new}, (\hat{\boldsymbol{P}})_{k+1}^T)$, Proj $((\hat{\boldsymbol{x}}_{k+1})_{int} | \hat{\boldsymbol{t}}_{int}, (\hat{\boldsymbol{P}})_k^T)$ and Proj $((\hat{\boldsymbol{x}}_{k+1})_{old} | \hat{\boldsymbol{t}}_{old}, (\hat{\boldsymbol{P}})_k^T)$
7. Compute the reconstruction errors for all the reconstructed data. The reconstruction errors for $(\hat{\boldsymbol{x}}_{k+1})_{new}$, $(\hat{\boldsymbol{x}}_{k+1})_{int}$ and $(\hat{\boldsymbol{x}}_{k+1})_{old}$ are calculated as shown in Equation **4-3**.

$$(\boldsymbol{e}_{k+1})_\bullet = (\boldsymbol{x}_{k+1})_\bullet - (\hat{\boldsymbol{x}}_{k+1})_\bullet \qquad \textbf{4-3}$$

Above, • denotes *new*, *old*, or *int*.

8. Compute the squared prediction errors for all the reconstructed data as shown in Equation **4-4**.

$$(q_{k+1})_\bullet = (\boldsymbol{e}_{k+1})_\bullet \cdot (\boldsymbol{e}_{k+1})_\bullet \qquad \textbf{4-4}$$

Above, • denotes *new*, *old*, or *int*.

9. Compute the modified Hotelling's T^2 statistic for all the scores. Using: $\boldsymbol{\Lambda}_{new}=\boldsymbol{\Lambda}_{k+1}$, $\boldsymbol{\Lambda}_{int}=\boldsymbol{\Lambda}_{old}=\boldsymbol{\Lambda}_k$, $\boldsymbol{P}_{new}=(\hat{\boldsymbol{P}})_{k+1}$, and $\boldsymbol{P}_{int}=\boldsymbol{P}_{old}=(\hat{\boldsymbol{P}})_k$, the values for $\hat{t}_{new}$, $\hat{t}_{old}$ and $\hat{t}_{int}$ are calculated as shown in Equation **4-5**.

$$((\hat{t}^2)_{k+1})_\bullet = (\boldsymbol{x}_{k+1})_\bullet \boldsymbol{P}_\bullet \boldsymbol{\Lambda}_\bullet^{-1} \boldsymbol{P}_\bullet^T (\boldsymbol{x}_{k+1}^T)_\bullet \qquad \textbf{4-5}$$

where, $\boldsymbol{\Lambda}_{k+1}$ and $\boldsymbol{\Lambda}_k$ are respectively the diagonal matrices of $(\hat{\boldsymbol{\lambda}})_{k+1}$ and $(\hat{\boldsymbol{\lambda}})_k$ values; and • denotes *new*, *old*, or *int*.

4.1.1.1 Model update and condition to update

1. If for *z* consecutive number of $((\hat{t}^2)_{k+1})_{int} > ((\hat{t}^2)_\alpha)_k$ or $(q_{k+1})_{int} > (q_\alpha)_k$, trigger an alarm. Also, set the monitoring statistics of the current observation to $((\hat{t}^2)_{k+1})_{old}$ and $(q_{k+1})_{old}$. Set the critical values to $(q)_k$ and $((\hat{t}^2)_\alpha)_k$. $\boldsymbol{D}_k = \boldsymbol{D}_k$ for application to next observation.
2. Else, if for *z* consecutive number of $((\hat{t}^2)_{k+1})_{int} < ((\hat{t}^2)_\alpha)_k$ or $(q_{k+1})_{int} < (q_\alpha)_k$, set the monitoring statistics of the current observation to $(q_{k+1})_{new}$ and $((\hat{t}^2)_{k+1})_{new}$. Set the critical values to $(q)_{k+1}$ and $((\hat{t}^2)_\alpha)_{k+1}$. $\boldsymbol{D}_k = \boldsymbol{D}_{k+1}$ for application to next observation.

The subsequent outlined offline and online procedure of the application puts the method discussed above into perspective.

Offline (Training)

1. Obtain initial window $\boldsymbol{D}_k \epsilon R^{n \times m}$.
2. Run PCA on $\boldsymbol{D}_k$.

Online (Deployment)

1. Obtain new observation $\boldsymbol{d}_{k+1}$.
2. Compute the updated means $\boldsymbol{\mu}_{k+1}$, standard deviations $\boldsymbol{\sigma}_{k+1}$ and correlation matrix $\boldsymbol{C}_{k+1}$ of the updated window $\boldsymbol{D}_{k+1} \epsilon R^{(n+1) \times m}$ [see Equations **3-15** to **3-18**].
3. Compute and retain the PCs $(\widehat{\boldsymbol{P}})_{k+1}$ and variances $(\hat{\boldsymbol{\lambda}})_{k+1}$ of $\boldsymbol{C}_{k+1}$.
4. Compute the retained scores $(\widehat{\boldsymbol{T}})_{k+1}$ [see Equation **3-6**].
5. Compute the modified Hotelling's T^2 statistic $(\hat{\boldsymbol{t}}^2)_{k+1}$ and its critical value $((\hat{t}^2)_\alpha)_{k+1}$.
6. Compute the intermediate, updated and down-dated statistics [see Steps to 1 to 8 of Section 4.1.1].
7. Update model parameters if appropriate (using Steps 1 to 2 of the model update conditions presented in Section 4.1.1.1).

Hyperparameter tuning is done by using validation data (and test data if available) and following the online deployment procedure to test how the derived model parameters and thresholds perform for unseen NOC data and faults. This is done by verifying with the validation data monitoring statistics and readjustments of the retained parameters and thresholds if possible.

The next section, therefore, considers the application of MWPCA which is the other considered APCA approach for fault detection.

4.1.2 Application of MWPCA

The application of MWPCA for the updated window follows the same pattern as that for RePCA. The main differences between the two are the procedure of updating form the previous window to the next window.

The online procedure outlined below provide the guide for updating the initial window.

Offline (Training)

The offline application for MWPCA follows the same procedure as listed in the offline application for RePCA in Section 4.1.1.

Online (Deployment)

The online application of RePCA holds for that of the MWPCA once the updated window is created. The procedure of updating from the initial window is listed as follows:

1. Obtain new observation $\boldsymbol{d}_{k+1}$.
2. Compute the means $\boldsymbol{\mu}_{k-1}$ and standard deviations $\boldsymbol{\sigma}_{k-1}$ and correlation matrix $\boldsymbol{C}_{k-1}$ of the down-dated window $\boldsymbol{D}_k \epsilon R^{(n-1)\times m}$ [see Equations **3-19** to **3-21**].
3. Compute the means $\boldsymbol{\mu}_{k+1}$ and standard deviations $\boldsymbol{\sigma}_{k+1}$ and correlation matrix $\boldsymbol{C}_{k+1}$ of the updated window $\boldsymbol{D}_{k+1} \epsilon R^{n\times m}$ [see Equations **3-22** to **3-24**].

Once the updated window is created, Steps 1 to 7 of the online application of RePCA as listed in Section 4.1.1 is followed. Hyperparameter tuning is done using the same approach as that for RePCA.

The next section considers the application GMM which is a multimodal approach for fault detection.

4.1.3 Application of GMM

GMM application involves the use of training and validation methods for the offline stage before online deployment. The training data is used as an input for the GMM development. The maximum likelihood parameter estimates are thereafter approximated using the EM algorithm for the various covariance structures and cluster numbers (see Sections 3.5.2.3 and 3.5.2.4). The best model is then selected using the mAB criterion (see Section 3.5.2.4) and saved for online deployment after the generalisability is checked with a validation data and model parameter readjustments made if possible.

For a new observation, the NLPDF value is computed using the saved model parameters and checked against the detection thresholds and thereafter flagged as normal or abnormal. The sequence of implementation is shown in the subsequent offline and online procedure.

Offline (Training)

1. Obtain training data $\boldsymbol{D} \epsilon R^{n\times m}$.
2. Compute the maximum plausible cluster number r_{max} [Equation **3-36**].

3. Approximate the maximum likelihood estimates $\boldsymbol{\theta}^*$ for the various covariance structures coupled with the number of clusters from $r = 1$ to r_{max} using the EM algorithm [see Section 3.5.2.2].
4. Compute and retain the best GMM using the mAB method [see Section. 3.5.2.4].
5. Compute the monitoring statistics **NLPDF** and the global critical value NLPDF_α and the individual (local) critical values $\mathbf{NLPDF}_{\boldsymbol{\alpha}}$ [see Section 3.5.4].
6. Obtain the validation data $\boldsymbol{D_v} \in R^{n \times m}$ (obtained by splitting the original data).
7. Compute the **NLPDF** using the GMM.
8. Test the generalisability of the derived model parameters and the detection thresholds by verifying with the validation data monitoring statistics and readjust the model parameters and thresholds if possible.
9. Save model for online deployment.

Online (Deployment)

1. Obtain new observation $\boldsymbol{d}$.
2. Compute the NLPDF and cluster membership for the new observation.
3. For local detection thresholds, if a z consecutive number of NLPDF values are beyond its respective cluster detection thresholds, trigger an alarm.
4. For a global detection threshold, if a z consecutive number of NLPDF values are beyond the detection threshold NLPDF_α, trigger an alarm.

4.1.4 Application of PCA-based GMM

The initial stages involve computing the scores of the training data as outlined in PCA (Section 4.1). The computed retained scores are used as an input for fitting the GMM.

For a new observation, the score is computed using the retained normalization and PCA model parameters. The NLPDF of the score is then computed using the GMM and checked against the respective global and local limits in a similar procedure to those listed for GMM online application for fault detection (see Section 4.1.3). The sequence of implementation is shown in the subsequent offline and online procedure.

Offline (Training)

1. Obtain the training data $\boldsymbol{D} \in R^{n \times m}$.
2. Follow Steps 2 to 5 outlined in PCA offline to compute the retained scores $\hat{\boldsymbol{T}}$.
3. Follow Steps 2 to 5 outlined in GMM offline on $\hat{\boldsymbol{T}}$ to compute the monitoring statistics and critical value.

4. Obtain the validation data $\boldsymbol{D_v} \epsilon R^{\acute{n} \times m}$ (obtained by splitting the original data).
5. Follow Steps 11 to 12 outlined in PCA to compute the scores.
5. Follow Step 8 of GMM to test the generalisability of the derived model parameters and readjust the model parameters and thresholds if possible.
6. Save model for online deployment.

Online (Deployment)

1. Obtain new observation $\boldsymbol{d}$.
2. Follow Steps 2 to 3 outlined in PCA online to compute the score $\hat{\boldsymbol{t}}$ corresponding to the retained score for the new observation.
3. Compute the NLPDF and cluster membership.
4. For local detection thresholds, if a z consecutive number of NLPDF are beyond the respective cluster detection thresholds, trigger an alarm.
5. For a global detection threshold, if a z consecutive number of NLPDF are beyond the detection threshold NLPDF_{α}, trigger an alarm.

The next section considers APCA-based GMM which is the combination of APCA and GMM.

4.2 APCA-based GMM

4.2.1 Overview

APCA-based GMM combines APCA methods with multimodal methods to make provision for slow and natural process changes that occur in a multimodal process. Like APCA, the initial training data window is periodically augmented with new NOC data. This consequently changes the model parameters of the training window and the respective scores of the PCs computed thereafter.

Although the scores of the training data are updated at each interval, the GMM training is only done once with a reservoir block of data $\boldsymbol{D}_b \epsilon R^{c \times m}$ when it becomes available. The computation of the probability values of the updated scores is therefore done with the initial GMM until a number observations c is available. The constant c is a hyperparameter which describes an amount of data that can cause variation in the process and consequently the GMM. Moreover, use of a block-wise update keeps computational load reasonable.

$\boldsymbol{D}_b \epsilon R^{c \times m}$ is therefore obtained by accumulating new observations that exhibit NOC data over time. For a time instance, let $\boldsymbol{D}_k \epsilon R^{n \times m}$ be the data window (initially set at the outset). Let $\boldsymbol{D}_1 \epsilon R^{c \times m}$ represent the oldest c observations of $\boldsymbol{D}_k$. The data window after removing $\boldsymbol{D}_1$ from

$\boldsymbol{D}_k$ is $\boldsymbol{D}_2 = [\boldsymbol{d}_{c+1}, \boldsymbol{d}_{c+2}, \ldots, \boldsymbol{d}_n]$. The updated window matrix is $\boldsymbol{D}_{k+1} = [\boldsymbol{D}_2, \boldsymbol{D}_b]$. The updated window is then used to create a new GMM which is kept until the next block update. Figure 4.1 provides a conceptual diagram of how the PCA model is updated for new observations while Figure 4.2 shows how the initial GMM model is updated with the new block of observations. (The illustrations are made in the input data space while implementations are done in the reduced feature space.)

The updated window is used to compute a new GMM with updated parameters and model statistics. The model update procedure for the scores and the GMM are presented in the next section.

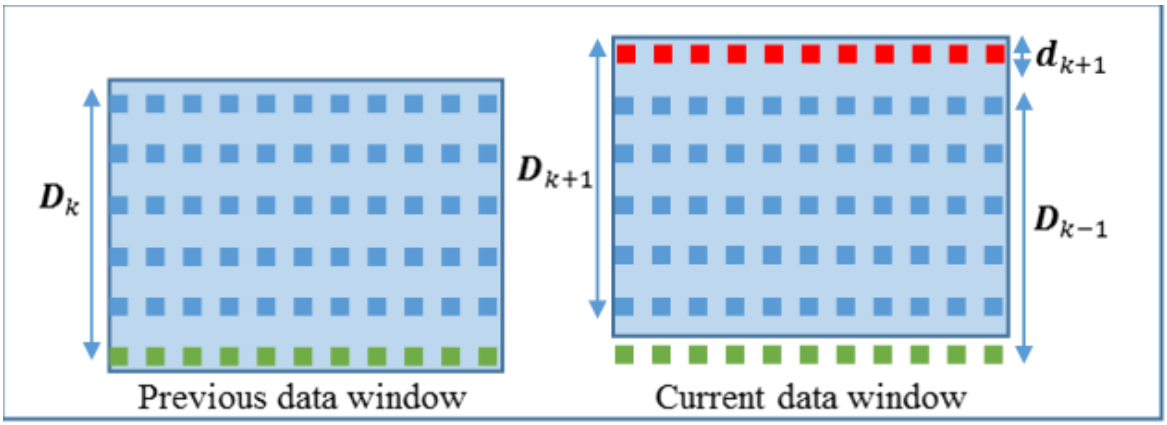


Figure 4.1: Sample-wise update of the PCA model for new observations.

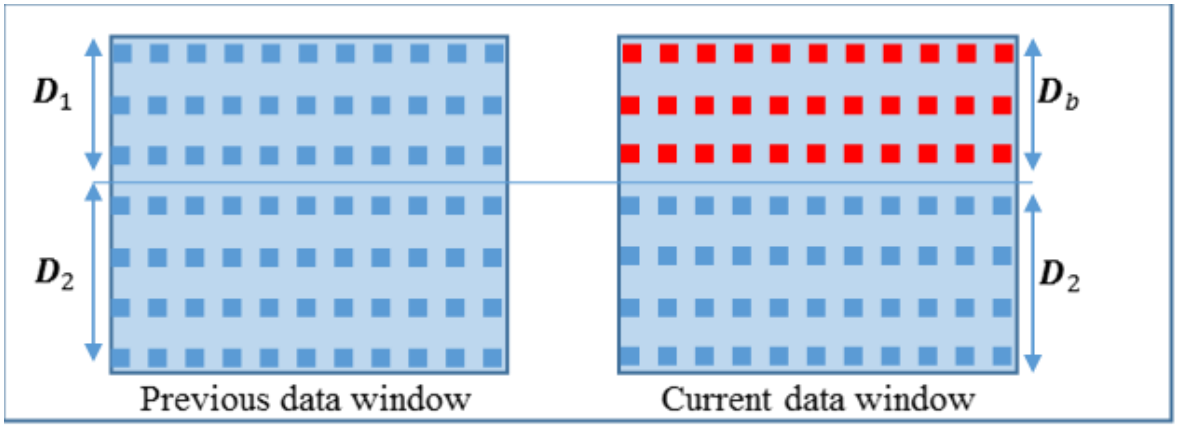


Figure 4.2: Block-wise update of the GMM for new observations.

4.2.2 Algorithm

Adaptation of the PCA model parameters of the updated window follows the procedure outlined in MWPCA algorithm in Section 3.4.2 to compute the updated scores $\boldsymbol{T}_{k+1}$. The retained scores $(\hat{\boldsymbol{T}})_{k+1}$ are then used to compute new monitoring statistics using the initial GMM (this is done by computing the probability values of each observation score in $(\hat{\boldsymbol{T}})_{k+1}$. The

monitoring statistics and thresholds $(\mathbf{NLPDF}_{\boldsymbol{\alpha}})_{k+1}$ are consequently updated as outlined in Section 3.6.2.

The GMM is only updated once there are *c* new observations added to the initial window that was used to create the GMM. New GMM parameters are then computed to replace the pre-existing GMM model parameters. This updated model will then serve as the current model for new observations until another GMM is created.

4.2.3 Monitoring statistics and control limits

APCA-based GMM employs the same monitoring statistics and detection thresholds as the GMM in Section 3.5.3 but with changing score matrix.

4.2.4 Fault detection

APCA-based GMM employs the same fault detection procedure as shown for the GMM in Section 3.5.4.

4.2.5 Application

The initial training data is normalized and the retained scores are computed. Similar to PCA-based GMM, the retained scores are used in the GMM development. The NLPDF values for each retained score in the training data are computed as well as the individual cluster posterior probabilities. The individual cluster posterior probabilities are thereafter used to assign cluster membership to each observation. The current vector of critical values for the respective clusters are computed and denoted as $(\mathbf{NLPDF}_{\boldsymbol{\alpha}})_k$. These serve as the critical values to be used as in local monitoring thresholds, where each observation is monitored by its corresponding cluster threshold. The current overall critical value $(\mathrm{NLPDF}_{\alpha})_k$ is computed over all the observations in all clusters, which is used in the global monitoring scheme.

For a new observation, the initial window is updated using the MWPCA procedure and new normalization parameters and scores are computed. The updated retained scores $(\widehat{\boldsymbol{T}})_{k+1}$ of the updated window are used to compute updated thresholds of the monitoring statistic and denoted as $(\mathbf{NLPDF}_{\boldsymbol{\alpha}})_{k+1}$. This is done by calculating the probability values for each score in $(\widehat{\boldsymbol{T}})_{k+1}$ using the pre-existing GMM and assigning cluster memberships using the responsibilities (for local monitoring). An appropriate percentile is taken over all the clusters for the global threshold and the individual clusters for the local thresholds (see Section 3.5.3)

The effect of how the addition of the new observation perturbs the existing PCA model is then analysed by computing pseudo-updated statistics as listed below:

1. Compute a new normalized datum by Norm $((\boldsymbol{x}_{k+1})_{new} \mid \boldsymbol{d}_{k+1}, \boldsymbol{\mu}_{k+1}, \boldsymbol{\sigma}_{k+1})$
2. Compute an old normalized datum by Norm $((\boldsymbol{x}_{k+1})_{old} \mid \boldsymbol{d}_{k+1}, \boldsymbol{\mu}_{k}, \boldsymbol{\sigma}_{k})$
3. Compute a new score by Proj $(\hat{\boldsymbol{t}}_{new} \mid (\boldsymbol{x}_{k+1})_{new}, (\hat{\boldsymbol{P}})_{k+1})$
4. Using $(\boldsymbol{x}_{k+1})_{int} = (\boldsymbol{x}_{k+1})_{new}$, compute an intermediate score by Proj $(\hat{\boldsymbol{t}}_{int} \mid (\boldsymbol{x}_{k+1})_{int}, (\hat{\boldsymbol{P}})_{k})$
5. Compute old score by Proj $(\hat{\boldsymbol{t}}_{old} \mid (\boldsymbol{x}_{k+1})_{old}, (\hat{\boldsymbol{P}})_{k})$
6. Compute the NLPDF for all the scores (old, new and intermediate). Denoting new, int, or old by $\bullet$, the NLPDF for $\hat{\boldsymbol{t}}_{new}$, $\hat{\boldsymbol{t}}_{int}$ and $\hat{\boldsymbol{t}}_{old}$ are calculated as:

$$\text{NLPDF}(\hat{\boldsymbol{t}}_{\bullet}) = -\log \sum_{j=1}^{r} \varrho_j g_j \left(\hat{\boldsymbol{t}}_{\bullet} \mid \boldsymbol{\mu}_j, \boldsymbol{S}_j \right) \qquad \textbf{4-6}$$

7. Compute the responsibilities of each cluster to assign cluster membership. The responsibilities for $\hat{\boldsymbol{t}}_{new}$, $\hat{\boldsymbol{t}}_{int}$ and $\hat{\boldsymbol{t}}_{old}$ are respectively calculated as:

$$p(g_j \mid \hat{\boldsymbol{t}}_{\bullet}) = \frac{\varrho_j g_j(\hat{\boldsymbol{t}}_{\bullet} \mid \boldsymbol{\mu}_j, \boldsymbol{S}_j)}{\sum_{p=1}^{r} \varrho_p g_p(\hat{\boldsymbol{t}}_{\bullet} \mid \boldsymbol{\mu}_p, \boldsymbol{S}_p)} \qquad \textbf{4-7}$$

Here, $\bullet$ denotes new, int, or old.

4.2.5.1 Model update and condition to update

1. For local monitoring thresholds, if a z consecutive number of NLPDF$(\hat{\boldsymbol{t}}_{int})$ are greater than their respective cluster thresholds, trigger an alarm. Set the monitoring statistics of the current observation to NLPDF$(\hat{\boldsymbol{t}}_{old})$. $\boldsymbol{D}_k = \boldsymbol{D}_k$ for application to next observation.
2. Else, if a z consecutive number of NLPDF$(\hat{\boldsymbol{t}}_{int})$ are lesser than their respective cluster thresholds, set the monitoring statistics of the current observation to NLPDF$(\hat{\boldsymbol{t}}_{new})$. Set the critical values to $(\textbf{NLPDF}_{\alpha})_{k+1}$. Append $\boldsymbol{d}_{k+1}$ to $\boldsymbol{D}_b$. $\boldsymbol{D}_k = \boldsymbol{D}_{k+1}$ for application to next observation.
3. For a global monitoring threshold, if a z consecutive number of NLPDF$(\hat{\boldsymbol{t}}_{int})$ are greater than the detection threshold $(\text{NLPDF}_{\alpha})_k$, trigger an alarm. Set the monitoring statistics of the current observation to NLPDF$(\hat{\boldsymbol{t}}_{old})$. $\boldsymbol{D}_k = \boldsymbol{D}_k$ for application to next observation.
4. Else, if a z consecutive number of NLPDF$(\hat{\boldsymbol{t}}_{int})$ are lesser than the detection threshold $(\text{NLPDF}_{\alpha})_k$ set the monitoring statistics of the current observation to NLPDF$(\hat{\boldsymbol{t}}_{new})$. Set the critical value to $(\text{NLPDF}_{\alpha})_{k+1}$. Append $\boldsymbol{d}_{k+1}$ to $\boldsymbol{D}_b$. $\boldsymbol{D}_k = \boldsymbol{D}_{k+1}$ for application to next observation.

5. Size the accumulated data $\boldsymbol{D_b}$. If the number of observations equal *c*, set the current window $\boldsymbol{D}_{k+1}$ as an initial window $\boldsymbol{D}_k$ and compute new GMM. The updated window is applied to a new observation as an initial window. Also set $\boldsymbol{D_b}$ to an empty set. New NOC data are appended to $\boldsymbol{D_b}$ until a new GMM is created.

The sequence of implementation is shown in the subsequent offline and online procedure.

Offline (Training)

1. Obtain training data $\boldsymbol{D}_k \in R^{n\times m}$.
2. Compute the maximum plausible cluster number r_{max} [see Equation **3-36**].
3. Run PCA on $\boldsymbol{D}_k$ to produce $(\hat{\boldsymbol{T}})_k$**.**
4. Use $(\hat{\boldsymbol{T}})_k$ as input for fitting the GMM.
5. Approximate the maximum likelihood estimates $\boldsymbol{\theta}^*$ for the various covariance structures coupled with the number of clusters (with a maximum value of r_{max}) using the EM algorithm [see Section 3.5.2.2].
6. Compute and retain the best GMM using the mAB method [see Section. 3.5.2.4].
7. Test the generalisability of the derived model parameters and the detection thresholds by verifying with the validation data monitoring statistics and readjust the model parameters and thresholds if possible.
8. Save model for online deployment.

Online (Deployment)

6. Obtain new observation $\boldsymbol{d}_{k+1}$.
7. Compute the means $\boldsymbol{\mu}_{k-1}$, standard deviations $\boldsymbol{\sigma}_{k-1}$ and correlation matrix $\boldsymbol{C}_{k-1}$ of the down-dated window $\boldsymbol{D}_k \in R^{(n-1)\times m}$ [see Equations **3-19** to **3-21**].
8. Compute the updated means $\boldsymbol{\mu}_{k+1}$, standard deviations $\boldsymbol{\sigma}_{k+1}$and correlation matrix $\boldsymbol{C}_{k+1}$ of the updated window $\boldsymbol{D}_{k+1} \in R^{n\times m}$ [see Equations **3-22** to **3-24**].
9. Compute and retain the PCs $(\hat{\boldsymbol{P}})_{k+1}$ and variances $(\hat{\boldsymbol{\lambda}})_{k+1}$ of $\boldsymbol{C}_{k+1}$ [Equation **3-8**].
10. Compute the retained scores $(\hat{\boldsymbol{T}})_{k+1}$ of the new window [Equation **3-6**].
11. Compute the monitoring statistic $(\mathbf{NLPDF})_{k+1}$ and cluster membership of each $\boldsymbol{t}$ in $(\hat{\boldsymbol{T}})_{k+1}$ using the existing.
12. Compute the updated global critical value $(\mathrm{NLPDF}_\alpha)_{k+1}$ and local critical values$(\mathbf{NLPDF}_\alpha)_{k+1}$.
13. Compute the intermediate, updated and down-dated statistics [Equations **4-8** to **4-9**].

14. Update model parameters if appropriate (using the stated Steps 1 to 5 of the model update condition).

Hyperparameter tuning is done using the same approach as that for RePCA (in Section 4.1.1).

In summary, the applications of the various approaches considered were presented with the outline of the training and deployment steps provided.

The next chapter considers the fault detection results for and the simulated case studies.

CHAPTER 5: RESULTS

The chapter begins with a brief introduction to the case studies (the Tennessee Eastman process and the non-isothermal CSTR process) used in our investigations. Next, an overview of the monitoring performance metrics used is given as an introduction to the presentation format used in this thesis.

Section 5.2 then considers the need for APCA monitoring. It was established that APCA monitoring provides better results than PCA, which confirms previous studies. Further analysis then considers the optimal implementation of APCA. This produced the best results for MWPCA, again confirming previous work. The proposed monitoring approach is also compared against pre-existing techniques to verify its superiority, which is novel work.

Section 5.3 discusses the limitations of PCA methods in handling multimodal data. We confirm previous studies establishing GMM as a better alternative. An in-depth analysis was carried out to provide the best monitoring approach for GMM, which is novel work.

Section 5.4, the final part, provides an extension of APCA to GMM after the need for monitoring models that can handle both multimodal data and drift behaviour was established, which is novel work. APCA-based GMM performed best at overall monitoring performance for all considered cases.

5.1 Case studies

Two case studies are used in this work: a non-isothermal continuous stirred tank reactor (CSTR) process, as well as the Tennessee Eastman (TE) process which is a well-known benchmark process monitoring problem. While the pre-generated dataset by Russell, Chiang, and Braatz (2000) is used for the TE process, various datasets are generated from a simulator for the CSTR process for specific purposes. While the TE process serves as the benchmark process, the CSTR process provides the prospect to simulate a variety of abnormal situations at various magnitudes and combinations, enabling robustness and sensitivity testing of process monitoring algorithms. These are described in detail under the relevant sections. For now, brief descriptions of the two processes are presented in the next sections.

5.1.1 Non-isothermal CSTR model

The CSTR process as shown in Figure 5.1 is a non-isothermal system with a single reactor tank. The reactor takes an input of premixed reactants (reactant *a* and solvent *s*) and produces the product *b*. The reaction rate is a first order reaction based on the assumptions that there is perfect mixing, physical properties remain constant, and there is negligible shaft work. The process is

an exothermic one with a coolant maintaining the desired temperature. The reaction rate r is presented in Equation **5-1**.

$$r = k_o e^{-E/RT} C \qquad \textbf{5-1}$$

Here, k_o is the pre-exponential kinetic constant; E is the activation energy for the reaction; R is the ideal gas constant; C is the concentration and T is temperature.

The behaviour of the non-stationary process is described by the mass balance of the reactant a, shown in Equation **5-2**, and the total energy balance as shown in Equation **5-3**.

$$\frac{dC}{dt} = \frac{F}{V}(C_i - C) - r \qquad \textbf{5-2}$$

$$\frac{dT}{dt} = \frac{F}{V}(T_i - T) - \frac{qF_c{}^{b+1}}{V\rho c_p(F_c + qF_c{}^{b}/2\rho_c c_{pc})}(T_i - T_c) - (\Delta H_r)Vr \qquad \textbf{5-3}$$

Here, V is the volume of reaction mixture in the tank, F and F_c are the flow rates of the inlet reaction mixture and the coolant respectively, and ρ and ρ_c are the densities of the inlet reaction mixture and the coolant respectively. The heat of reaction is denoted by ΔH_r, T_i and T are respectively the reactor inlet temperature and outlet temperature. The constants q and b are constants of the empirical relationship ($UA = qF_c{}^{b}$), that relates the heat transfer coefficient (UA) to the coolant flow rate. Parameters c_p and c_{pc} are the specific heat capacities of the reaction mixture and the coolant respectively.

The concentration of the inlet reaction mixture C_i is given as an average of the concentrations of solvent s and reactant a in Equation **5-4**.

$$C_i = \frac{F_a C_a + F_s C_s}{F_a + F_s} \qquad \textbf{5-4}$$

Here, F_a and F_s are respectively the flow rates of reactant a and solvent s. C_s is the concentration of solvent s. Note that $F_a + F_s = F$.

The measured variables are the input variables F_a, F_s, F_c, C_a, C_s, T_c, T_i and the process output variables T and C. For a closed loop process, the temperature of the reactor T is controlled by manipulating F_c. The concentration is also controlled by manipulating F_a. A detailed description of the reactor and reaction parameters as well as the Simulink model and simulator is presented in Appendix A.

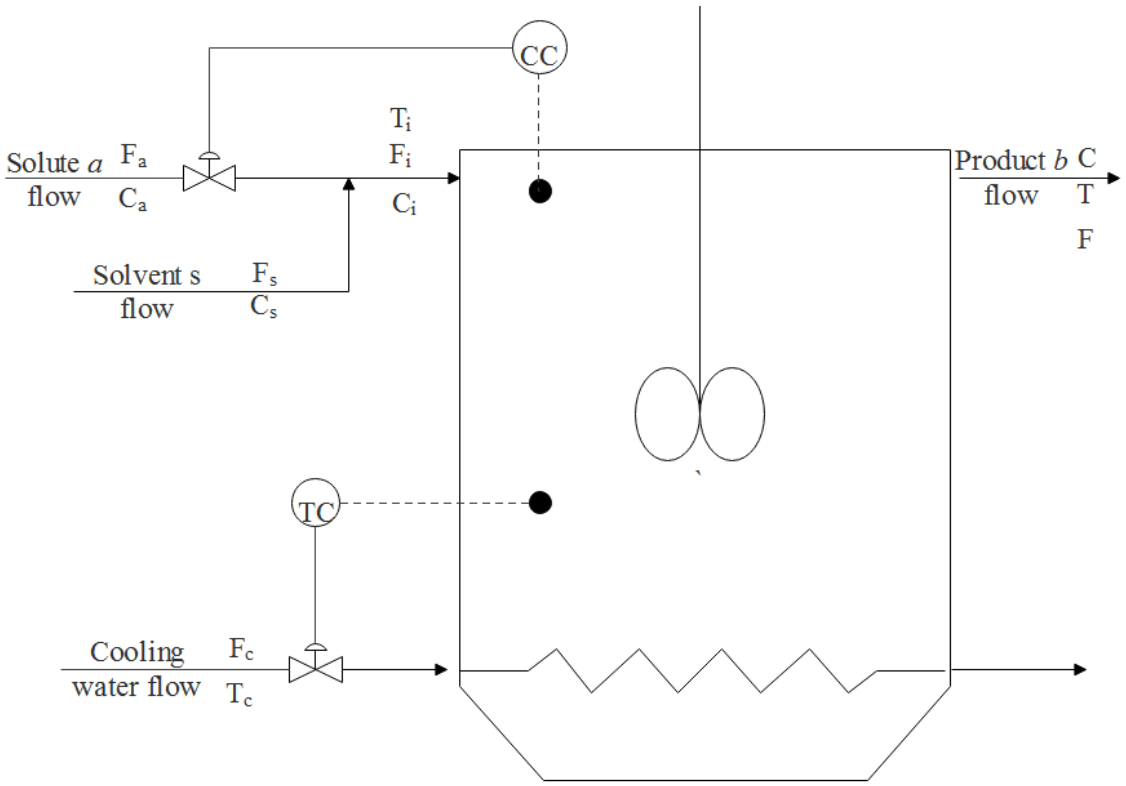


Figure 5.1 Non-isothermal CSTR flow diagram with a concentration controller (CC) and temperature controller (TC).

5.1.2 Tennessee Eastman process

The TE process is a close simulation of a chemical process. It was developed to provide a realistic industrial process on which performance of process control and monitoring methods can be evaluated in a plant-wise manner (Downs and Vogel, 1993; Russel, Chiang, and Braatz, 2000). The overall process is made up of five major process units: a reactor, a condenser, a compressor, a separator, and a stripper. A total of eight components are associated with the process; five gaseous reactants: A, C, D, E, and B (inert); and liquid products: G, H, and by-product F formed.

The process flow sheet (in Figure 5.2) shows the formation of the liquid products from the gaseous reactants fed into the reactor. The reactions are irreversible, exothermic, and approximately first-order with respect to the reactant concentrations (Russell, Chiang, and Braatz, 2000). A summary of the reactions is presented in Equations **5-5** to **5-8**. The Arrhenius function of temperature represents the reaction rates, with the formation of product G having a higher activation energy than that for H.

$$A(g) + C(g) + D(g) \longrightarrow G(l) \qquad \textbf{5-5}$$

$$A(g) + C(g) + E(g) \longrightarrow H(l) \qquad \textbf{5-6}$$

$$A(g) + E(g) \longrightarrow F(l) \qquad \textbf{5-7}$$

$$3D(g) \longrightarrow 2F(l) \qquad \textbf{5-8}$$

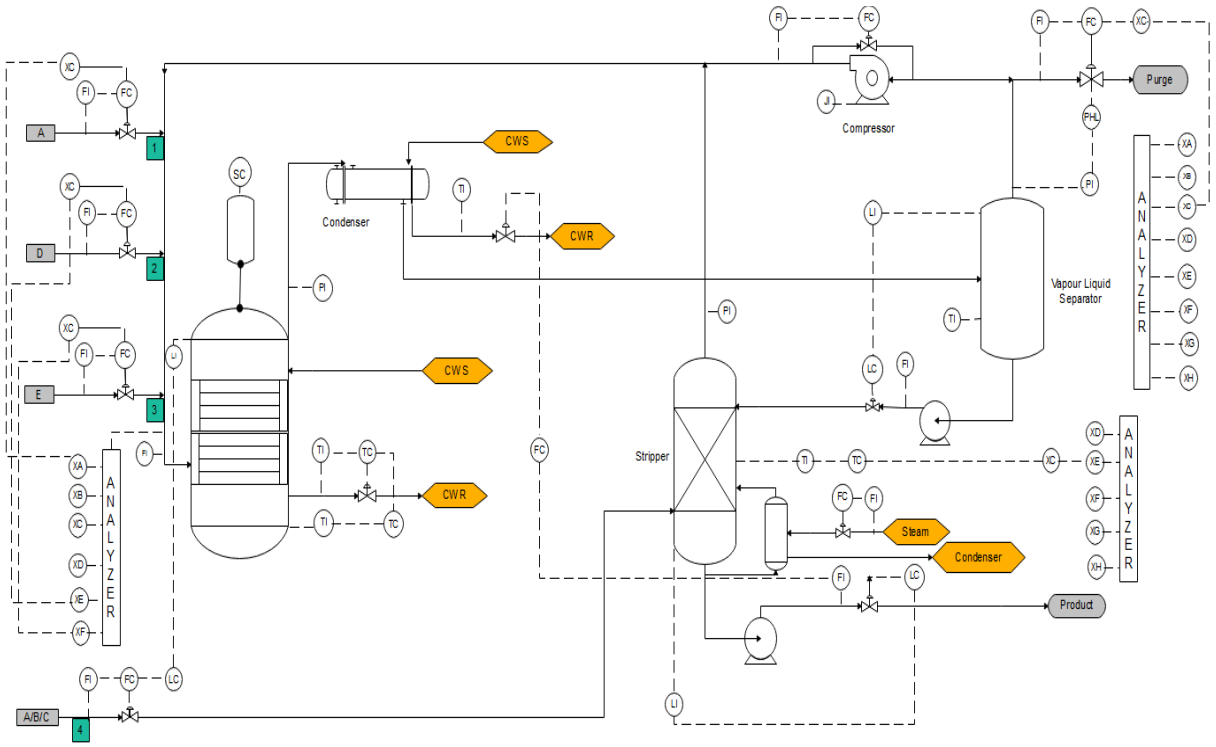


Figure 5.2: Process flow diagram for the TE process (Redrawn from Russell et al., 2000).

The products from the reactor are cooled in the condenser and then fed into and separated in a vapour-liquid separator. The non-condensed vapour is recycled to the reactor by using a compressor, with an amount of it purged to prevent the build-up of the by-product B and inert A. The stripper is then used to remove the remaining reactants from the liquid produced by the separator. The removed reactants are recycled while the remaining liquids are the products: G and H.

Simulation data and Fortran 77 code for the TE process simulator (with plant-wide closed-loop and open-loop) are available for 22 cases. This consists of NOC and 21 process fault

conditions. A total of 52 process variables which comprised of 41 process variables and 11 manipulated variables are generated from the process. For each fault case, the provided simulated data consists of 480 samples of training (NOC) data for the fault. The faultless case consists of 500 samples of training (NOC) data which is the overall training data. The actual faultless data consists of 960 samples and is validation data. For each of the 21 fault cases (summary of faults in Table 5.1), the data consists of 960 samples, with the fault ensuing after 161-time steps within these 960 samples.

Table 5.1: Summary of fault descriptions for the TE process.

Fault	Description	Type
1	*A*/*C* feed ratio, *B* composition constant (Stream 4)	Step
2	*B* composition, *A*/*C* feed ratio constant (Stream 4)	Step
3	*D* feed temperature (Stream 2)	Step
4	Reactor cooling water inlet temperature	Step
5	Condenser cooling water inlet temperature (Stream 2)	Step
6	*A* feed loss (Stream 1)	Step
7	*C* header pressure loss (reduced availability) (Stream 4)	Step
8	*A*, *B*, *C* feed composition (Stream 4)	Random variation
9	*D* feed temperature (Stream 2)	Random variation
10	*C* feed temperature (Stream 4)	Random variation
11	Reactor cooling water inlet temperature	Random variation
12	Condenser cooling water inlet temperature	Random variation
13	Reaction kinetics	Slow drift
14	Reactor cooling water valve	Sticking
15	Condenser cooling water valve	Sticking
16-20	Unknown	
21	The valve for Stream 4 was fixed at the steady state position	Constant position

5.1.3 Introduction to results

Monitoring performance is normally evaluated using the following metrics (see Section 2.3.1): alarm rates (false and missed) and detection delays. In this thesis, the monitoring results for the various case studies and implementations will be presented as shown in Figure 5.3 and Figure 5.4; the information summarised in these figures is described in detail below.

For the purposes of this work, alarm rates stated are functions of the number of consecutive observations (indicated by the hyperparameter z) required to trigger an alarm. This approach is considered because the point of interest to an operator is instances in which an alarm is triggered and not each instance of violation of the control chart. As an example, Figure 5.3 shows a monitoring chart with no false alarm indicated although there is observation exceeding the threshold. The presented summary of the graph shows 0% false alarm rate (FAR) because the specified z value is 3. In the case of a specified value of $z = 1$, the alarm rates are comparable with the conventional ones (which count every violation as an alarm as shown in Figure 5.4).

Although detection delays (DD) reported are dependent on the z values, the reported values count the first violation on which the fault is detected, i.e. if $z = 3$ and a fault start time is 101, a fault detected at time 103 has a DD of 0. A detection time of 104 would, however, have a DD value of 1.

The 'Method' in the summary text associated with each graph denotes the applied method, while the 'Retained components' denotes the number of components retained in PCA methods in order to achieve the specified amount of retained variance. 'Confidence level' refers to the level of significance at which the critical value was computed and 'Alarm trigger' denotes the number of observations z required to trigger an alarm. The 'Data' refers to data under analysis (either training, validation or testing) and 'Statistic' refers to the monitoring statistic implemented. 'Data type' takes on two states: unlagged or lagged, which states whether the data matrix is lagged or not. In the case of the latter, an additional label 'Order of lag' is specified. The 'Total FAR', 'Total MAR', and 'Average DD' respectively refer to the total false alarm rates and missed alarm rates as well as average detection delays. In cases where there is more than one fault, the 'Average DD' computes the mean of the detection delays for all the faults. In the case where only a single fault occurs, the total FAR is same as FAR. For training and validation data, DD is not applicable as no faults are involved.

For a typical control chart (which is generated by applying a specific method to collected data from the simulations), if multiple consecutive statistics cross the threshold, an alarm is raised – depending on the value of z. Black circles indicate cases where a fault is not reported by an alarm, while red stars are used to flag false alarms (i.e. when an alarm is raised although there is no fault). Thus, if there are no false or missed alarms, there should be no red stars or black circles respectively on the plot – e.g. Figure 5.3.

Also, to differentiate between Tennessee Eastman and the CSTR process faults, the prefix 'T' and 'C' are respectively used. For example, 'Fault C01' refers to the CSTR process fault number one.

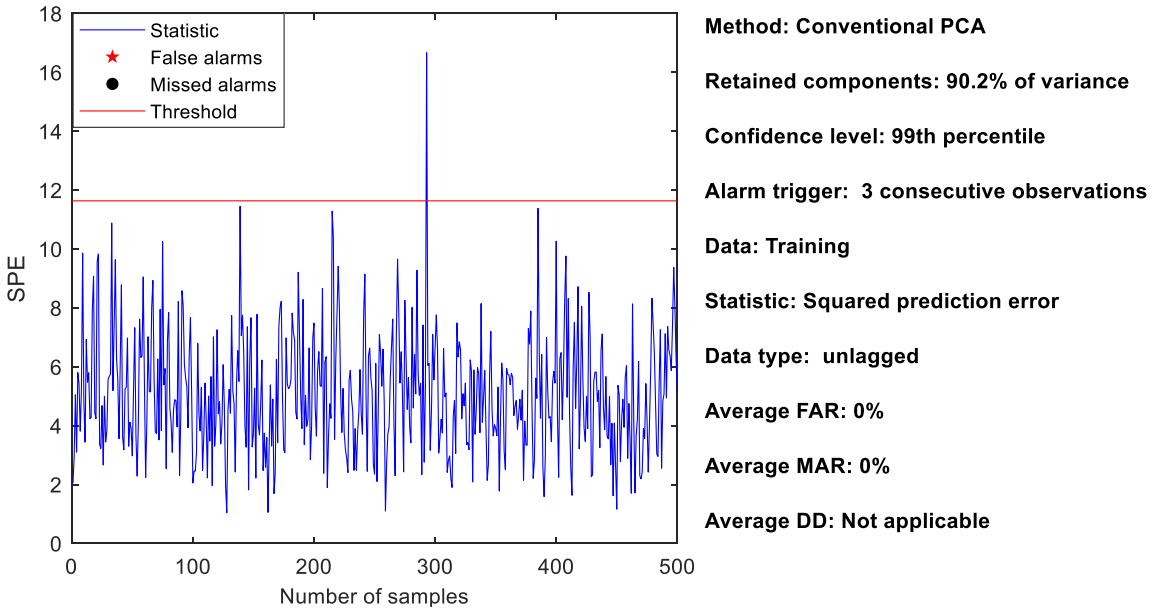


Figure 5.3: Alarm rates as a function of three consecutive observations ($z = 3$).

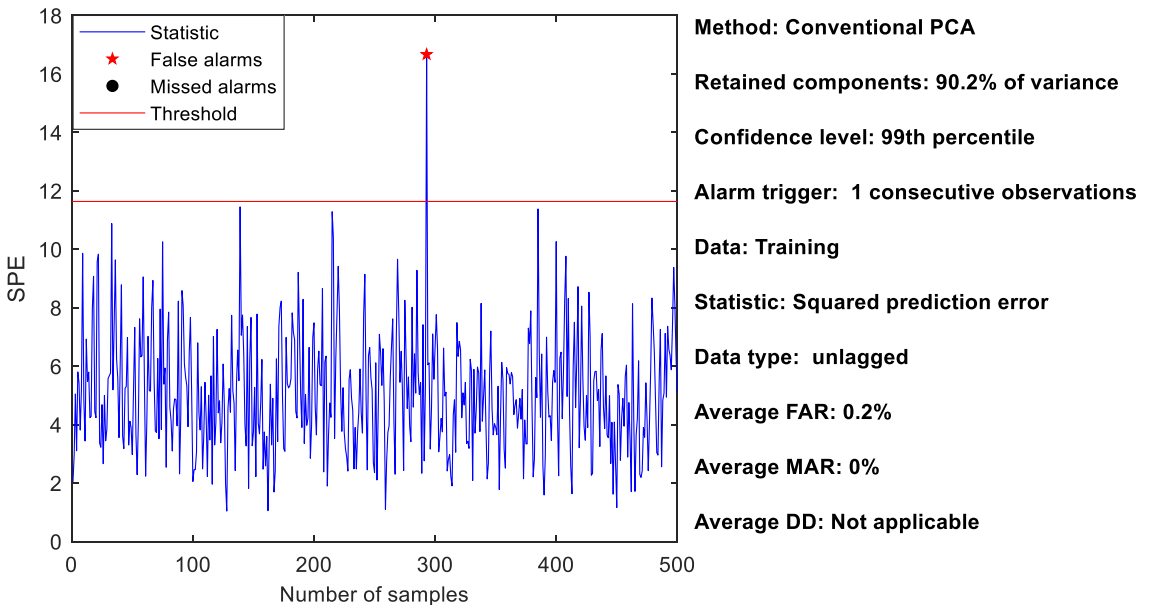


Figure 5.4: Alarm rates as a function of one consecutive observations ($z = 1$).

5.2 Motivation for APCA Methods

As stated earlier, process monitoring involves flagging extreme observations (defined by set limits) as faults. This section, therefore, investigates the implementation of PCA and APCA methods and their pitfalls for respective process changes (faults and drifts) and the robustness

of some model update techniques (specific to adaptive methods). Section 5.2.1 provides a comparison between PCA and APCA while Section 5.2.2 provides a comparison of the various model update as well as APCA techniques.

5.2.1 Comparison of PCA and APCA methods

Section 5.2.1.1 considers the performance of APCA and PCA at a basic level in order to evaluate the performance for stationary process cases. The extra advantage of APCA is subsequently assessed in non-stationary process changing cases.

Section 5.2.1.2 considers various implementation techniques available in the literature for APCA, analyzing the respective implementation techniques and possible pitfalls. After which a proposed heuristic is provided to improve the performance.

Datasets considered were the TE process data (described previously) and the specific CSTR data (which will be described later).

For PCA methods, the factors that come into play in monitoring performance are the retained PCs (PCs), the detection threshold (established at specified confidence level) and the consecutive number of observations to trigger an alarm.

5.2.1.1 Base case (no process drift)—TE process

Representative results for adaptive (in Figure 5.5) and conventional PCA (in Figure 5.6) produce a substantially similar performance. Also, sample AUC results for APCA and conventional PCA (in Figure 5.7)) shows a value of around 0.945 achievable for both methods (computed over plausible thresholds). Sample DD values (in Figure 5.8) produce a superior performance for APCA with a comparable MAR.

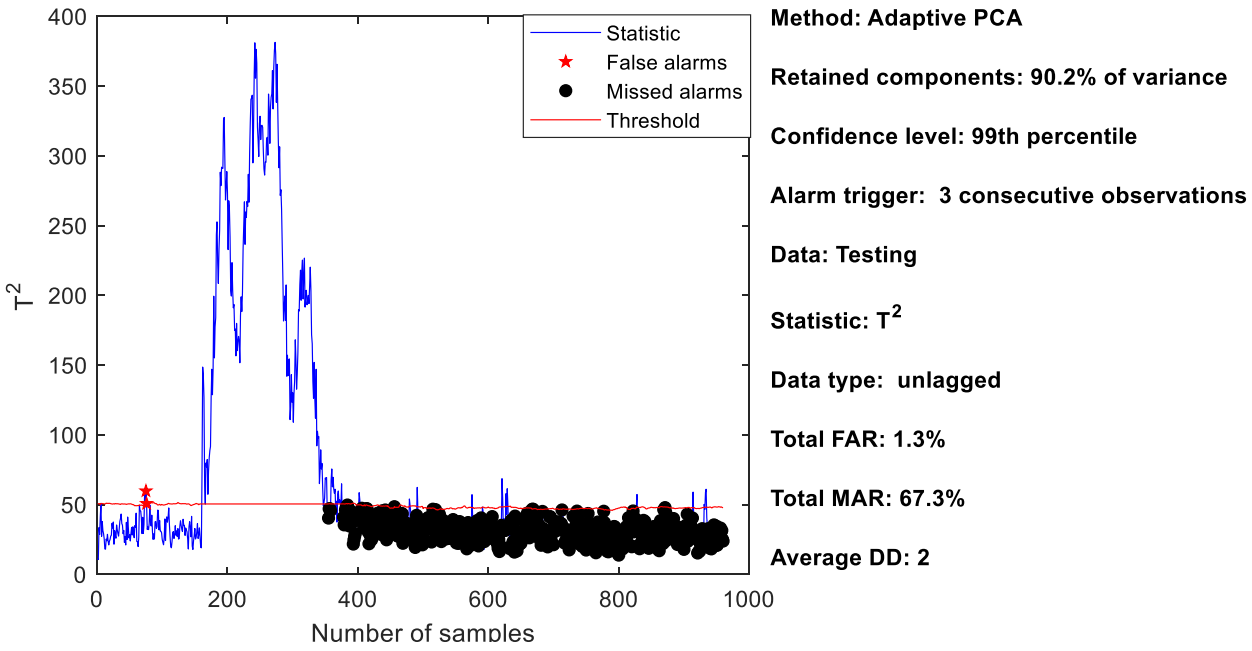


Figure 5.5: APCA T^2 statistic for Fault T05 showing similar performance for CPCA in Figure 5.6.

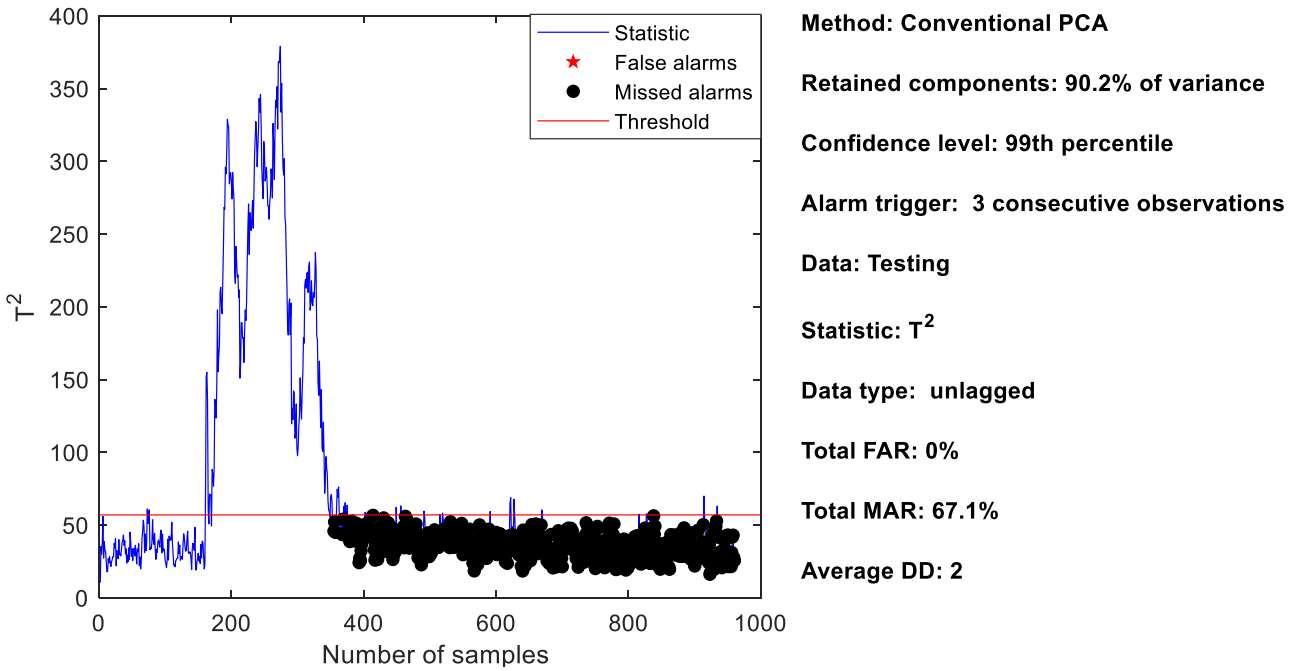


Figure 5.6: Conventional PCA T^2 statistic for Fault T05.

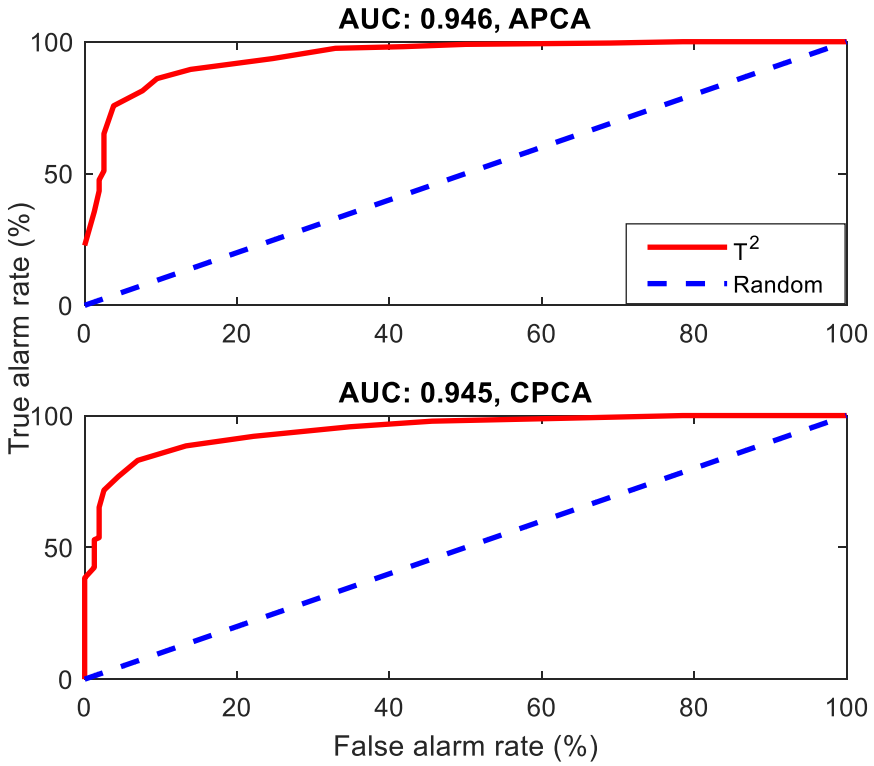


Figure 5.7: ROC curves for Fault T05 evaluated at $z = 3$ and 90.2% of variance.

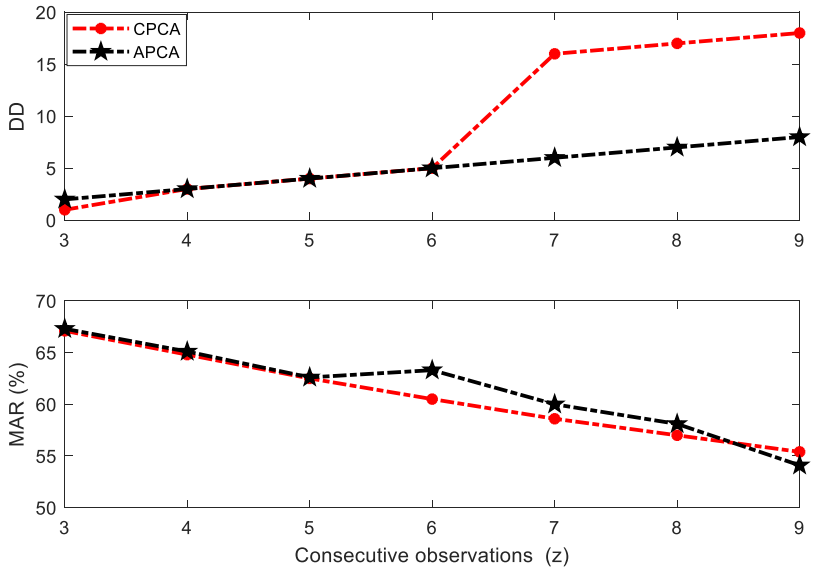


Figure 5.8: DD and MAR for T^2 statistic of Fault T05 evaluated at $z = 3$ and 90.2% of variance.

Further comparison of sample faults for the TE process is presented in Appendix B.

Apart from faults 3, 9 and 19 which are 'almost impossible' to detect, as observed and confirmed by Russell, Chiang and Braatz (2000), all other faults in the TE process simulation are detectable by both approaches, with similar performance. A full comparison of the developed PCA model with that produced by Russell, Chiang and Braatz (2000) as well as

analysis of all faults for APCA and conventional PCA is provided in Appendix C which shows similar performance for both methods.

This section established that both conventional and APCA achieve substantially similar results for processes that involve no drift.

5.2.1.2 Process drift case—CSTR process

The previous section demonstrated that both conventional and APCA are able to detect process faults without non-stationary changes (also known as process drift). Moving forward, the performance of these methods during non-stationary changes (drift) are assessed by using simulated data from the CSTR process.

The performance of the SPE statistic for conventional PCA and APCA are presented in Figures 5.9 and 5.10 respectively. The conventional PCA clearly demonstrate an inability to cope with the process changes; APCA, however, copes with the changes which consequently results in no false alarms.

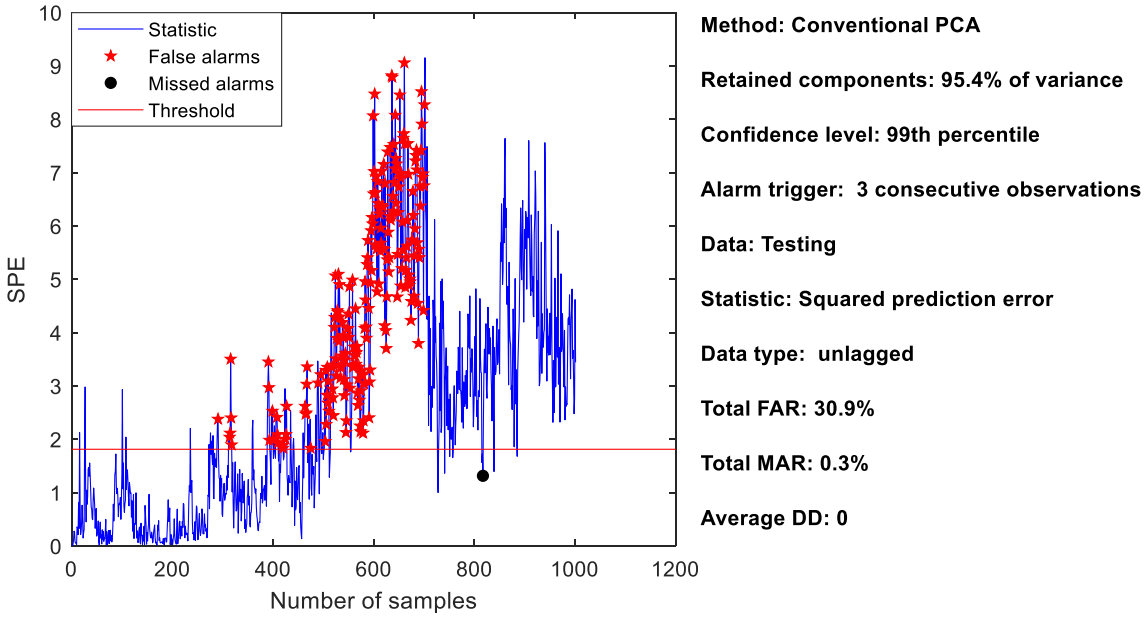


Figure 5.9: Conventional PCA SPE statistic for Fault C02 showing high FAR rates.

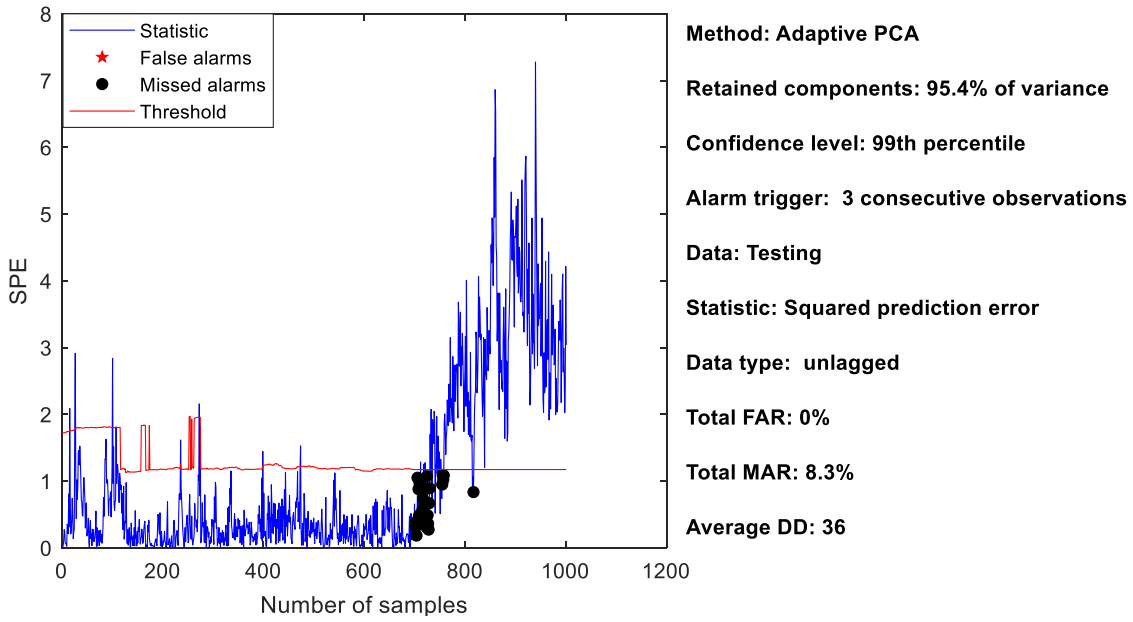


Figure 5.10: APCA SPE statistic for Fault C02 showing better FAR as compared to Figure 5.9. (For a full description of Fault C02, see Appendix D.)

In summary, the results show that APCA produces similar monitoring performance results to those achieved with conventional PCA in cases without process drift. In cases where there is process drift, APCA performs better than conventional PCA.

The next section, therefore, considers the various APCA methods.

5.2.2 Comparison of APCA methods (RePCA and MWPCA)

It was established in the previous section that APCA performs better than conventional PCA when monitoring non-stationary processes. Two of the existing APCA methods considered as stated earlier are the recursive and MWPCA (RePCA and MWPCA, respectively). As stated earlier, while the MWPCA updates the model by moving the data window for new observations, RePCA expands the data window to include new observations.

The obvious potential issue is a slow response in the case of RePCA. MWPCA adapts more quickly, but the obvious challenge is a selection of a reasonable window size to enhance monitoring performance.

The next section, therefore, compares the performance of MWPCA and RePCA using simulated CSTR case studies.

5.2.2.1 Base case (Drift and step fault and back to control)—CSTR process

Sample results for MWPCA (in Figure 5.11) show a similar performance to RePCA (in Figure 5.12) at a window size of 100% training data size (i.e. 1001 samples). The results show the

ability of the two methods to perform at an approximately-equal substantial level. Figure 5.13 shows MWPCA performing better in most cases, and worse for both FAR plots, at least. The general trend of alarm rates shows that the window size above 50% (i.e. 501 samples) provides the lowest sum of alarm rates (i.e. TAR + FAR) for all statistics. A way to get an insight of performance for a specific window size for an adaptive monitoring approach is to use the sum of alarm rates evaluated at that particular window size.

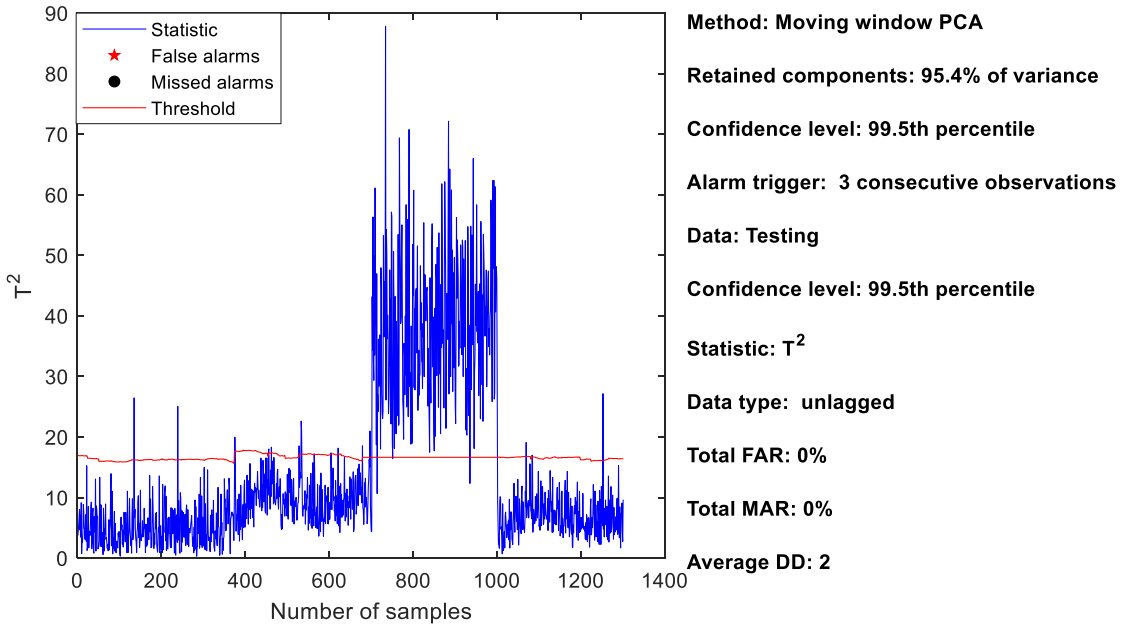


Figure 5.11: MWPCA T^2 statistic performance for Fault C03.

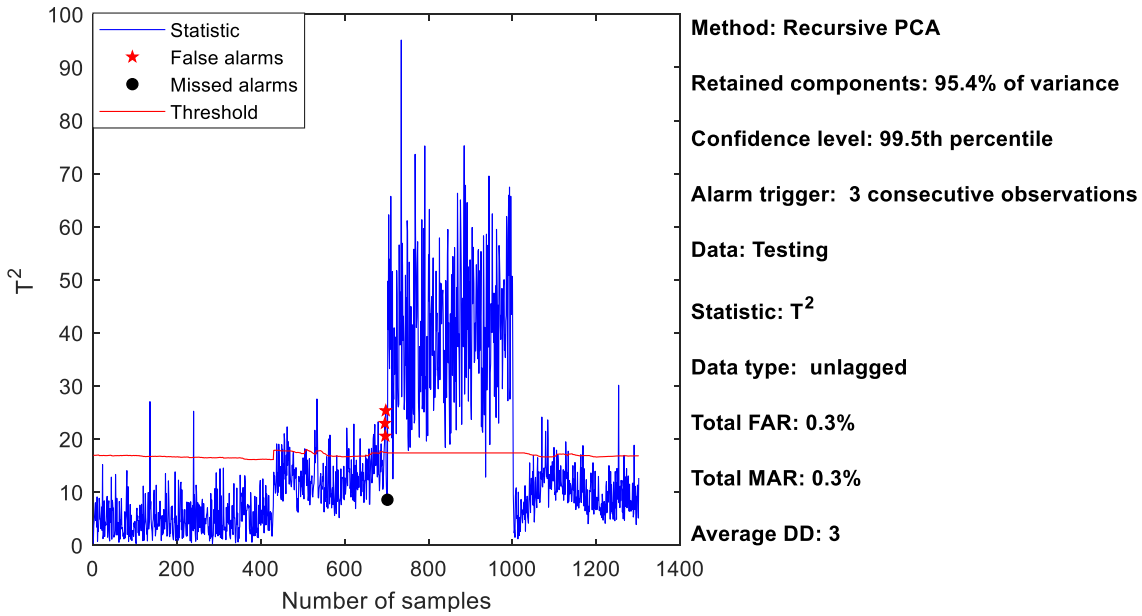


Figure 5.12: RePCA T^2 statistic performance for Fault C03.

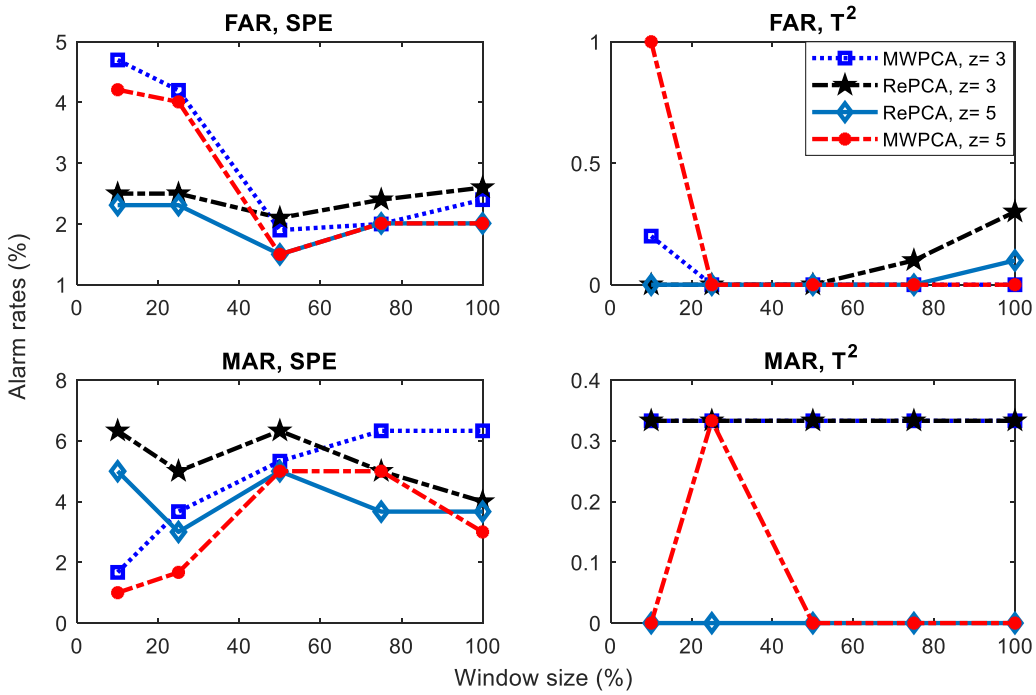


Figure 5.13: Alarm rates for Fault C05 of MWPCA and RePCA for different retained PCs, window sizes, and *z* values at 99.5 confidence level. (For full analysis and description of Fault C03, see Appendix E.1.)

RePCA struggles with an increase in the number of observations (5001 in this case). This is because new observations cause slow changes in the model parameters once a large number of observations have been incorporated into the model. The initial window of both recursive and MWPCA model is the same for the training data (i.e. 1001). Consequently, the RePCA model would have accumulated about 4202 observations (i.e. 1001 from the training stage and 3001 from the test stage) in the model before the process drift begins at time 3202. The MWPCA model, however, keeps the same window size (equal to 1001) at each time stamp. One can see this slow response to changes for the RePCA in Figure 5.14 as opposed to MWPCA in Figure 5.15.

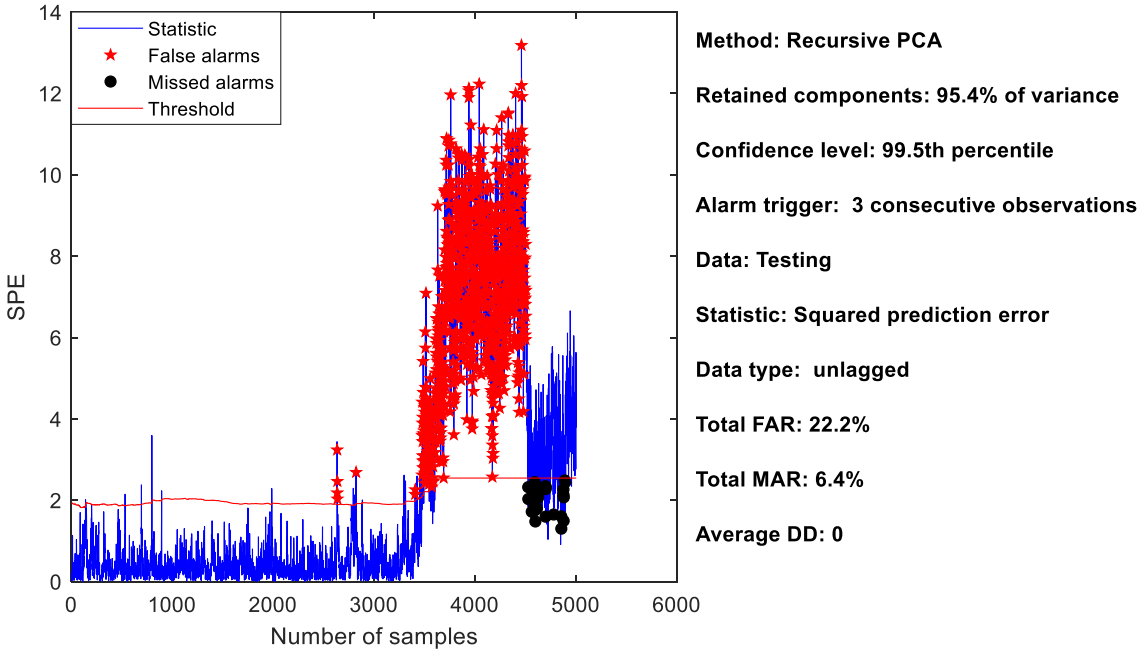


Figure 5.14: RePCA SPE statistic for Fault C04.

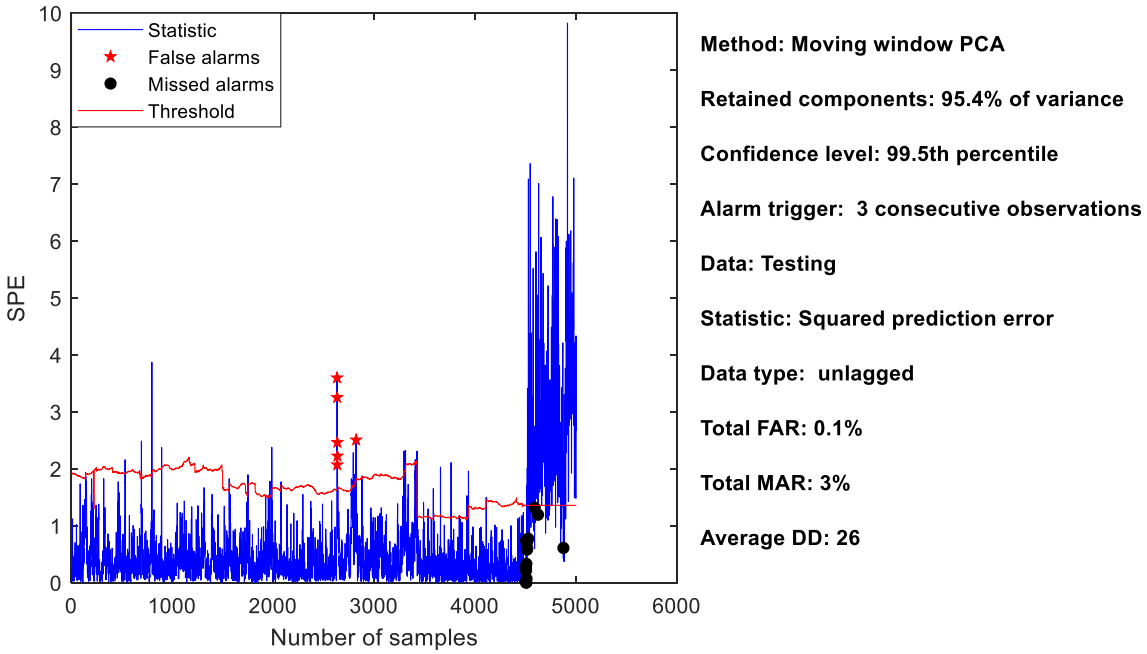


Figure 5.15: MWPCA SPE statistic for Fault C04. (For full analysis and description of Fault C04, see Appendix E.3.)

In summary, RePCA is good for improving a PCA model with new data over time when adaptation is not required. However slow responsiveness means MWPCA is better for adaptive scenarios.

The next part considers some heuristics used in the model update and their performance for different fault types.

5.2.2.2 Comparison of adaptation techniques for model update

In the previous section, it was confirmed that MWPCA is then a better-performing monitoring approach than RePCA for adaptive monitoring. Various update techniques (see Section 3.3.5) are suggested in literature as a basis for a model update. These update techniques are considered in this section to analyze their performances.

The fourth update method (UM-4) is our proposed method, which uses the heuristic that computes the pseudo-updated model parameters for statistics (see Section 4.1.1) as well as the requirement that condition UM-1 is met, and all z observations for at least one of the monitoring statistics (SPE or T^2) are in control.

Figures 5.16 and 5.17 respectively show the FAR and MAR for SPE statistic performance for a process that experiences drift, then a step fault, and returns to control. MAR for all methods apart from UM-1 look satisfactory for most of the instances considered. The FAR, however, show high values at low thresholds for the methods UM-2 and UM-3 at 95.4% of explained variation. The sum the alarm rates (i.e. MAR + FAR) results in a better performance for UM-4.

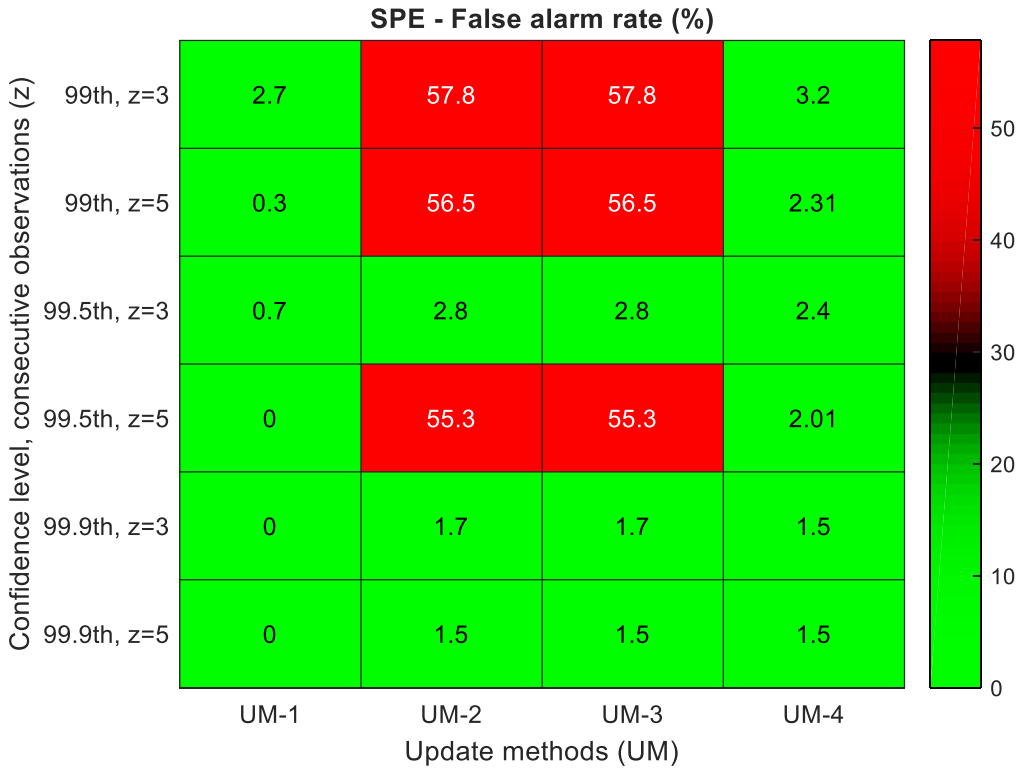


Figure 5.16: False alarm rates for Fault C03 of SPE statistic showing worst performances for UM-2 and UM-3.

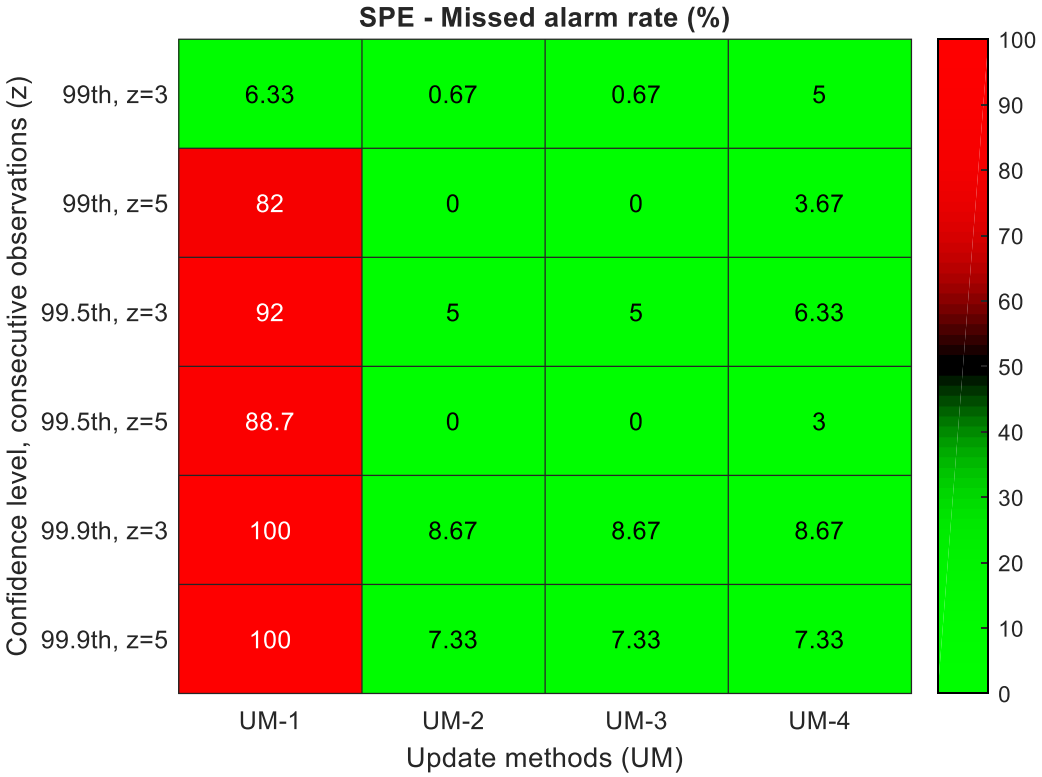


Figure 5.17: Missed alarm rates for Fault C03 of SPE statistic showing worst performances for UM-1.

Figures 5.18 and 5.19 respectively show the SPE statistic performance using the update methods UM-3 and UM-4. The poorest performance at this stage is that for UM-3. This is as a result of UM-3 requiring all z consecutive observations ($z = 3$ in this case) to be in control before an update is performed. The failure of UM-3 is as a result of the limits (thresholds) being challenged at each instance (which results in failure to update). Failure to update an earlier stage serves as a precedent not to update future observations.

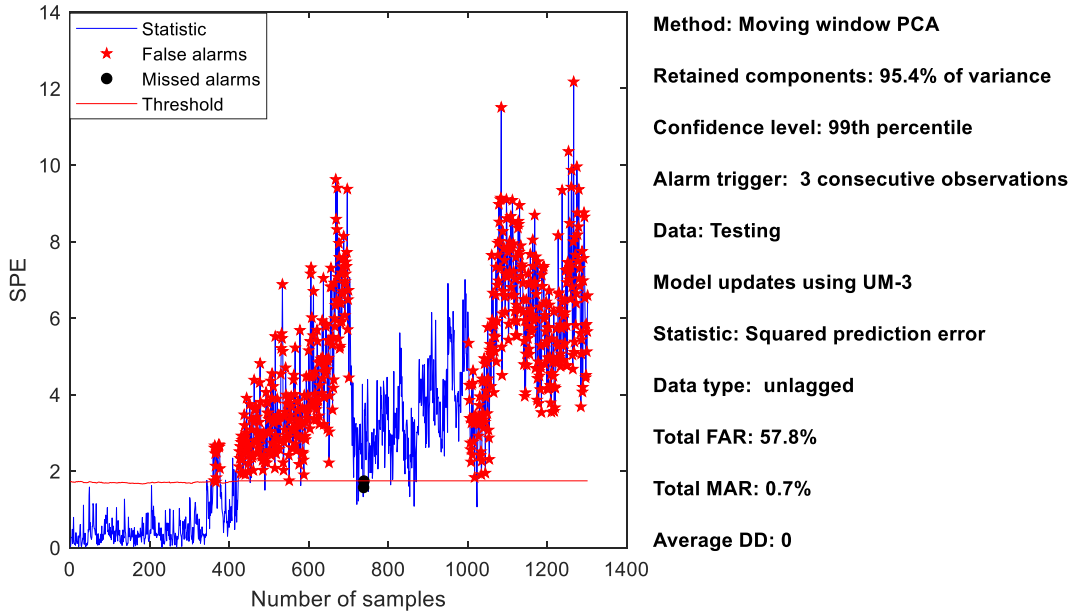


Figure 5.18: SPE statistic for UM-3 for Fault C03 showing high FAR.

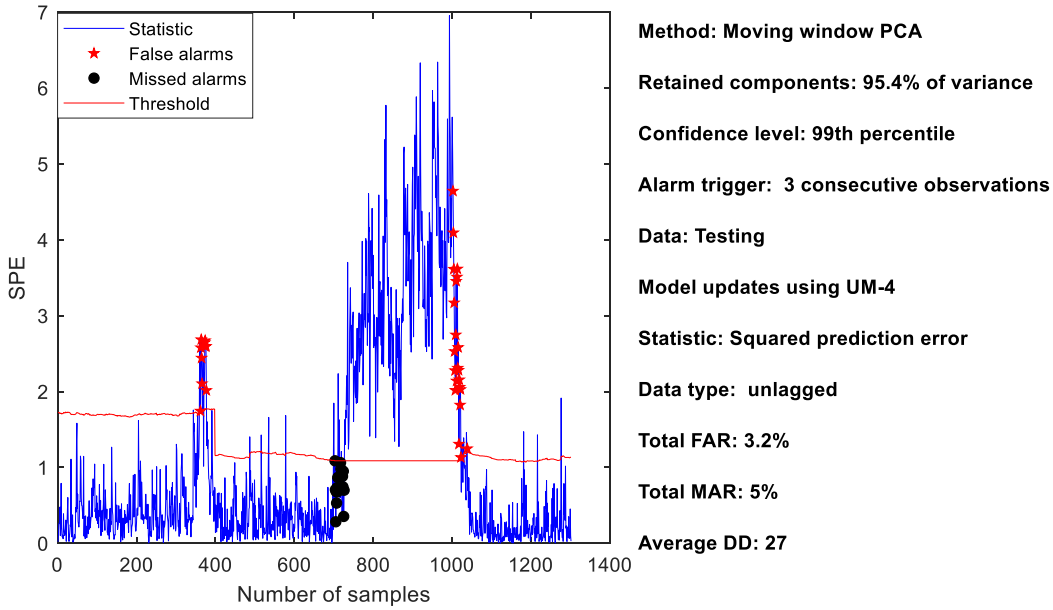


Figure 5.19: SPE statistic for UM-4 Fault C03 showing better FAR. (For a full analysis of Fault C03, see Appendix F.1.)

Figures 5.20 and 5.21 respectively show the FAR and MAR for SPE statistic performance evaluated at the specified levels for a process with drift, intermittent, and step fault. All update methods show similar results for FAR apart from UM-3 which exhibits significantly higher values for most instances. However, UM-3 achieves zero MAR. The cumulative worst performance was produced by UM-1. UM-2 provided a good result for lower threshold values while UM-3 performs opposite to UM-2 overall. UM-4 produced good results except for a threshold value evaluated at the 99.9^{th} percentile and $z = 3$. The generally best performing update method, for Fault C06, is achieved by UM-4.

Failure of the UM-1 is as a result of continuous adaptation in presence of all observations resulting in negligible FAR but higher MAR. UM-2 which requires only the current observation to be in control performs well with lower thresholds. However, when high thresholds are used for UM-3, the limits are not challenged by the observations which had lower values than the thresholds although they are faulty, resulting in model updates with the faults. This gradually declined performance. UM-3, on the other hand, performed better for the higher thresholds due to the requirement of all consecutive observations to be in control. Therefore, at lower significance levels, the drift results in exceedance of the thresholds leading to high FAR. UM-4, on the other hand, showed generally good results for all instances apart from the 99.9^{th} percentile with $z = 3$. Once again UM-4 requires no alarm to be triggered and any of the statistics to have all observations in control to cause an update. The chances of 5 consecutive observations to all stay in control even at a high threshold level is less likely than for 3 which consequently results in lower MAR for $z = 5$ than for $z = 3$, due to a smaller update probability.

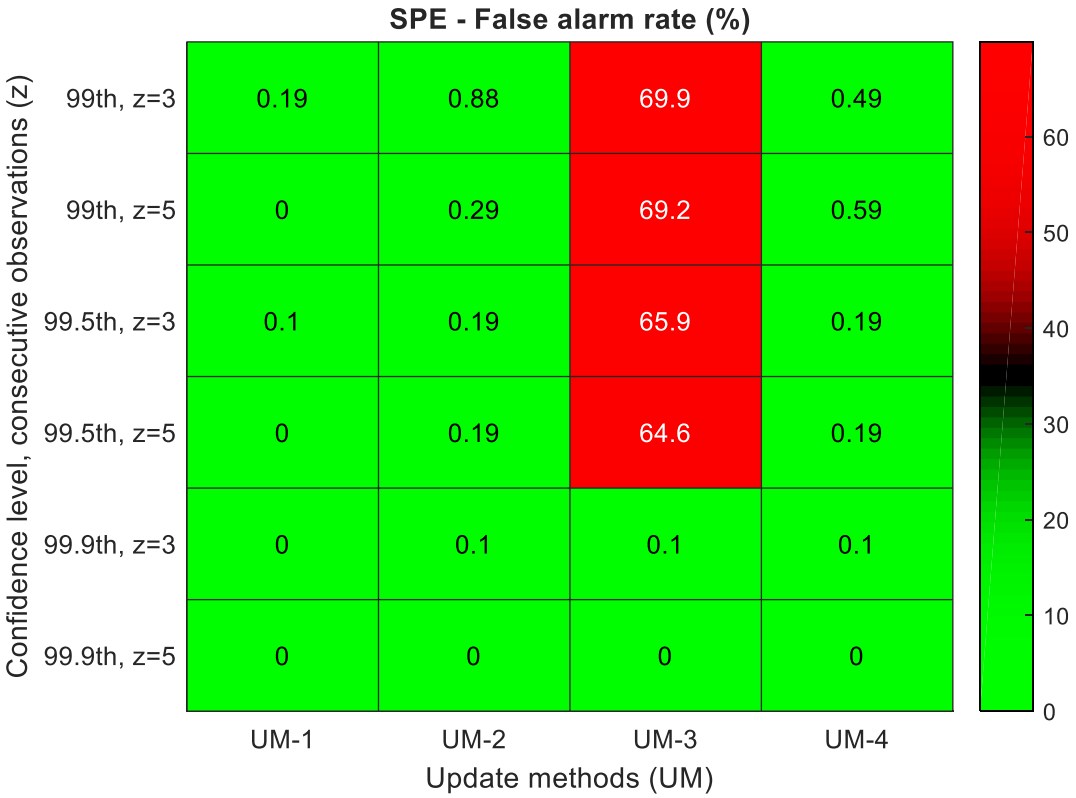


Figure 5.20: SPE statistic FAR for Fault C06 for specified update methods.

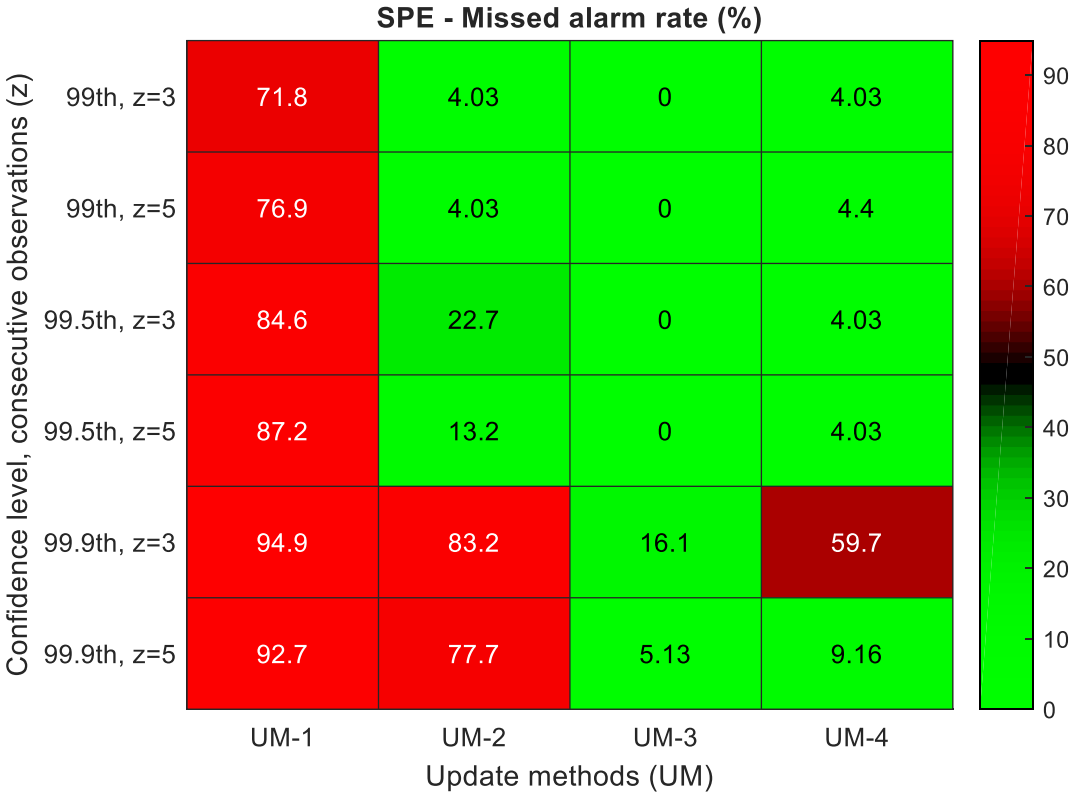


Figure 5.21: SPE statistic MAR for Fault C06 for specified update methods. (For a full analysis and description of Fault C06, see Appendix F.4.)

Overall performance for UM-2 shows low alarm rates for all situations excluding the 99.9th percentile. UM-3 also seems to perform well at the 99.9th percentile. UM-4 provides satisfactory results across all significance levels with a substantially high amount of alarm rates in two instances of high threshold (99.9th percentile) with lower *z* value, and low percentile (99th threshold) with high *z* value.

In summary, UM-1 should never be used, the update method UM-2 requires low percentile while UM-3 requires high thresholds values. UM-4 does well for all thresholds but just like UM-2 and UM-3, high z values are required. Overall, for a unimodal process, MWPCA is recommended with UM-4.

The next section considers the performance of APCA for processes that exhibit multimodal properties and the need for multimodal monitoring approaches.

5.3 Motivation for multimodal monitoring techniques

APCA approaches including MWPCA were established as a better monitoring approach for non-stationary processes in the previous section. A general limitation of PCA and APCA techniques is poor performance on multimodal data. GMM, on the other hand, learns a suitable model representing each mode in the multimodal case.

To establish the performance of GMM against PCA, a case study involving multimodal data is considered using data simulated from the CSTR process.

5.3.1.1 **Multimodal process—CSTR process**

Representative results in Figures 5.22 and 5.23 respectively show the NLPDF (negative log-likelihood of the probability density function) for PCA-based GMM and the T^2 statistic for MWPCA. The results show that GMM better detects the associated faults (lower MAR). The MWPCA results show an infinity (inf) value as the detection delay because it missed at least a fault. Also, observations belonging to the same group are highlighted in the same colour.

Extra hyperparameter settings, such as monitoring type and covariance type in Figure 5.22 are optimized at this stage and will be discussed in more detail in the ensuing sections.

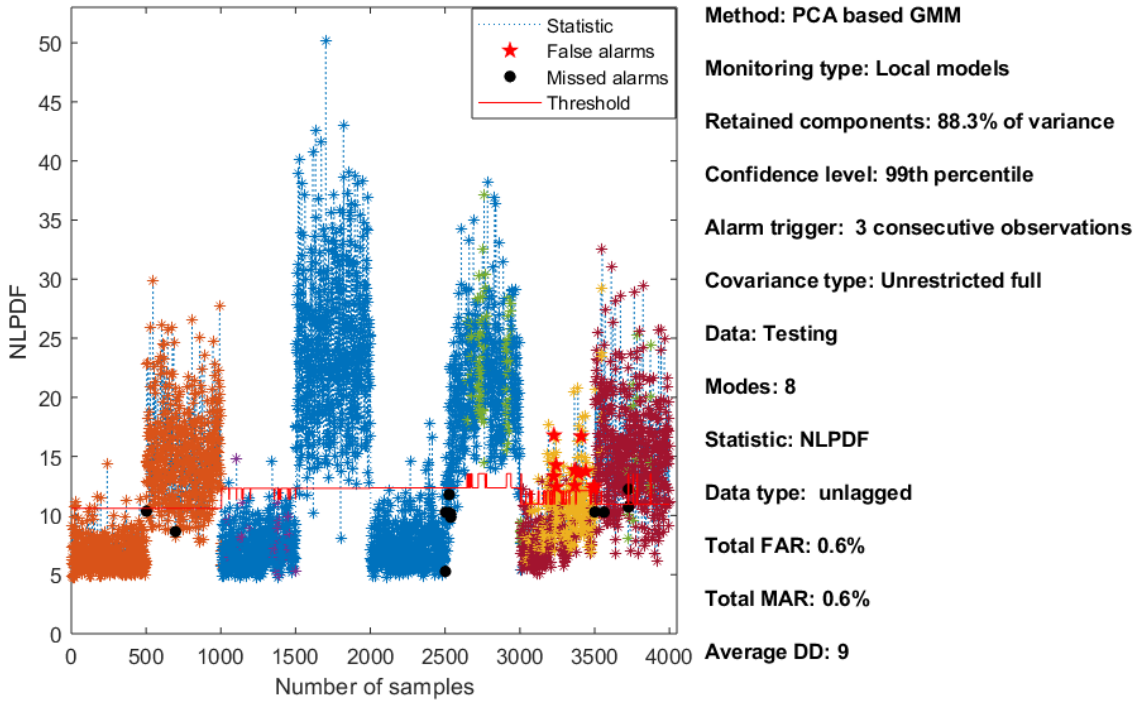


Figure 5.22: NLPDF statistics for Fault C07 of GMM.

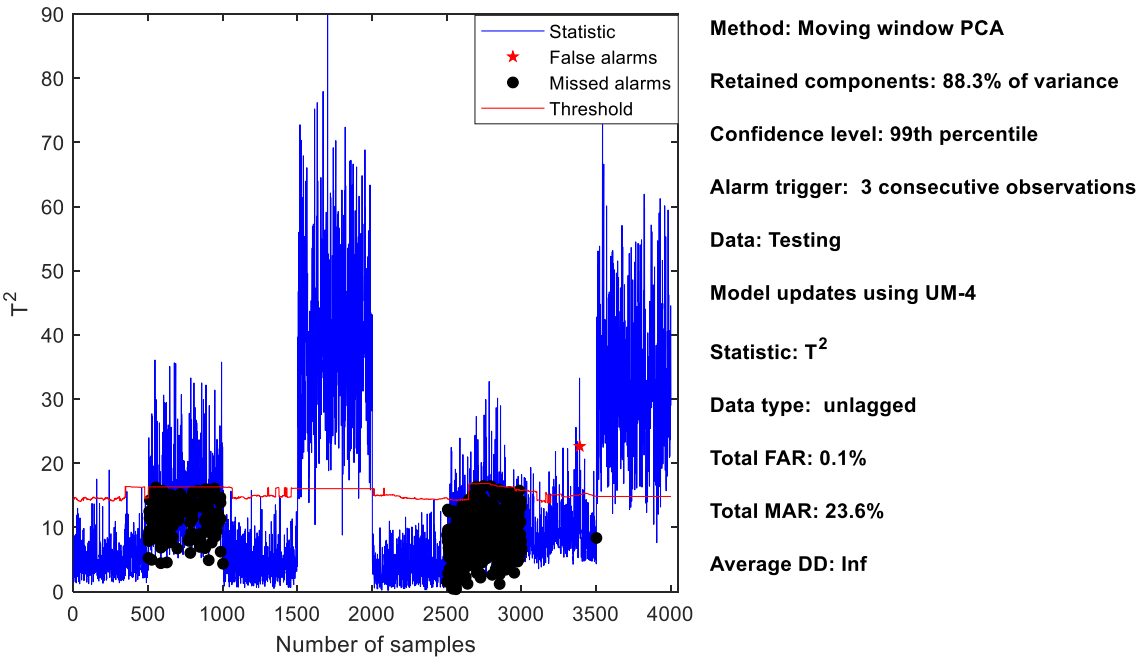


Figure 5.23: T^2 statistic for Fault C07. (See Appendix G.1.1 for a full analysis and description of Fault C07.)

The GMM clearly outperforms the PCA-based technique because the GMM identify each mode as a model on its own and appropriately detect faults within each, as opposed to fitting a unimodal distribution over all the data.

5.3.2 Monitoring model development

Development of a GMM-based method for monitoring requires careful design and investigation of a number of factors/hyperparameters. The hyperparameters of importance are data type (normalized, raw or PCA scores), covariance type and orientation and the number of modes. Another important factor is whether the contribution of each mode is weighted and combined into a single monitoring model or individual models are deployed for each mode (identified by the GMM).

Although GMM is able to handle non-linearity in data, the effect of the high correlation between process data impacts the covariance matrix. This may produce a singular covariance matrix which is not invertible to compute the needed determinant of the covariance matrix. Normalized data is preferred over raw data due to the effect of different variable scales as clustering processes usually rely on the concept of measuring of distance to check dissimilarity. This might be dominated by the variables on a finer scale. In addition, since working with accurately few dimensions (which can be achieved by using the retained PCs) is desired, the need to establish the performance of scores in this regard is important.

The covariance structure types (full and diagonal) as well as the orientations (restricted and unrestricted specify the parsimony (i.e. taking into account model complexity) of the developed model.

Also, the ability of the model to not fit a cluster to few outliers has two impacts: for global monitoring, it raises the model's detection threshold and might impact detection of some faults, and for local monitoring, the probability of assigning future faults to the cluster created by the outliers may be high.

Section 5.3.2.1 considers the prediction of the 'true' number of clusters for unimodal processes (TE and CSTR) by the selection criteria types (AIC and BIC) for specified data types and covariance shapes. All PCs were retained for computation of the PCA scores.

Thereafter, Section 5.3.2.2 considers at a multimodal CSTR process data.

5.3.2.1 Base case (Unimodal process)—TE process and CSTR process

Since the fitting process for GMM usually converges to a local maximum which is dependent on the initial parameter values, there is the need to ascertain the effect of random initializations on the selected number of clusters.

The second level of analysis involves the effect of using diagonal and full covariance matrices, as well as whether the covariance matrices are shared or unshared. Shared refers to the covariance matrix for each of the cluster being identical while unshared refers to the covariance matrix for each of the cluster being unidentical.

Figure 5.4 shows the results for the selected number of clusters for the AIC and BIC at each instance of the run (i.e. a total of 50) for the random initializations. The maximum number of iterations was fixed at 1500. The results show an almost perfect selection of one cluster for the PCA scores for both criteria. While the BIC maintains this behaviour over all data types, the AIC typically selected four to six clusters for the raw and normalized data[5]. Results for the covariance types selected (in Figure 5.25) show that both AIC and BIC for the normalized and raw data consistently selected a full-shared covariance, while for the PCA scores (which have orthogonal features), the diagonal-shared covariance type is selected by both the AIC and BIC in almost all instances.

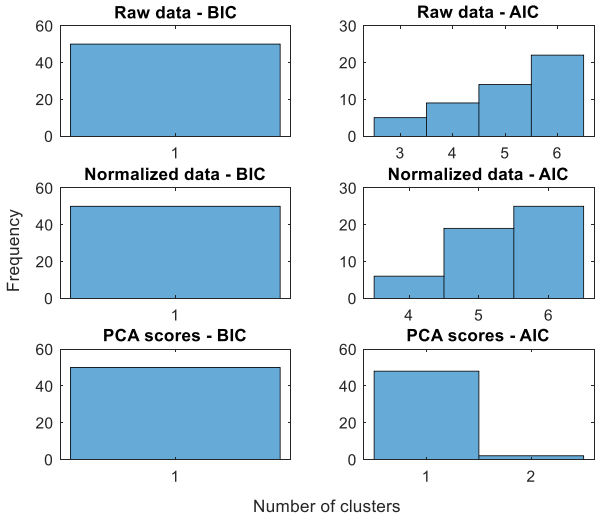


Figure 5.24: Selected number of clusters for various data types using TE training data (with the optimized cluster types selected shown in Figure 5.25)

[5] Once more, normalized data defines data where the raw data variables have been centered and scaled such that the resulting data has a mean of zero and standard deviation of one, while the PCA scores refer to the data's transformed coordinates when projected onto the retained PCs.

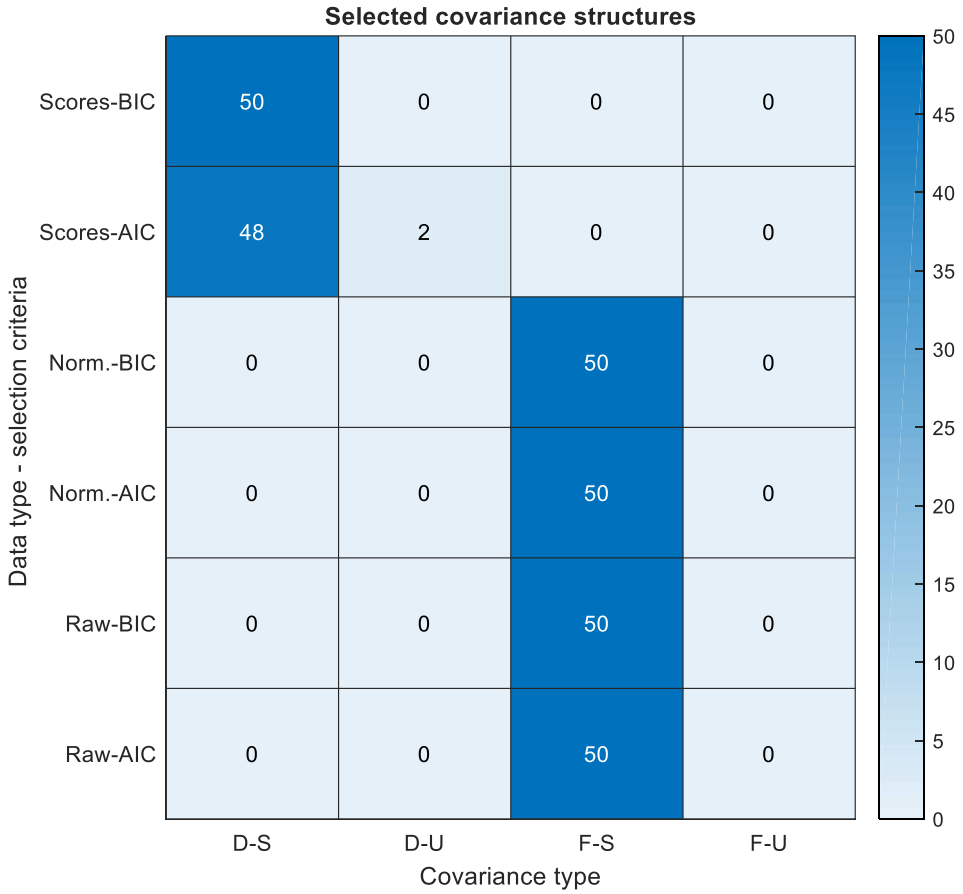


Figure 5.25: Frequency of covariance types: diagonal-shared (D-S), diagonal-unshared (D-U), full-shared (F-S), full-unshared (F-U) for various data types using TE training data. See Appendix G.2.1.1 for further results.

Results for the unimodal CSTR case are similar to those for the TE process and presented in Appendix G.2.1.2. The next section, therefore, considers the results of a simulated multimodal case.

5.3.2.2 Multimodal process—CSTR process

Figure 5.26 shows the results for the selected number of clusters for the AIC and BIC and minimum AIC-BIC (mAB) (see Section 3.5.2.4) for each instance. Results for the PCA scores show BIC to be the best performing criterion while AIC averages eight clusters. The normalized data results showed the closest to the 'true' number of clusters and achieved by the BIC. The raw data, on the other hand, have AIC averaging a lower overall number of clusters selected than the BIC.

The minimum AIC-BIC criterion (which compares the corresponding AIC and BIC scores for the best model selected by the BIC and AIC) coincides with the results of the BIC in all cases.

For all the unimodal processes, there was only one instance in which it selected the best model by AIC. It is noteworthy that simulated training, validation, and test data from the CSTR process are respectively differentiated by the prefix ‘Ct’, ‘Cv’, and ‘C’ (see Appendix D for an illustration).

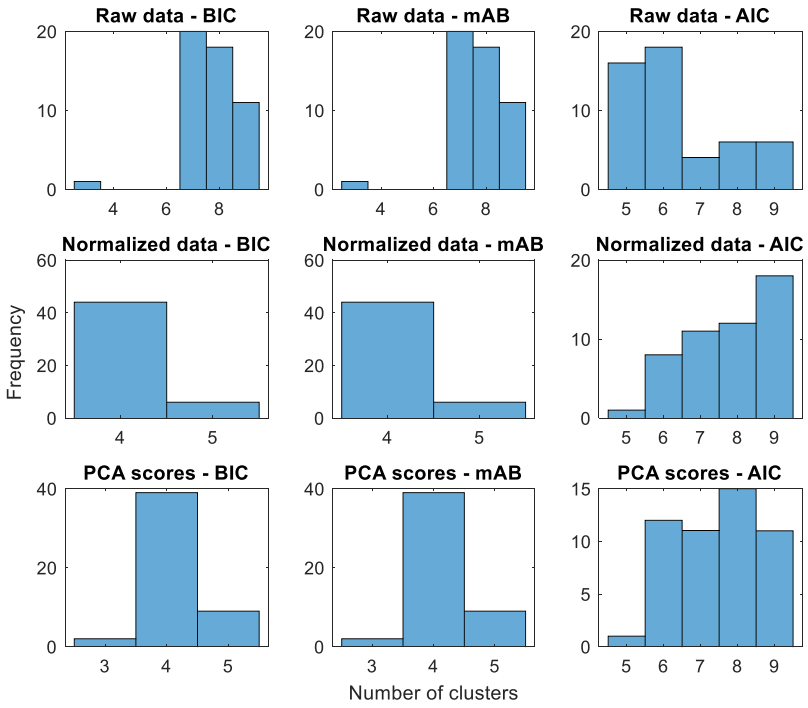


Figure 5.26: Selected number of clusters for various data types using CSTR training data Ct01. The optimized cluster types selected are shown in Figure 5.27. (See Appendix G.2.2 for further results and description of Ct01.)

The results for the selected covariance shapes (in Figure 5.27) shows the selection of full-unshared covariance in most cases apart from the BIC for the raw data. These results present the first time since the unimodal clustering stages in which a diagonal covariance is not selected for the scores by the BIC and also no selection of a diagonal covariance matrix by any criterion.

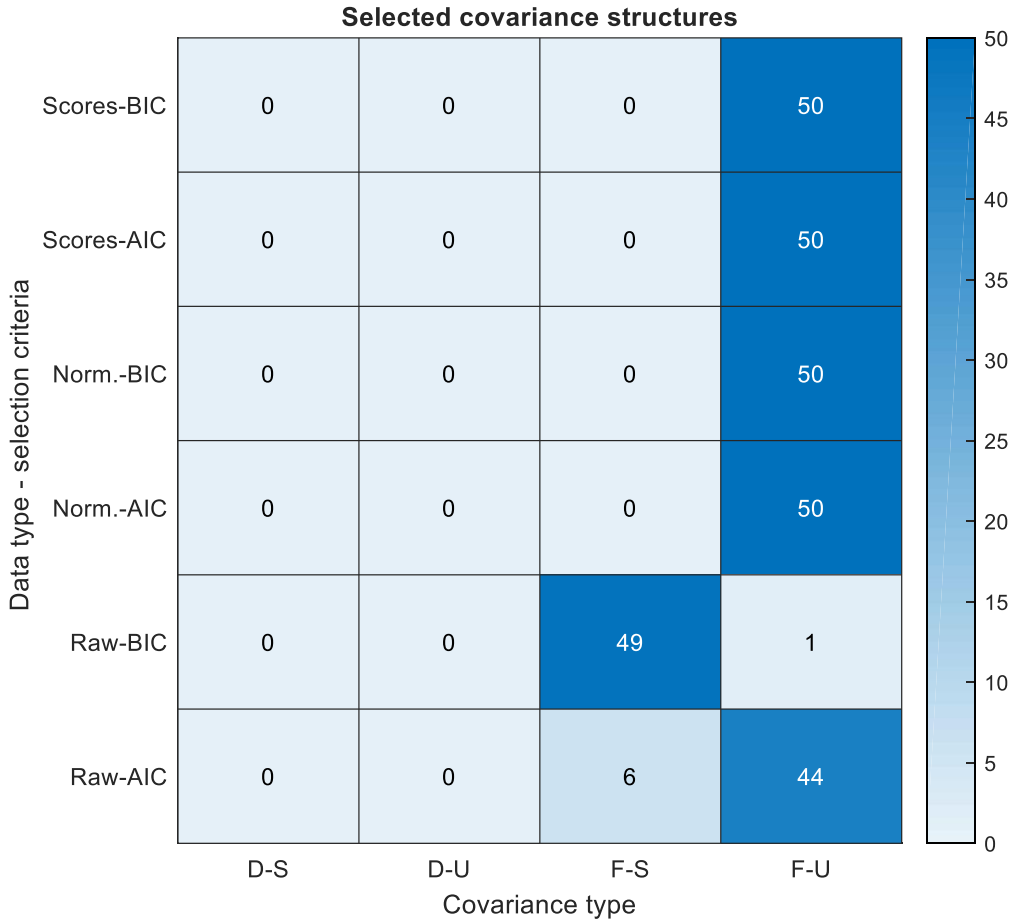


Figure 5.27: Frequency of covariance types: diagonal-shared (D-S), diagonal-unshared (D-U), full-shared (F-S), full-unshared (F-U) for various data types using CSTR training data Ct01. (See Appendix G.2.2 for further results and description of Ct01.)

Overall realization in the selection of covariance types shows the selection of diagonal covariance for the unimodal PCA scores while a full covariance is selected for the other data types. The BIC consistently performed better than the AIC and the PCA scores and almost always perform well for all covariance types.

In summary, the BIC should be used for model selection, while the full covariance type should be coupled with the normalized and raw data. Also, the scores could work with both covariance types.

The next section considers the implications of selecting various covariance shapes and the impact on monitoring performance.

5.3.3 Implementation of the developed model

For monitoring implementation, an additional factor to be considered apart from the covariance types for the respective clusters is the monitoring type implemented. This takes on the option of global or local monitoring as stated earlier in Section 3.5.3.

MAR for monitoring using local models (in Figure 5.28) generally showed improved results with a decrement in the confidence level of the thresholds. Full-unshared covariance provides the best performance over all the data types. The worst MAR performance was produced using the scores with 6 PCs (90% variance) using the diagonal-unshared category.

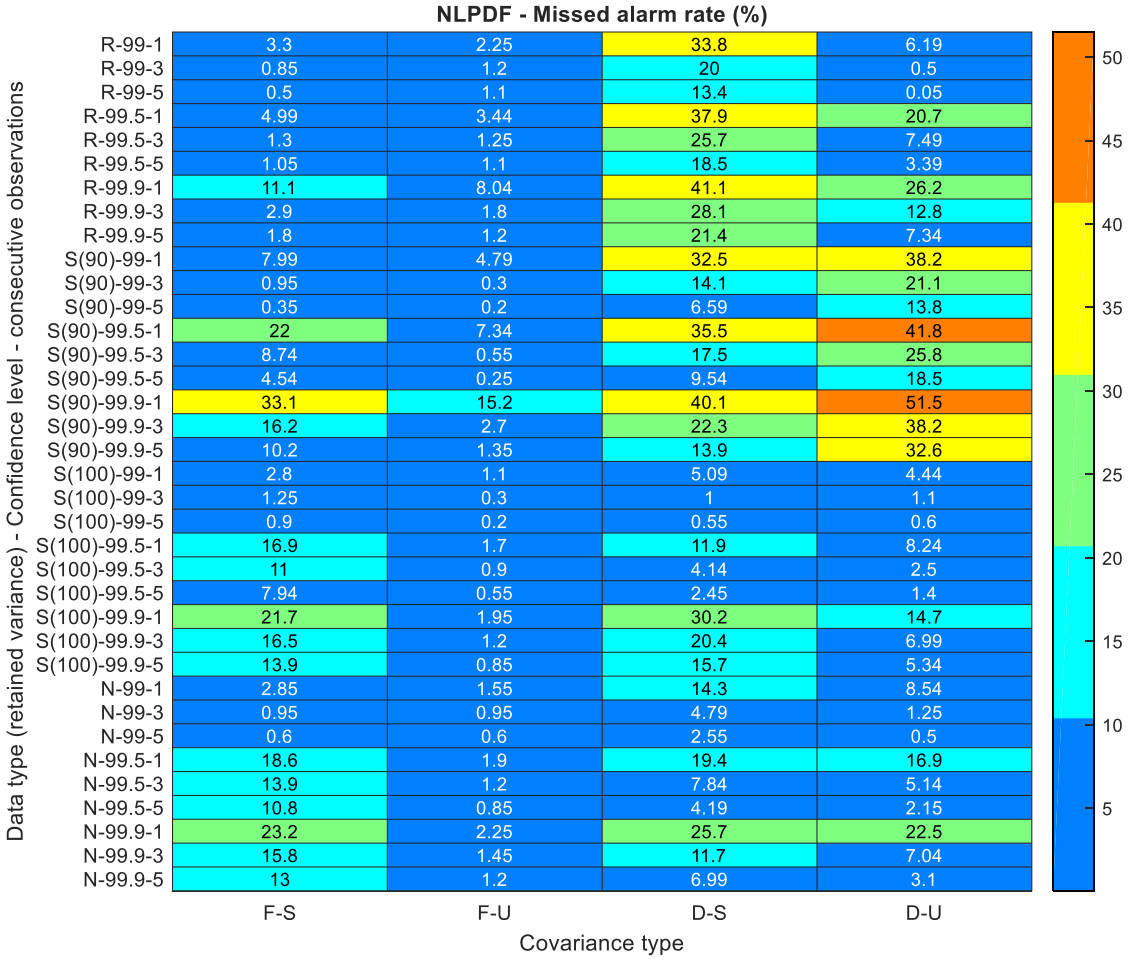


Figure 5.28: MAR for various data types using local models approach. (S = PCA scores, R = raw data and N = normalized data).

Figure 5.29 shows the FAR for the local monitoring approach. High FARs are expectedly received with $z = 1$. The worst performing data type is the raw data. Cumulative results for the FAR and MAR show the full structure performing better than the diagonal covariance matrix

structures. Ignoring the results of the PCA scores with 90% retained variance, however, puts diagonal-unshared covariance and full-unshared covariance at par.

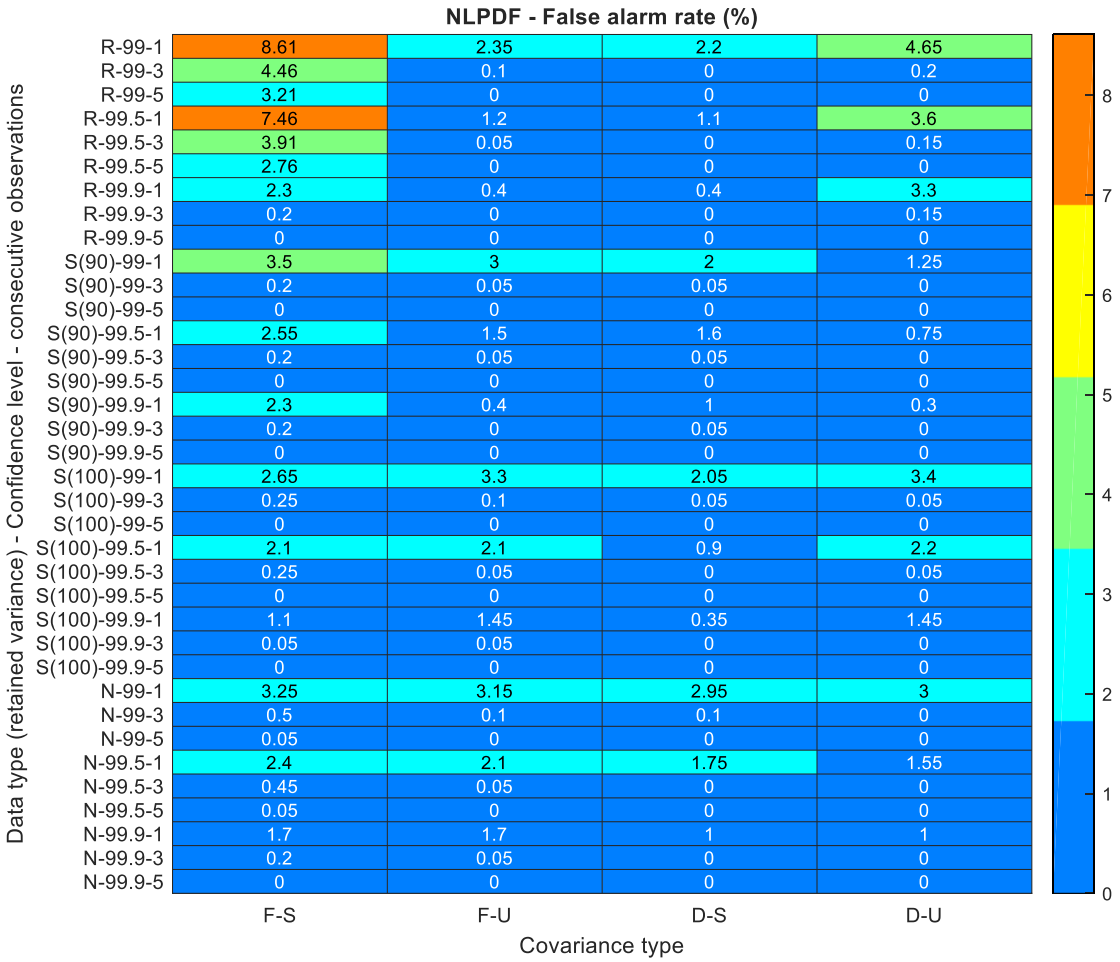


Figure 5.29: FAR for various data types using local models approach. (S = PCA scores, R = raw data and N = normalized data).

MAR results for the global monitoring approach (shown in Figure 5.30) presents a rather high MAR for almost every data type; with the diagonal-shared covariance providing the worst results. The full-unshared covariance still almost always produced the best results over almost all cases considered.

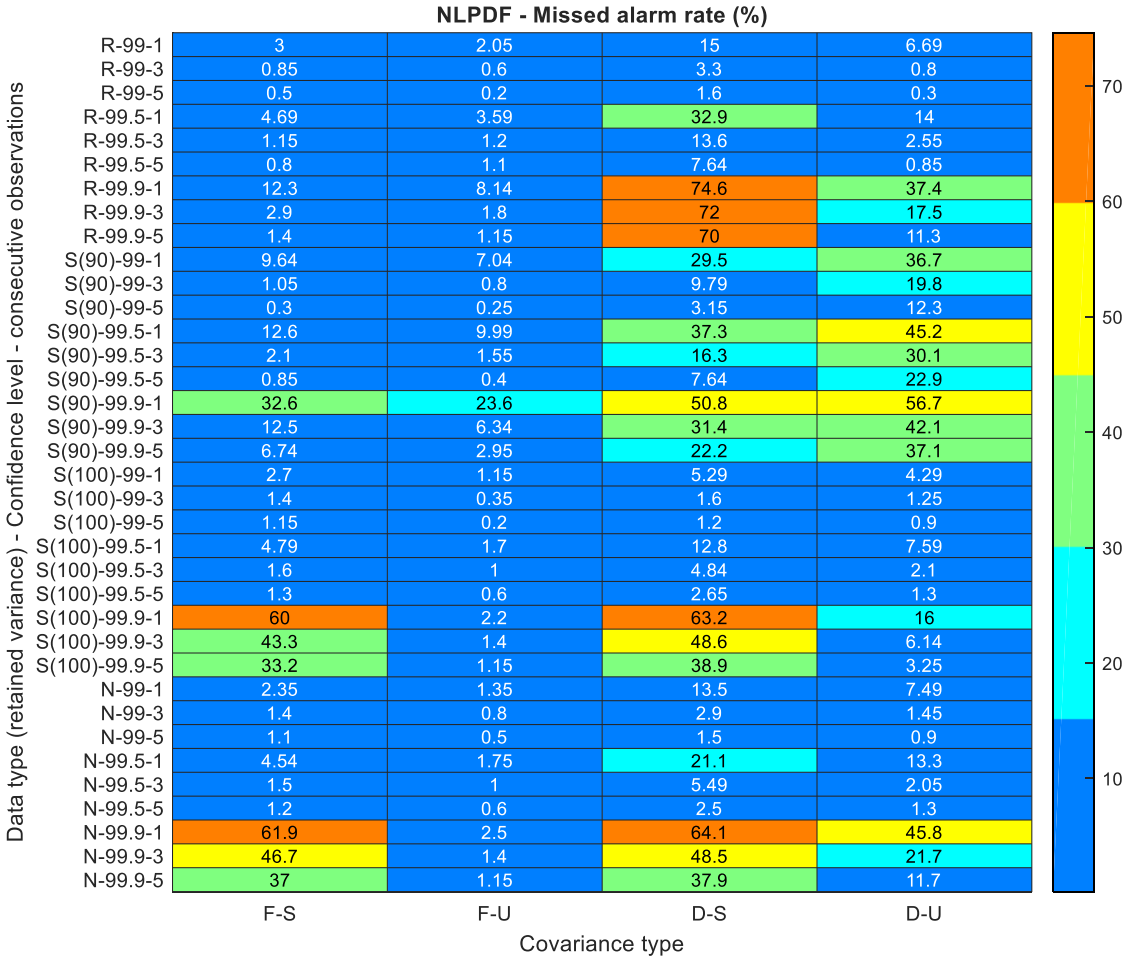


Figure 5.30: MAR for various data types using global models approach. (S = PCA scores, R = raw data and N = normalized data).

The FAR results for the global models (in Figure 5.31), on the other hand, shows relatively lower values than those for the local monitoring approach. The raw data is the worst performing in most instances; with the diagonal-shared covariance still providing the best results.

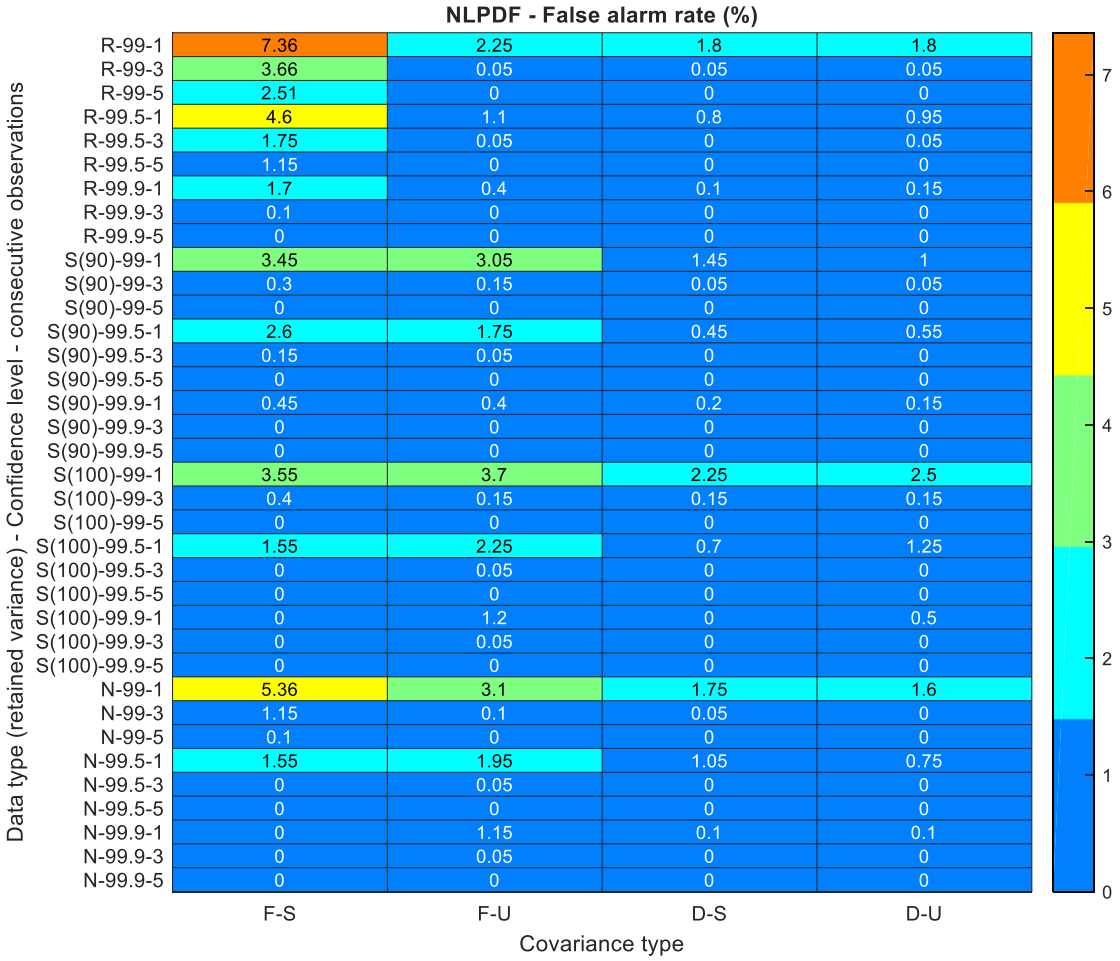


Figure 5.31: FAR for various data types using a global monitoring approach. (See Appendix G.3 for further results and comparison of sample faults.)

The sum of alarm rates (FAR+MAR), however, put the performance of the local monitoring approach ahead of the global monitoring approach. The full-unshared covariance performed best in this case study but the diagonal-unshared is also not far off.

Overall, we observe that there is a loss of some form of fault detection ability with the discard of some PCs. The results achieved with the PCA scores are however not far off the results for the normalized data and or the raw data if close to 100% variance is retained. The information loss, however, does not make PCA-based GMM at a specific number of retained PCs inferior to PCA at the same level. It either performs better or at par for a unimodal case. GMM surely has the upper hand in a multimodal case (see Figure 5.22 and Figure 5.23).

Although the full covariance matrix is less desirable because it is less parsimonious, it provides better results than the diagonal covariance if limits on the number of PCs used require discarding a significant amount of variance (i.e. in the case where a significant fraction of the

variance cannot be captured/retained either due to visualization purposes or computational demand). The diagonal covariance type would produce close results to that obtained for the normalized data if a high amount of variance (where high is around 95%) is retained. The normalized and raw data, on the other hand, almost always fail using the diagonal covariance.

The BIC performs better in this work in consistently predicting the 'true' number of clusters and was almost always (i.e. more than 99%) picked by the minimum AIC-BIC criterion.

In summary, the BIC should be used as well as the local monitoring approach. The scores should be used and with the diagonal covariance matrix only if almost 100% of variance is retained. In the case where the discarded PCs are significant, the scores should be used with a full covariance matrix.

5.3.4 Limitation of PCA-based GMM

Although GMM is shown to produce superior results for multimodal processes; multimodal processes experience drift too. This makes the challenges experienced for conventional PCA resurface in a similar manner for GMM.

GMM is applied to Fault C03 of the CSTR process to verify this assertion.

Sample results for the PCA-based GMM (in Figure 5.32) show the inability of the monitoring model to cope with the changes. Figure 5.33, on the other hand, presents the results for APCA-based GMM. The results prove better due to the model's ability to combine both properties of APCA (i.e. MWPCA) and GMM–this is discussed in the next section as an extension of APCA to GMM. It is noteworthy that the threshold changes for the PCA-based GMM are that for changes in modes and not for adaptation (each mode is represented with distinct colour). This is also true for APCA-based GMM when a global model is used and applies to other results involving GMM approaches.

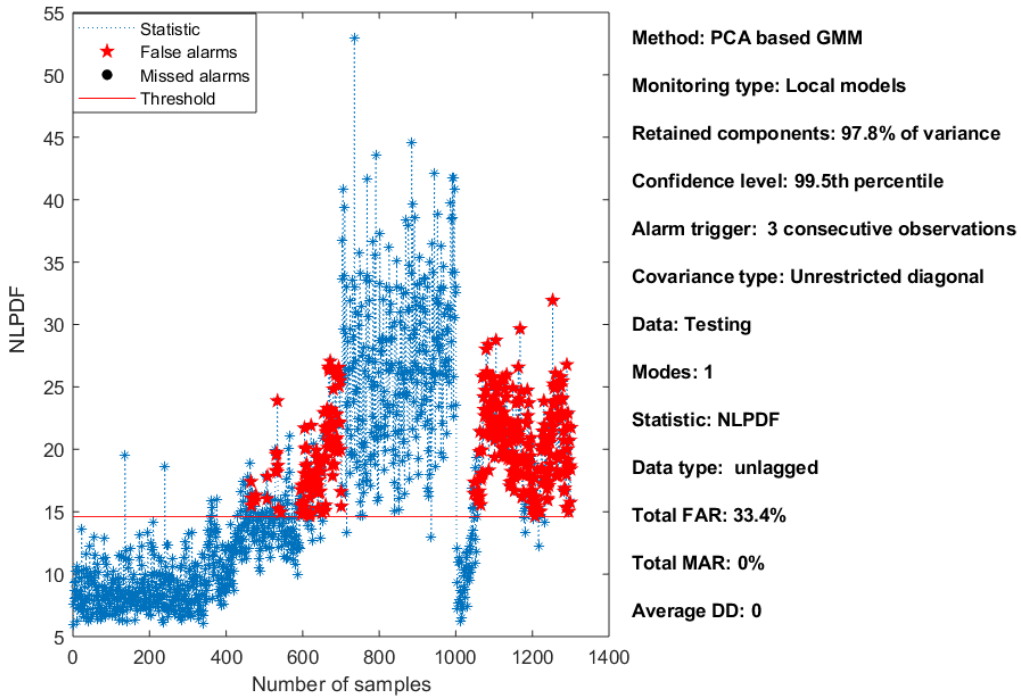


Figure 5.32: NLPDF statistics for Fault C03 of PCA-based GMM showing high FAR.

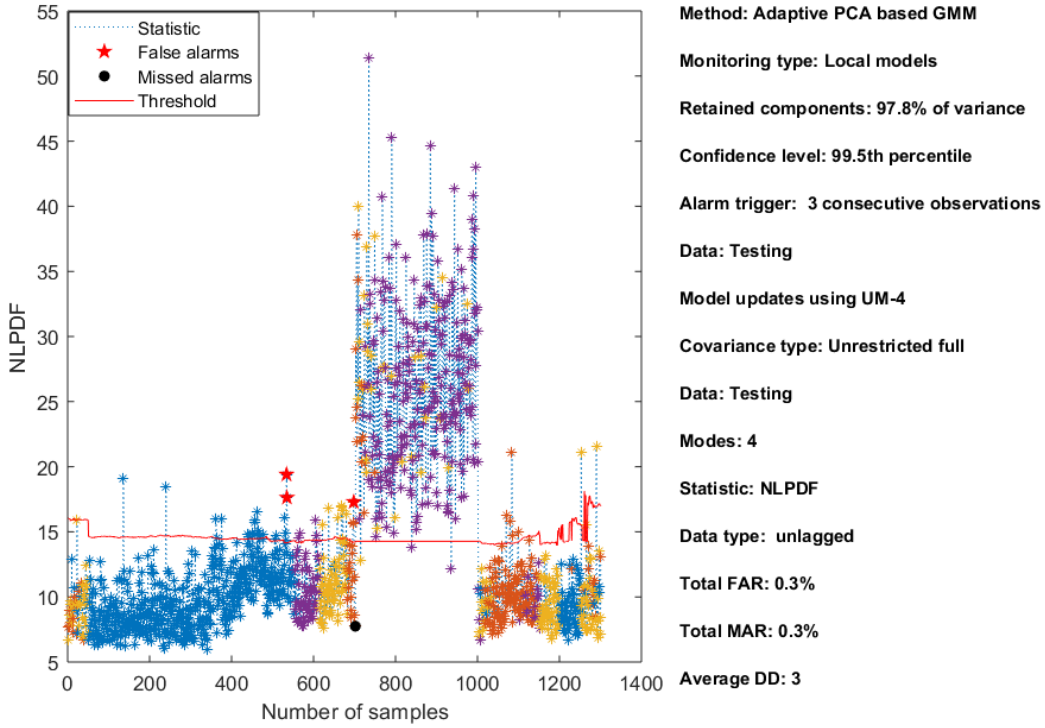


Figure 5.33: NLPDF statistics for Fault C03 of APCA-based GMM showing low FAR.

5.4 Extension of APCA to GMM

APCA-based GMM combines the factors established in APCA such as initial window size, the *z* value, among others, and that from the GMM such as covariance type, monitoring model type (global or local), and the number of clusters. The optimal approaches defined in the previous considerations (i.e. APCA and GMM) extend directly into this method as it is a combination of both methods. In the subsequent results figures, APCA-based GMM is sometimes referred to as MWPCA-GMM and PCA-based GMM as PCA-GMM.

Comparison of the results for a unimodal process with drift by the respective methods (shown in Figure 5.34) shows similar results for the MWPCA T^2 statistic and the APCA-GMM statistic. Also, conventional PCA performs similarly to PCA-GMM in this case. The results also show that APCA-based GMM performs better than the PCA-based GMM.

The SPE statistic for MWPCA shows an infinity value as the detection delay because it entirely missed at least one fault. The DD values presented are averaged over all the intermittent faults and step fault, and failure to detect a fault reports an infinity value which averages to infinity. For cases whereby this occurs, the median DD values are produced to provide further insight.

	PCA		**MWPCA**		**PCA-GMM**	**MWPCA-GMM**
	SPE	**T^2**	**SPE**	**T^2**	**NLPDF**	**NLPDF**
Total FAR	21.4	36.4	1.7	0	37.4	0.2
Total MAR	0	0	10	1.2	0	1.2
Avg DD	0.2	0	inf	2.8	0	2.8
Median DD	0	0	inf	3	6	6

Figure 5.34: Monitoring performance for Fault C05 (a unimodal process with drift) evaluated at 6 retained PCs, 99.5^{th} percentile threshold and $z = 3$.

Further results for another unimodal process with drift (in Figure 5.35) confirms the SPE statistic of conventional PCA as the most severe sufferer from process drift. The SPE of the MWPCA has a high chance of missing intermittent faults and hence the infinity DD value as well as high MAR. Also, results for APCA-based GMM show more favourable performance than that for conventional PCA and PCA- based GMM.

	PCA		MWPCA		PCA-GMM	MWPCA-GMM
	SPE	T^2	SPE	T^2	NLPDF	NLPDF
Total FAR	21.2	51.8	0.1	0.6	53	1.5
Total MAR	3.3	0	17.6	2.2	0	3.3
Avg DD	inf	0	inf	3.2	0	3.8
Median DD	2	0	5	3	6	6

Figure 5.35: Monitoring performance for Fault C06 (a unimodal process with drift) evaluated at 6 retained PCs, 99.5th percentile threshold and z = 3.

The comparison of the results for Fault C07, a multimodal process without drift (in Figure 5.36) shows an expected improvement in performance for the multimodal methods relative to the unimodal methods.

	PCA		MWPCA		PCA-GMM	MWPCA-GMM
	SPE	T^2	SPE	T^2	NLPDF	NLPDF
Total FAR	5	0.1	0.1	0.1	0.2	0.2
Total MAR	17.4	16.7	39.1	8	0.6	0.8
Avg DD	19.5	60.25	inf	53.75	9.25	9.75
Median DD	2.5	15.5	49.5	2.5	4	4

Figure 5.36: Monitoring performance for Fault C07 evaluated at 6 retained PCs, 99th percentile and z = 3.

Finally, results for a multimodal process with drift (Fault C08) shown in Figure 5.37 presents a better performance for APCA-based GMM due to its ability to cope with the drift. Figure 5.38 and Figure 5.39 provide sample plots for the implementation of the two approaches.

	PCA-GMM	MWPCA-GMM
	NLPDF	NLPDF
Total FAR	5.3	0
Total MAR	1.2	0.8
Avg DD	2.5	2.75

Figure 5.37: Monitoring performance for Fault C08 evaluated at 6 retained PCs, 99th percentile and z = 3.

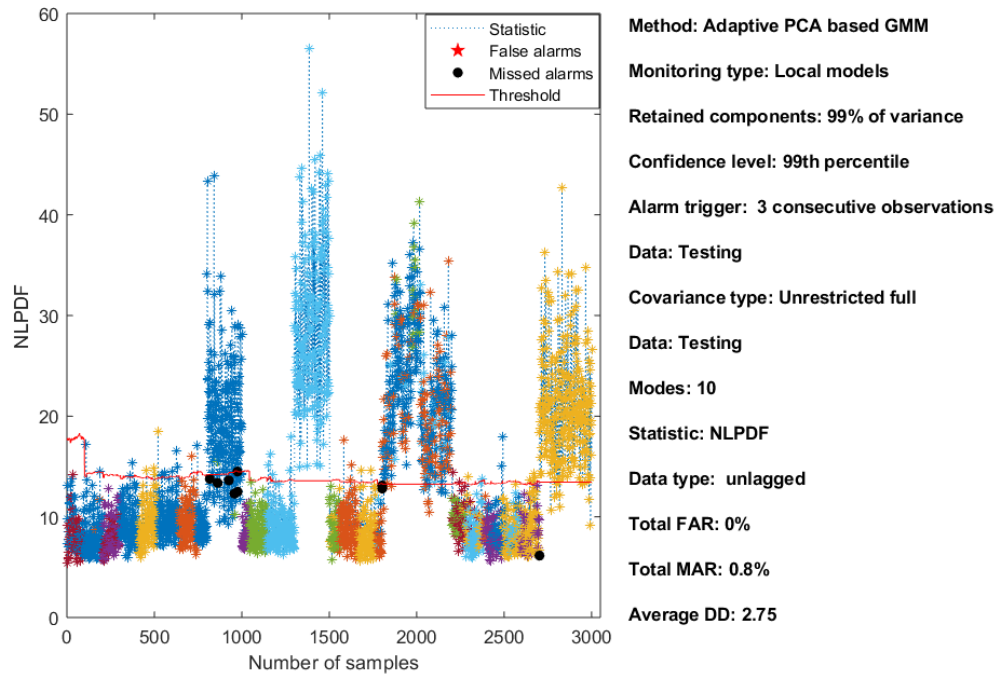


Figure 5.38: NLPDF statistics for Fault C08 of APCA-based GMM.

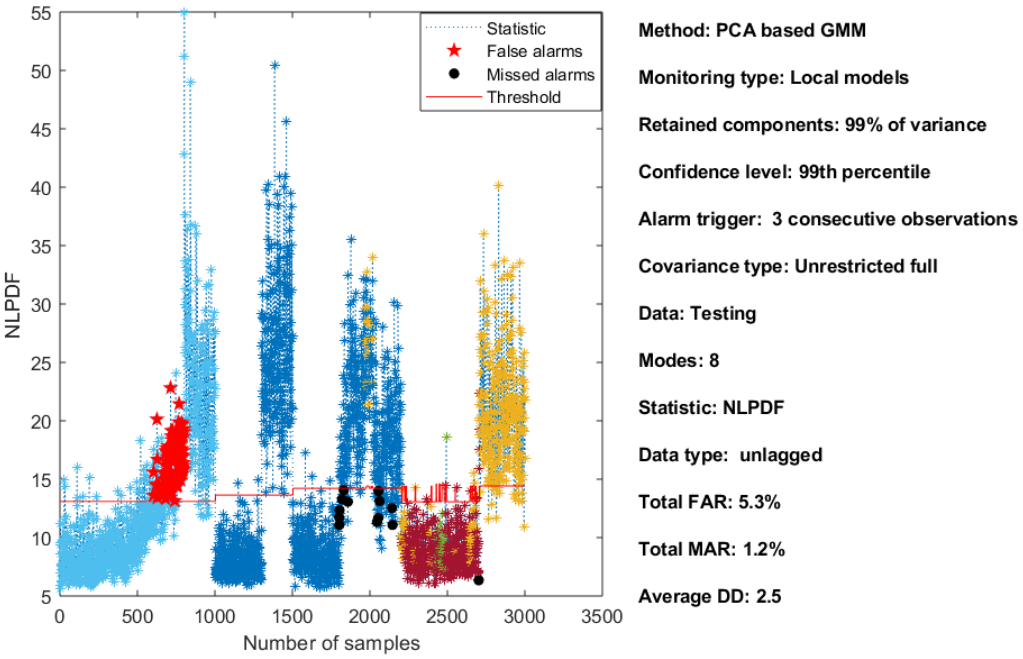


Figure 5.39: NLPDF statistics for Fault C07 of PCA-based GMM. (See Appendix H for further analysis of sample results and description of Fault C08).

In summary, the results give guidance supporting APCA-based GMM as a better performer for processes that may be multimodal and with drift behaviours which is a likely industry scenario.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

This work focused on contributing to the improvement of process monitoring by the implementation of adaptive monitoring methods. To this end, monitoring with unimodal and multimodal approaches, i.e. PCA and GMM were the baseline methods considered. A thorough comparison of PCA and APCA was successfully done using the TE process and simulated unimodal cases from the non-isothermal CSTR process.

The two APCA methods, MWPCA, and RePCA were subsequently thoroughly compared using simulated case studies from the non-isothermal CSTR. Thereafter, various heuristics for when to update monitoring models for APCA methods were also considered.

The limitations of PCA techniques with multimodal approaches established the need for GMM development. Model development for GMM application was then assessed by considering the preprocessing, covariance shapes and model selection criteria that provided optimal clustering results. The impact of implementing various GMM monitoring approaches on performance was ascertained in the case of the various factors pre-listed and also the global and local monitoring approaches. All the analysis was done using the TE process and simulated unimodal and multimodal data from the non-isothermal CSTR.

The final part of the work provided an extension of APCA to GMM that once more considered combining APCA methods (specifically MWPCA) with GMM. The developed method provides a more robust monitoring approach and this was assessed by comparison with the PCA and APCA methods. The performance of the APCA-based GMM was established using the TE process and some simulated non-stationary multimodal processes of the CSTR.

The conclusions drawn from the above-listed investigations are detailed under the ensuing sections.

6.1 PCA vs APCA

From the results of the comparative study (see Section 5.2.1), PCA and APCA achieve substantially equal performance in detecting faults for stationary processes. APCA has the extra potential for coping with process non-stationarity. Implementation of APCA in a non-optimal way (i.e. using sub-optimal update techniques, z values, among others), however, could lead to missed detections; the chances of missing out on such faults is less likely with the complementary implementation of the two monitoring statistics (i.e. SPE and T^2).

6.2 MWPCA vs RePCA

MWPCA outperforms RePCA in fault detection (see Section 5.2.2) mainly due to the latter's slow response and adaptation. A relatively large window size of MWPCA would make it approach the insensitivity to changes experienced by RePCA. MWPCA on the other end run the risk of incorporating faults if the initial data window does not capture the dynamism of the 'true' process.

6.3 Model adaptation methods for APCA

All model update methods (apart from the update method which allows the update of the model for the case when an alarm is not triggered) achieve substantially similar results when implemented at their best hyperparameter values (see Section 5.2.2.2). The proposed model adaptation method (UM-4) provides an all-around robust performance than the other update methods. The chances of an observation exceeding the threshold for only one of the statistics (SPE and T^2) indicating a fault is unlikely when an alarm is not triggered. The T^2 is also less sensitive to drift than the SPE statistic.

Also, update methods that depend on z consecutive observations for model update generally require higher z values as the confidence level of thresholds increase.

6.4 PCA techniques vs GMM

GMM outperforms PCA in fault detection for multimodal processes (see Section 5.3.2.2). Both methods have similar performance ability in a unimodal case. The T^2 statistic of the PCA results shows a close comparison to the probability density function (PDF) of the GMM results.

6.5 Data types for GMM clustering

Clustering in the score space allows the fitting of diagonal covariance matrices which are more parsimonious than full covariance matrices (see Section 5.3.3). Raw data and normalized data perform better when a full covariance matrix is used rather than diagonal matrices. Normalized data based GMM outperforms the raw data in this case. Normalized data also outperforms PCA scores when a non-negligible variance is lost. The reverse (i.e. PCA scores outperforms the normalized data) is true when 99-100% variance is retained.

An added advantage is then provided by the PCA scores by the usage of parsimonious models. Losing a reasonable amount of variance in the retained PCs would require fitting a full covariance matrix to the PCA scores for satisfactory performance.

Forcing the 'wrong' covariance on a data type would require a high number of modes to appropriately describe the original modes. Also, monitoring performance deteriorates with a wrong fitted covariance type.

6.6 Model selection criteria (AIC vs BIC)

BIC has better consistency than the AIC in reproducing the same results which are a better fit to the real number of clusters than that for the AIC (see Section 5.3.2). Also, both methods perform better in specifying the number of clusters in the score space than in the raw and normalized approach.

6.7 GMM monitoring approach (Global and local models)

Local models monitoring is superior to global model monitoring (see Section 5.3.3). The same performance is achieved if one cluster is identified since the global approach involves weighting the individual cluster probabilities–of which is just one.

6.8 APCA-based GMM

APCA-based GMM outperforms all the adaptive and multimodal methods when non-stationary multimodal processes are involved (see Section 5.3.4). The same performance is guaranteed in an adaptive unimodal case as APCA when optimally employed. Also, the same performance is guaranteed under GMM when the same factors are considered (performance sometimes is better for APCA-based GMM) for multimodal situations only.

6.9 Recommendations

The following recommendations are consequently made based on the conclusions drawn above:

1. For fault detection in unimodal case, APCA should be considered in the case where PCA would otherwise be deployed, since it provides the extra advantage of adaptivity for non-stationary processes, which are typical in industry.

2. For adaptive process monitoring, MWPCA should be favoured over RePCA.

3. For heuristics to update a model, the limits of the monitoring model would be challenged from time to time as processes drift. It is most suitable to use a high threshold in order to limit false alarms. Also, it is unlikely that a fault is present when a single observation's SPE or T^2 statistic (not both) exceeds the threshold, unless multiple such observations are consecutive, which would then trigger an alarm by the proposed update method. Hence, the proposed update method should be preferred.

4. For the multimodal case, GMM performs better than PCA techniques and hence should be preferred. It would do no harm to fit GMM to a unimodal process, hence GMM should be preferred overall.

5. For GMM model development, the PCA score space should be preferred since it could provide an additional advantage of fitting more parsimonious model as well as a better monitoring performance ability. The GMM should use a full covariance type if a significant amount of the variance is to be lost (i.e. below 95%) for the scores.

6. The BIC should be preferred over the AIC in model selection.

7. For the implementation of GMM, the use of local monitoring should be preferred over the global approach since the global threshold like the PCA methods uses a unified threshold over all clusters.

8. For an overall robust fault detection approach, APCA-based GMM should be preferred with the proposed model update approach. Also, quicker periodic retraining of the GMM model is recommended (i.e. when at most fifty times the number of process variables are incorporated).

6.10 Recommendations for future work

Some identified areas of interest in this work that would be suitable for future work are subsequently presented below.

1. Establishing the rate of drift which is acceptable (i.e. not a fault). This is to curb the probability of missing out on a slow developing fault because drift faults also occur.

2. Improvement in the computation aspect of implementing the developed approaches must be considered. Such work would involve how to update existing GMM model parameters instead of retraining from scratch. Also, the cluster assignments for each observation need to be modelled as it is unrealistic to move back and forth between operating modes in short succession.

REFERENCES

Akaike, H., 1973. Information theory and an extension of the maximum likelihood principle. In *International Symposium on Information Theory*. Budapest. Academiai Kiado, pp. 267–281. doi: 10.1007/978-1-4612-1694-0.

Alcala, C.F. and Qin, S.J., 2010. Reconstruction-based contribution for process monitoring with kernel principal component analysis. *Industrial and Engineering Chemistry Research*, *49*(17), pp.7849-7857.

Alkaya, A. and Eker, İ., 2011. Variance sensitive adaptive threshold-based PCA method for fault detection with experimental application. *ISA Transactions*, *50*(2), pp.287-302.

Ayech, N., Chakour, C. and Harkat, M.F., 2012. New adaptive moving window PCA for process monitoring. *Fault Detection, Supervision and Safety of Technical Processes*, *8*(1), pp.606-611.

Balakrishnama, S. and Ganapathiraju, A., 1998. Linear discriminant analysis - a brief tutorial. *Institute for Signal and Information Processing*, *18*, pp.1-8.

Bellman, R., 1957. Dynamic programming. *Princeton University Press*, *89*, p.92.

Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A., 2013. A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, *34*(3), pp.483-519.

Bozdogan, H., 1994. Mixture-model cluster analysis using model selection criteria and a new informational measure of complexity. In *Proceedings of the First US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach* (pp. 69-113). Springer, Dordrecht.

Bozdogan, H., 2000. Akaike's information criterion and recent developments in information complexity. *Journal of Mathematical Psychology*, *44*(1), pp.62-91.

Cao, L.J., Chua, K.S., Chong, W.K., Lee, H.P. and Gu, Q.M., 2003. A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. *Neurocomputing*, 55(1-2), pp.321-336.

Cattell, R.B., 1966. The scree test for the number of factors. *Multivariate Behavioral Research*, *1*(2), pp.245-276.

Celeux, G. and Govaert, G., 1995. Gaussian parsimonious clustering models. *Pattern Recognition*, *28*(5), pp.781-793.

Choi, S.W., Martin, E.B., Morris, A.J. and Lee, I.B., 2006. Adaptive multivariate statistical process control for monitoring time-varying processes. *Industrial and Engineering Chemistry Research*, *45*(9), pp.3108-3118.

Choi, S.W., Park, J.H. and Lee, I.B., 2004. Process monitoring using a Gaussian mixture model via principal component analysis and discriminant analysis. *Computers and Chemical Engineering*, *28*(8), pp.1377-1387.

Dayal, B.S. and MacGregor, J.F., 1997. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, *7*(3), pp.169-179.

Dempster, A.P., Laird, N.M. and Rubin, D.B., 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)*, pp.1-38.

Downs, J.J. and Vogel, E.F., 1993. A plant-wide industrial process control problem. *Computers and Chemical Engineering*, *17*(3), pp.245-255.

Erar, B., 2011. Mixture model cluster analysis under different covariance structures using information complexity. Master's Thesis, University of Tennessee.

Figueiredo, M.A.T. and Jain, A.K., 2002. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), pp.381-396.

Hartigan, J.A. and Wong, M.A., 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), pp.100-108.

He, T., Xie, W.R., Wu, Q.H. and Shi, T.L., 2006. Process fault detection and diagnosis based on principal component analysis. In *Machine Learning and Cybernetics, 2006 International Conference on Cybernetics* (pp. 3551-3556). IEEE.

Helfand, E. and Stillinger Jr, F.H., 1968. Critical Solution Behavior in a Binary Mixture of Gaussian Molecules. *The Journal of Chemical Physics*, *49*(3), pp.1232-1242.

Hira, Z.M. and Gillies, D.F., 2015. A review of feature selection and feature extraction methods applied on microarray data. *Advances in Bioinformatics*, *2015*.

Hotelling, H., 1931. The economics of exhaustible resources. *Journal of Political Economy*, *39*(2), pp.137-175.

Huang, H., Luo, F., Liu, J. and Yang, Y., 2015. Dimensionality reduction of hyperspectral images based on sparse discriminant manifold embedding. *ISPRS Journal of Photogrammetry and Remote Sensing*, *106*, pp.42-54.

Humberstone, M., Wood, B., Henkel, J. and Hines, J.W., 2012. Differentiating between expanded and fault conditions using principal component analysis. *Journal of Intelligent Manufacturing*, *23*(2), pp.179-188.

Ipek, H., Ankara, H. and Ozdag, H., 1999. The application of statistical process control. *Minerals Engineering*, *12*(7), pp.827-835.

Jackson, J.E. and Mudholkar, G.S., 1979. Control procedures for residuals associated with principal component analysis. *Technometrics*, *21*(3), pp.341-349.

Jeng, J.C., 2010. Adaptive process monitoring using efficient recursive PCA and moving window PCA algorithms. *Journal of the Taiwan Institute of Chemical Engineers*, *41*(4), pp.475-481.

Jolliffe, I.T., 2002. *Principal component analysis*. 2nd ed. New York, NY: Springer-Verlag.

Kourti, T., 2002. Process analysis and abnormal situation detection: from theory to practice. *IEEE Control Systems*, *22*(5), pp.10-25.

Kruger, U. and Xie, L., 2012. *Statistical Monitoring of Complex Multivatiate Processes: With Applications in Industrial Process Control*. John Wiley and Sons.

Ku, W., Storer, R.H. and Georgakis, C., 1995. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *30*(1), pp.179-196.

Li, W., Yue, H.H., Valle-Cervantes, S. and Qin, S.J., 2000. Recursive PCA for adaptive process monitoring. *Journal of Process Control*, *10*(5), pp.471-486.

Mansouri, M., Nounou, M., Nounou, H. and Karim, N., 2016. Kernel PCA-based GLRT for nonlinear fault detection of chemical processes. *Journal of Loss Prevention in the Process Industries*, *40*, pp.334-347.

Martinez, A.M. and Zhu, M., 2005. Where are linear feature extraction methods applicable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(12), pp.1934-1944.

Mason, R.L. and Young, J.C., 2002. *Multivariate statistical process control with industrial applications* (Vol. 9). SIAM.

Mnassri, B. and Ouladsine, M., 2015. Reconstruction-based contribution approaches for improved fault diagnosis using principal component analysis. *Journal of Process Control*, *33*, pp.60-76.

Morais, M.C., Okhrin, Y., Pacheco, A. and Schmid, W., 2008. EWMA charts for multivariate output: some stochastic ordering results. *Communications in Statistics - Theory and Methods*, *37*(16), pp.2653-2663.

Owen, J.A., 2014. Principal component analysis: Data reduction and simplification. *McNair Scholars Research Journal*, *1*(1), p.2.

Paalanen, P., 2004. Bayesian classification using Gaussian mixture model and EM estimation: Implementations and comparisons. *Information Technology Project*, Lappeenranta University of Technology.

Peng, X., Tang, Y., Du, W. and Qian, F., 2017. Multimode process monitoring and fault detection: a sparse modeling and dictionary learning method. *IEEE Transactions on Industrial Electronics*, *64*(6), pp.4866-4875.

Portnoy, I., Melendez, K., Pinzon, H. and Sanjuan, M., 2016. An improved weighted recursive PCA algorithm for adaptive fault detection. *Control Engineering Practice*, *50*, pp.69-83.

Posada, D. and Buckley, T.R., 2004. Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests. *Systematic Biology*, *53*(5), pp.793-808.

Pudil, P., Novovičová, J. and Kittler, J., 1994. Floating search methods in feature selection. *Pattern Recognition Letters*, *15*(11), pp.1119-1125.

Qin, S.J., 1998. Recursive PLS algorithms for adaptive data modeling. *Computers and Chemical Engineering*, *22*(4-5), pp.503-514.

Rashid, M.M. and Yu, J., 2012. A new dissimilarity method integrating multidimensional mutual information and independent component analysis for non-Gaussian dynamic process monitoring. *Chemometrics and Intelligent Laboratory Systems*, *115*, pp.44-58.

Russell, E., Chiang, L. and Braatz, R., 2000. *Data-driven methods for fault detection and diagnosis in chemical processes*. 1st ed. London [etc.]: Springer.

Russell, E.L., Chiang, L.H. and Braatz, R.D., 2000. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *51*(1), pp.81-93.

Schmitt, E., Rato, T., De Ketelaere, B., Reis, M. and Hubert, M., 2016. Parameter selection guidelines for Adaptive PCA-based control charts. *Journal of Chemometrics*, *30*(4), pp.163-176.

Schwarz, G., 1978. Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), pp.461-464.

Serradilla, J., Shi, J.Q. and Morris, A.J., 2011. Fault detection based on Gaussian process latent variable models. *Chemometrics and Intelligent Laboratory Systems*, *109*(1), pp.9-21.

Shewhart, W.A., 1925. The application of statistics as an aid in maintaining quality of a manufactured product. *Journal of the American Statistical Association*, *20*(152), pp.546-548.

Shlens, J., 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Slišković, D., Grbić, R. and Hocenski, Ž., 2012. Multivariate statistical process monitoring. *Tehnicki Vjesnik-Technical Gazette*, *19*(1), pp.33-41.

Stone, J.V., 2002. Independent component analysis: an introduction. *Trends in Cognitive Sciences*, *6*(2), pp.59-64.

Tien, D.X., 2005. *Moving PCA for process fault detection - a performance and sensitivity, study*. Master's Thesis, National University of Singapore.

Tresp, V., 2001. Mixtures of Gaussian processes. In *Advances in Neural Information Processing Systems* (pp. 654-660).

Verónica Bolón, A.B., Amparo, M. and Sánchez, C.N., 2017. *Artificial Intelligence: Foundations, Theory, and Algorithms Feature Selection for High-Dimensional Data*. Springer.

Vines, S.K., 2000. Simple principal components. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *49*(4), pp.441-451.

Vlassis, N. and Likas, A., 2002. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, *15*(1), pp.77-87.

Voegtlin, T., 2004. Recursive PCA and the structure of time series. In *IEEE International Conference on Neural Networks - Conference Proceedings*, pp. 1893–1897.

Wang, X., Kruger, U. and Irwin, G.W., 2005. Process monitoring approach using fast moving window PCA. *Industrial and Engineering Chemistry Research*, *44*(15), pp.5691-5702.

Wang, Y., Seo, H. and Jeon, S., 2008. Automatic transition detection of segmented motion clips using PCA-based GMM method. In *Proceedings of the 2008 International Conference on Cyberworlds* (pp. 567-572). IEEE. doi: 10.1109/CW.2008.92.

Wold, S., 1994. Exponentially weighted moving principal components analysis and projections to latent structures. *Chemometrics and Intelligent Laboratory Systems*, *23*(1), pp.149-161.

Woodall, W.H. and Adams, B.M., 1993. The statistical design of CUSUM charts. *Quality Engineering*, *5*(4), pp.559-570.

Xia, L., Chu, J. and Geng, Z., 2013. Process monitoring based on improved recursive PCA methods by adaptive extracting principal components. *Transactions of the Institute of Measurement and Control*, *35*(8), pp.1024-1045.

Xie, X. and Shi, H., 2012. Dynamic multimode process modeling and monitoring using adaptive Gaussian mixture models. *Industrial and Engineering Chemistry Research*, *51*(15), pp.5497-5505.

Yoon, S. and MacGregor, J.F., 2001. Fault diagnosis with multivariate statistical models part I: using steady state fault signatures. *Journal of Process Control*, *11*(4), pp.387-400.

Yu, J. and Qin, S.J., 2008. Multimode process monitoring with Bayesian inference-based finite Gaussian mixture models. *AIChE Journal*, *54*(7), pp.1811-1829.

Yu, J., 2011. Fault detection using principal components-based Gaussian mixture model for semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, *24*(3), pp.432-444.

Zhao, S.J., Xu, Y.M. and Zhang, J., 2004. A multiple PCA model based technique for the monitoring of processes with multiple operating modes. *Computer Aided Chemical Engineering*, *18*, pp.865-870.

APPENDIX A: NON-ISOTHERMAL CSTR PROGRAM

Appendix A provides information on the description and development of the non-isothermal CSTR simulator used in the generation of simulated data. The Simulink model development, as well as the simulator software, is presented.

A.1 Overview

The non-isothermal continuous stirred tank reactor (CSTR) model is a popular model used by several authors (Choi et al., 2005; Alcala and Qin, 2010; Jeng, 2010; Mnassri, El Adel and Ouladsine, 2015; Mansouri et al., 2016) in data generation for fault detection, fault diagnosis and other relevant purposes. Although it's mentioned in the above mentioned works, there is no readily available model for the simulator. Also, the different works (mentioned earlier) model different fault scenarios for the specific purposes without a generic simulator created.

This work builds a Simulink model and a MATLAB program that uses graphical user interfaces (GUI) for simulating the Non-isothermal CSTR which was first developed by Yoon and MacGregor (2001) in fault detection. The model incorporates different versions of the non-isothermal CSTR into a combined model that simulates a unimodal or multimodal operation data and twenty five process faults as well as closed and open loop operation.

The subsequent sections provide a description of the reactor model (presented in Section A.2), Simulink modelling of the CSTR (presented in Section A.3) and the GUI of the MATLAB program (presented in Section A.4).

A.2 Reactor description

See Section 5.1.1 for reactor description.

A.3 Simulink Model

The primary CSTR model is built using the material and energy balances provided in Equations **5-2** and **5-3**. The relevant model parameters and modelling of process faults and multimodal operation and incorporated other process changes are presented in the subsequent sub-sections.

A.3.1 Parameters and constants

The Simulink model uses simulation parameters and simulation conditions adopted from Yoon and MacGregor (2001). The descriptions and values of the relevant parameters and initial conditions are respectively presented in Table A.1 and Table A.2.

Table A.1: Summary of reactor parameters.

Notation	Parameter	Value	Unit
V	Reaction mixture volume	1	m^3
ρ	Density of reaction mixture	10^6	g/m^3
ρ_c	Density of coolant	10^6	g/m^3
c_p	Specific heat capacity of the reaction mixture	1	$cal/(gK)$
c_{pc}	Specific heat capacity of the coolant	1	$cal/(gK)$
ΔH_r	Heat of reaction	-1.3×10^7	$cal/(gKmol)$
k_o	Pre-exponential kinetic constant	10^{10}	min^{-1}
E/R	Activation energy/ideal gas constant	8330	K
q	inner film heat transfer coefficient	-1.678×10^6	$cal/minK$
b	Empirical relationship between heat transfer and coolant flowrate constant	0.5	

Table A.2: Summary of reactor initial conditions.

Notation	Parameter	Value	Unit
F_s	Flowrate of solvent	15	m^3/min
T_i	Reactor feed inlet temperature	370	K
T_c	Coolant temperature	365	K
C_s	Concentration of solvent	0.3	$kmol/m^3$
C_a	Concentration of reactant	19.1	$kmol/m^3$
F_c	Flowrate of coolant	15	m^3/min
T	Temperature of reactor	368.25	K
F_a	Flowrate of the reactant	0.1	m^3/min
C	Concentration of reactor	0.8	$kmol/m^3$

A closed loop simulation employs a PI control system with the controller parameters adopted from (Yoon and MacGregor, 2001) as shown in Table A.3.

Table A.3: Summary of parameters of the PI controllers.

Notation	Parameter	Value
$Kc(T)$	Temperature controller gain	-1.5
$Kc(C_a)$	Concentration controller gain	0.4825

Notation	Parameter	Value
$\tau_I(T)$	Temperature integral time	5
$\tau_I(C_a)$	Concentration integral time	2

Also, all process disturbances are modelled as first order autoregressive (AR) processes as shown in Equation **A-1**.

$$y_{i+1} = \phi y_i + e \qquad \textbf{A-1}$$

Here, ϕ is the *AR* coefficient and e is process noise and given as $e \sim N(0, \sigma_e^2)$ of an input. Table A.4 shows the *AR* coefficients and process noise summary for the respective process inputs adopted from (Yoon and MacGregor, 2001).

Table A.4: Autoregressive model parameters of input variables.

Notation	ϕ	e
F_s	0.9	0.19×10^{-2}
T_i	0.9	0.475×10^{-1}
T_c	0.9	0.475×10^{-1}
C_s	0.5	1.875×10^{-3}
C_a	0.9	0.475×10^{-1}
F_c	0.9	1.0×10^{-2}
F_a	0.9	0.19×10^{-2}

All measured variables are contaminated with white Gaussian noise e_m and given as $e_m \sim N(0, \sigma_m^2)$. The respective σ_m^2 values adopted from Yoon and MacGregor (2001) for the measured variables are presented in Table A.5.

Table A.5: Measurement noise of process variables.

Notation	σ_m^2
F_s	4.0×10^{-5}
T_i	2.5×10^{-3}
T_c	2.5×10^{-3}
C_s	2.5×10^{-5}
C_a	0.19×10^{-2}

Notation	σ_m^2
F_c	1.0×10^{-2}
T	4.0×10^{-4}
F_a	4.0×10^{-6}
C	2.5×10^{-4}

A.3.2 Simulink modelling

Since various works employed different approaches in modelling of different process changes, it is relevant to provide an idea of all the relevant changes modelled in this work. Key among the modelling approaches in the Simulink model are multimode operations, reaction drift and process faults. For all the stated process changes, options are available to be activated or deactivated at user specified times.

Reaction drift

Reaction drift is attributed to performance degradation as a result of a number of plausible effects during a reaction. Key among such effects are performance degradation due to catalyst poisoning and fouling of cooling coils. This is modelled by introducing the performance coefficient β_r in the reaction rate. The constant value of one for β_r indicates a process with no performance changes. As shown in Figure A.1, a constant value of one is passed to the system until the drift is turned on. The drift is modelled as a ramp, where a negative slope indicates a performance degradation case; and a positive slope for performance improvement. The resultant reaction rate is shown in Equation **A-2**.

$$r = \beta_r k_o e^{-E/RT} C \qquad \textbf{A-2}$$

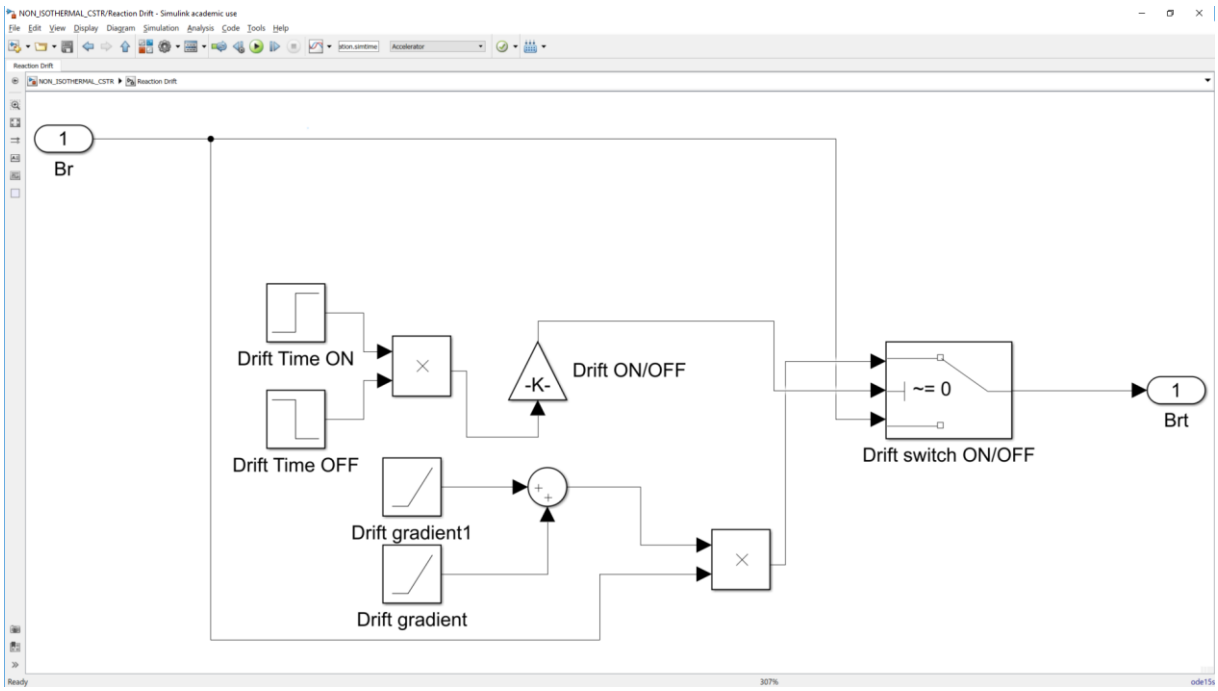


Figure A.1: Reaction drift model.

Multimodal operation

Multiple mode operations are introduced as a change in product specifications. This is modelled as changes in controller set points of the controlled variables. Mode changes are activated by causing step changes in either the concentration or the temperature set points at specific time intervals. The modelling as shown in Figure A.2 presents a view of how the set points are passed to the controller. It can be seen as the primary set point (one) is passed unless set point two is activated, then set point two is passed. For each multimodal option, a total of three set point changes can be effected and consequently four operation modes can be achieved.

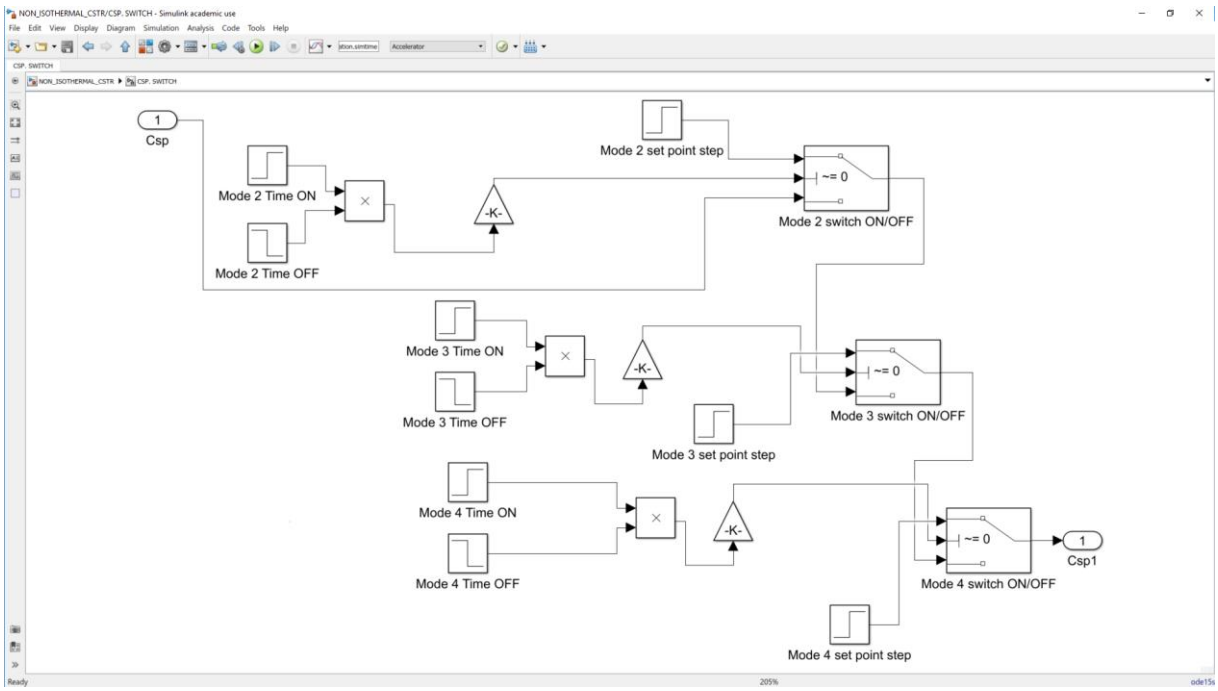


Figure A.2: Mode changes as a step change in concentration controller set point.

Process inputs

As mentioned earlier, all process inputs are modelled as first order AR processes as shown in Figure A.3. Changing the AR coefficients to a value of 1 converts the input process from an AR to a random walk process.

The overall model involves a tapped delay in the variable and the relevant AR coefficient, measurement noise and the constant term that makes sure the input does not have zero mean. The associated switch is an implementation of the step fault in the process input at a specific time. The general process faults are described in the subsequent section.

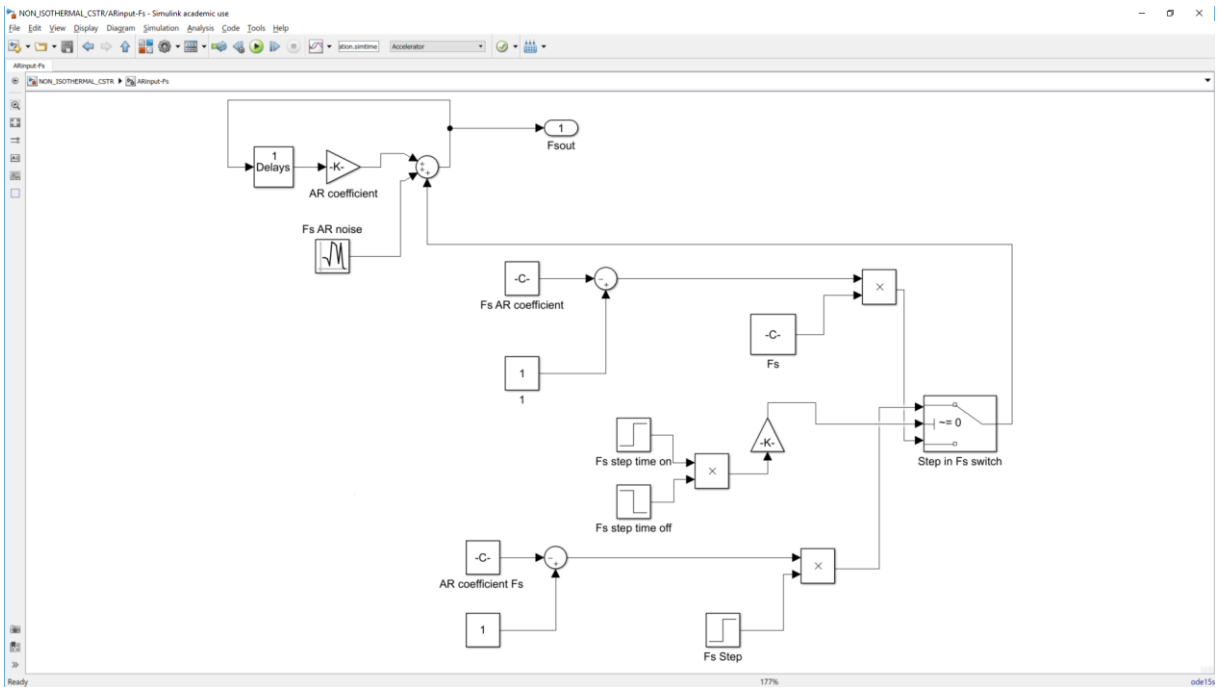


Figure A.3: An autoregressive process model for Fs input.

Process faults

The model incorporates three distinct faults, namely sensor drift faults, sensor bias faults, and process input step faults. While sensor drift and sensor bias faults are modelled for all the nine measured variables, the input step faults are available for the seven process inputs. Two of the step input faults, the step in F_c and F_a, can only be implemented when the temperature controller and concentration controllers are respectively off.

The sensor drift faults as shown in Figure A.4, are ramps with specific times to be implemented. Sensor bias fault and sensor drift fault in the same variable can only be implemented one after the other. If both sensor bias and drift fault in the same variable run simultaneously, only the sensor bias fault is implemented.

Classification of the process faults as simple or complex depends on the way the fault propagates in the system. A simple fault is reflected in only one variable, while a complex fault is reflected in an impact on other variables (Yoon and MacGregor, 2001; Mnassri, El Adel and Ouladsine, 2015). An example of a complex process fault is a step input in C_s which impacts the reactor concentration and forces the reaction of F_a in a closed loop process. A sensor bias or drift fault in C_s is classified as a simple fault because it only affects the variable C_s. Table A.6 provides a summary of all the modelled faults and their respective classifications. The faults registered as complex/simple refer to faults which are complex when the controllers are activated and simple when the controllers are deactivated.

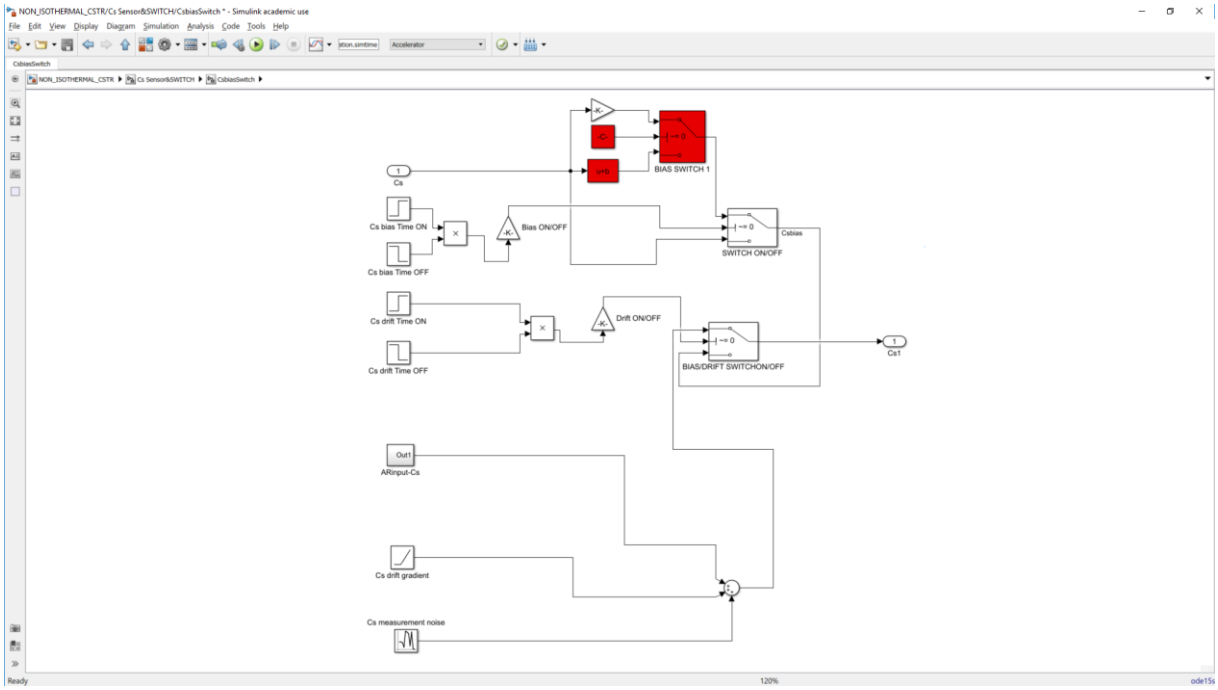


Figure A.4: Sensor bias/drift fault in Cs model (red highlights addition of bias as a fixed value).

Table A.6: Summary of fault descriptions as simple or complex faults.

Variable	Input step	Sensor Drift	Sensor Bias
F_s	complex	simple	simple
T_i	complex	simple	simple
T_c	complex	simple	simple
C_s	complex	simple	simple
C_a	complex	simple	simple
F_c	complex	simple	simple
F_a	complex	simple	simple
T		complex/simple	complex/simple
C		complex/simple	complex/simple

The overall process of complex and simple faults modelling for the sensors can be seen as shown in Figure A.5 where the sensor measurements are passed into the process for variables highlighted green and red for not passed. The difference is that when sensor faults affect only the measured values, they are classified as simple faults. In the case where the variable is controlled, there is reaction from the controller that forces response from other variables and the fault is then classified as a complex one because it impacts other variables.

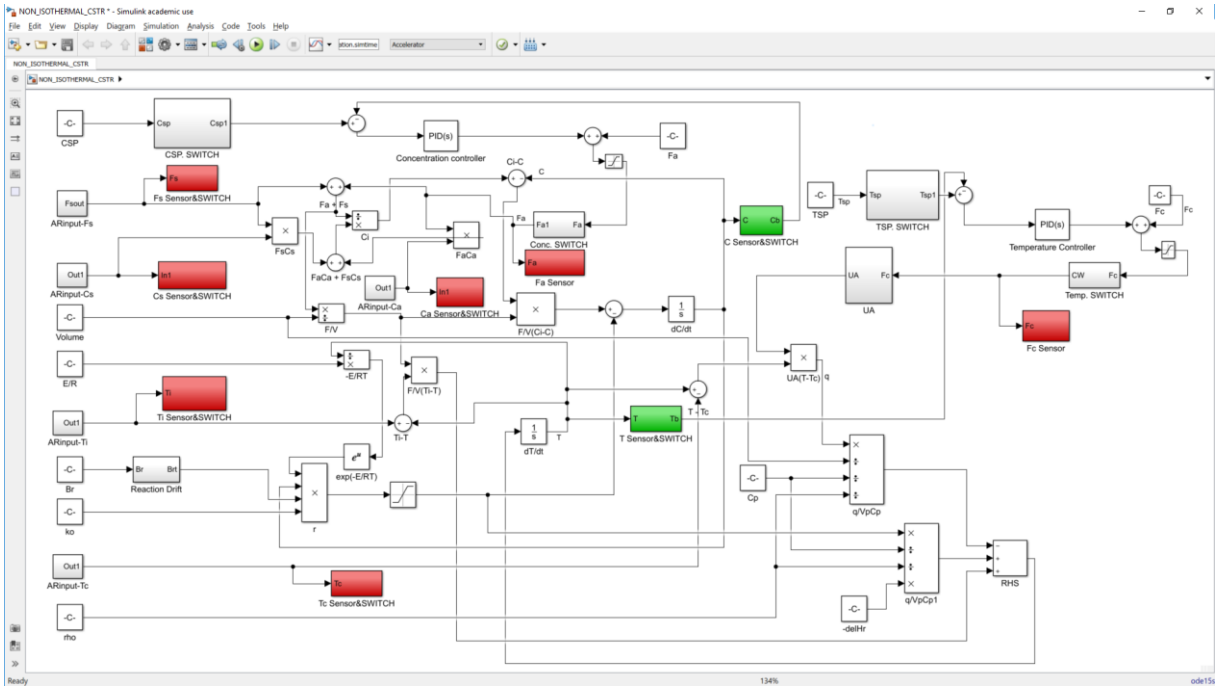


Figure A.5: Complex/simple sensor drift/bias faults. (Red colour refers to simple faults while green highlights complex faults).

Controllers

The model provides concentration and temperature controllers which are PI controllers that can be implemented in a closed loop simulation. The controller implementations are such that the allowable maximum number of inactive controllers is two for unimodal, and one for multimodal. This is a result of the need for a controller to cause the mode changes as described by the change in controller set points.

Seeds

All process and measurement noises make use of a random number generator, with the seeds of the random number generators being unique for the individual noises as well as each simulation.

A.4 MATLAB program and Graphical user interfaces (GUI)

The relevant simulation details required are the simulation parameters, simulation and sampling time and controller details for closed loop simulation. Another required information is for faults and their respective times in test data generation.

The various information required are captured in each user interface are presented in the subsequent sections. Generally, all interactive fields at each point have green backgrounds and

can either be a button, drop-down menu, a text field (for entering a value) and table (with editable fields). Anything with a grey background is just a placeholder and perform no action.

Please make use of the 'close' options provided in the program to exit figures, only close the MATLAB figures using the MATLAB in-built button when it is the only option. A temporary link to the simulator is available here (https://prince-addo.github.io/#Projects).

A.4.1 Home

The 'Home page' (shown in Figure A.6) is the first page after running the 'Reactor_simulator' script. It provides two buttons with options to specify reactor parameters from scratch or load the default parameters and edit the required fields. A 'reaction parameters' message box provides information about the button options.

Clicking on either button opens the respective pages for load default parameters (presented in Section A.4.2) and the new reactor parameters (presented in Section A.4.3). To exit the home page, the figure must be closed.

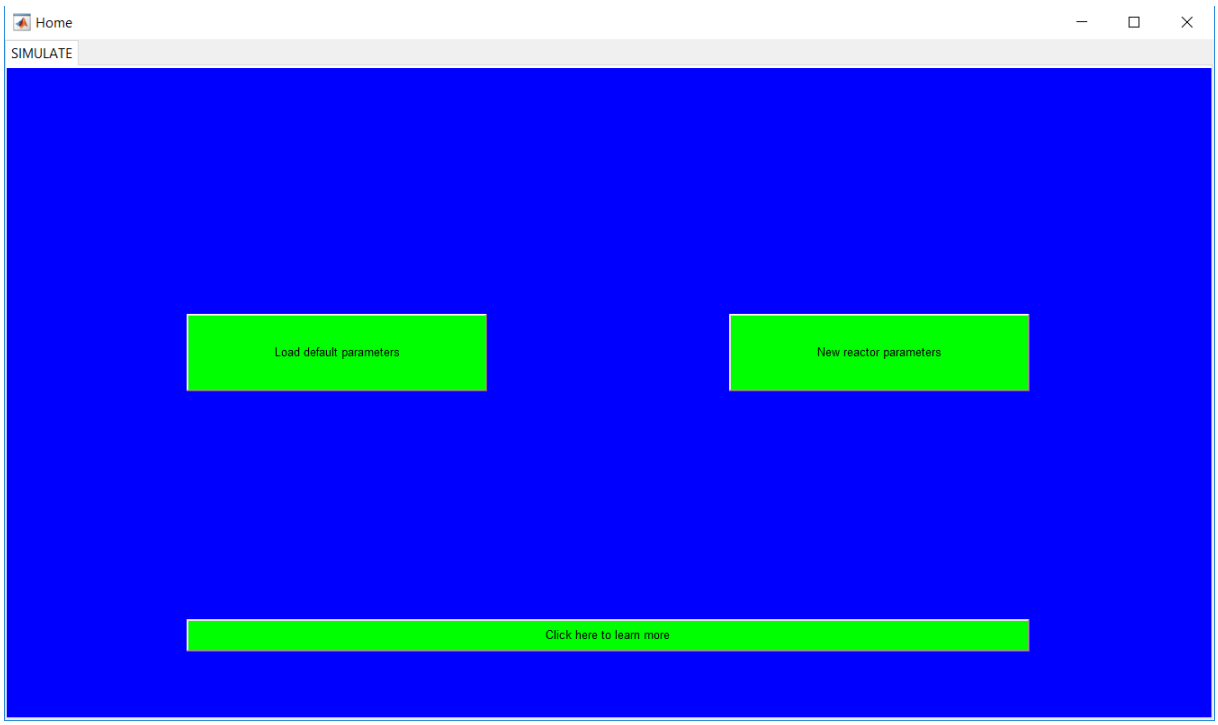


Figure A.6: Home page.

A.4.2 Load default parameters

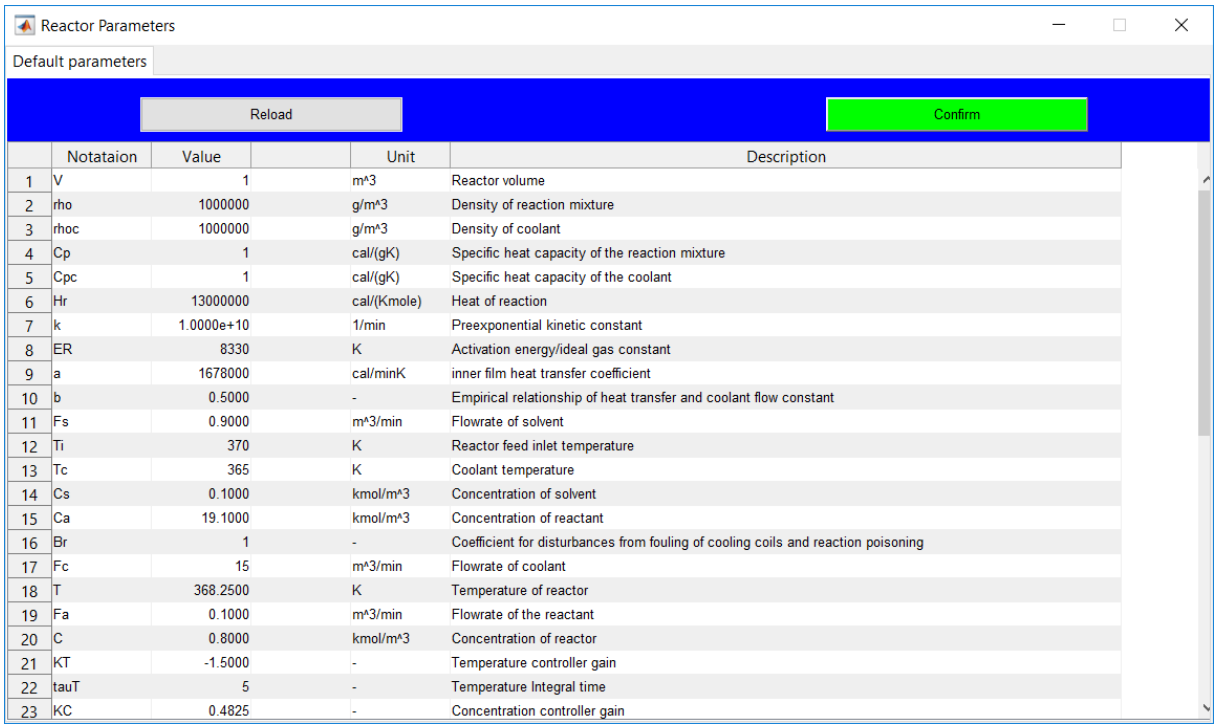
Selecting the option to load default parameters opens up the 'Reactor Parameters' page shown in Figure A.7 which contains a table preloaded with parameters values described in the previous sections and an editable 'value' column. The only form of data validation for the values are

checks for numeric inputs in the fields. This involves resetting the changed value to pre-existing value if conditions are not met and a warning sign that pops up to alert the user.

Inputs are saved by clicking elsewhere in the page after keying in a value or alternatively hitting the return/enter key.

- The 'Reload' button restores the table to its initial state and discards any user input.
- The 'Confirm' button saves the reactor parameters and continues to the 'Simulate types' page (presented in Section A.4.4). A 'Specify parameters' message box is associated with the page provides and information about the buttons.

Although values for the parameters of the PI controllers are required to be provided, the values keyed are not critical if an open loop simulation is needed since options to disable the respective controllers are presented in the 'Simulation specifications' page (presented in Section A.4.10).



	Notataion	Value	Unit	Description
1	V	1	m^3	Reactor volume
2	rho	1000000	g/m^3	Density of reaction mixture
3	rhoc	1000000	g/m^3	Density of coolant
4	Cp	1	cal/(gK)	Specific heat capacity of the reaction mixture
5	Cpc	1	cal/(gK)	Specific heat capacity of the coolant
6	Hr	13000000	cal/(Kmole)	Heat of reaction
7	k	1.0000e+10	1/min	Preexponential kinetic constant
8	ER	8330	K	Activation energy/ideal gas constant
9	a	1678000	cal/minK	inner film heat transfer coefficient
10	b	0.5000	-	Empirical relationship of heat transfer and coolant flow constant
11	Fs	0.9000	m^3/min	Flowrate of solvent
12	Ti	370	K	Reactor feed inlet temperature
13	Tc	365	K	Coolant temperature
14	Cs	0.1000	kmol/m^3	Concentration of solvent
15	Ca	19.1000	kmol/m^3	Concentration of reactant
16	Br	1	-	Coefficient for disturbances from fouling of cooling coils and reaction poisoning
17	Fc	15	m^3/min	Flowrate of coolant
18	T	368.2500	K	Temperature of reactor
19	Fa	0.1000	m^3/min	Flowrate of the reactant
20	C	0.8000	kmol/m^3	Concentration of reactor
21	KT	-1.5000	-	Temperature controller gain
22	tauT	5	-	Temperature Integral time
23	KC	0.4825	-	Concentration controller gain

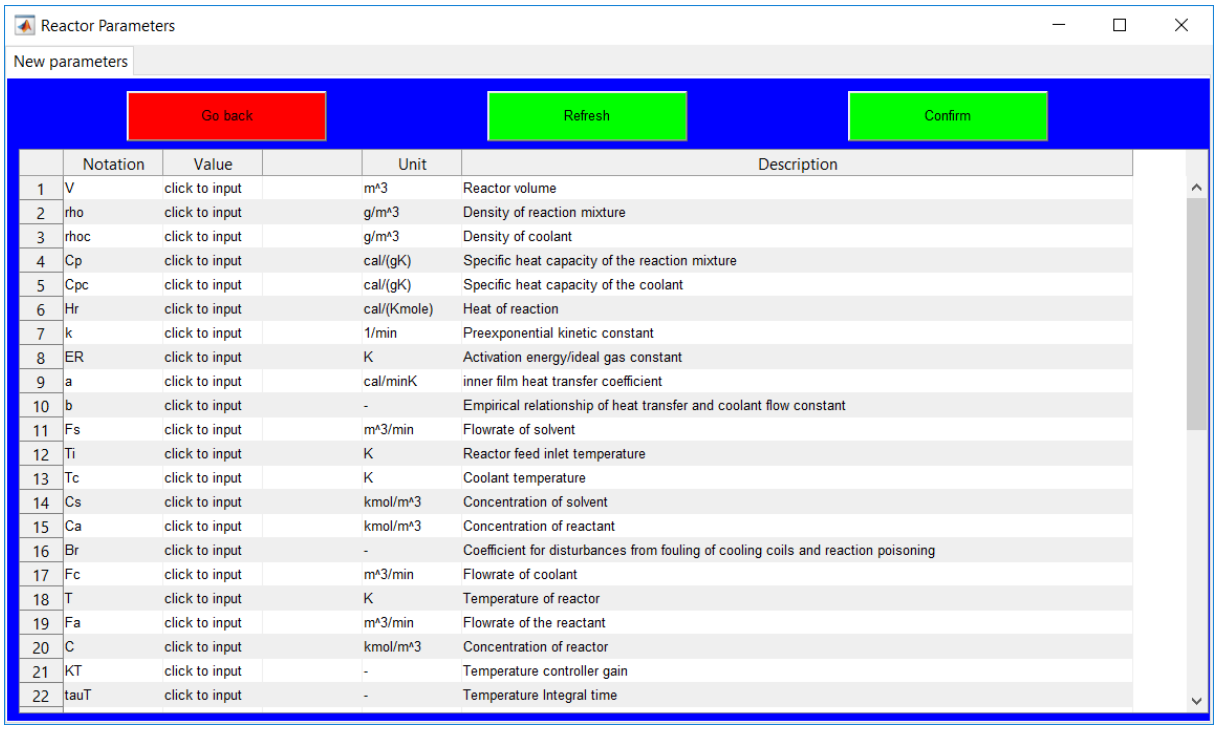
Figure A.7: Reactor parameters page for selecting option to load default parameters.

A.4.3 New reactor parameters

Selecting the option to provide new parameters opens up the 'Reactor Parameters' page with the same description like that for the default parameters but with no preloaded reactor data as shown in Figure A.8. The 'value' column rather contains 'click to input' texts that need to be cleared for specific inputs to be made.

- The ‘Go back’ button closes the existing page and returns the user to the ‘Home’ page.
- The ‘Refresh’ button, discards all changes made and presents the initialized table data.
- Until all the fields are appropriately completed, the ‘confirm’ button remains pops up a warning requesting that all fields must be completed. It, however, continues to the ‘Simulate types’ page if all inputs are validated as correct.

As with the ‘load default parameters’ page, a ‘Specify parameters’ message box is associated with the page provides information about the buttons.



	Notation	Value		Unit	Description
1	V	click to input		m^3	Reactor volume
2	rho	click to input		g/m^3	Density of reaction mixture
3	rhoc	click to input		g/m^3	Density of coolant
4	Cp	click to input		cal/(gK)	Specific heat capacity of the reaction mixture
5	Cpc	click to input		cal/(gK)	Specific heat capacity of the coolant
6	Hr	click to input		cal/(Kmole)	Heat of reaction
7	k	click to input		1/min	Preexponential kinetic constant
8	ER	click to input		K	Activation energy/ideal gas constant
9	a	click to input		cal/minK	inner film heat transfer coefficient
10	b	click to input		-	Empirical relationship of heat transfer and coolant flow constant
11	Fs	click to input		m^3/min	Flowrate of solvent
12	Ti	click to input		K	Reactor feed inlet temperature
13	Tc	click to input		K	Coolant temperature
14	Cs	click to input		kmol/m^3	Concentration of solvent
15	Ca	click to input		kmol/m^3	Concentration of reactant
16	Br	click to input		-	Coefficient for disturbances from fouling of cooling coils and reaction poisoning
17	Fc	click to input		m^3/min	Flowrate of coolant
18	T	click to input		K	Temperature of reactor
19	Fa	click to input		m^3/min	Flowrate of the reactant
20	C	click to input		kmol/m^3	Concentration of reactor
21	KT	click to input		-	Temperature controller gain
22	tauT	click to input		-	Temperature Integral time

Figure A.8: Reactor parameters page for selecting option to load specific parameters.

A.4.4 Simulation types

The ‘Simulation types’ page shown in Figure A.9 provides options to select the data type to generate. A ‘Simulation type’ message box is associated with the page and provides information about the buttons.

- The ‘Training data’ button provides options to simulate data free of faults.
- The ‘Validation data’ button provides the same options as the ‘Training data’ button in generating fault free data.
- The ‘Test data’ button provides the same options as the two other buttons but with an extra information about faults to generate in subsequent pages.
- The ‘Go back’ button returns the user to the ‘Reactor Parameters’ page which is the previous page.

Selecting any of the three data generation buttons on the page opens a pop-up menu that requires the user to specify whether unimodal or multimodal data is to be generated. The options for unimodal/multimodal simulation is presented in Section A.4.5.

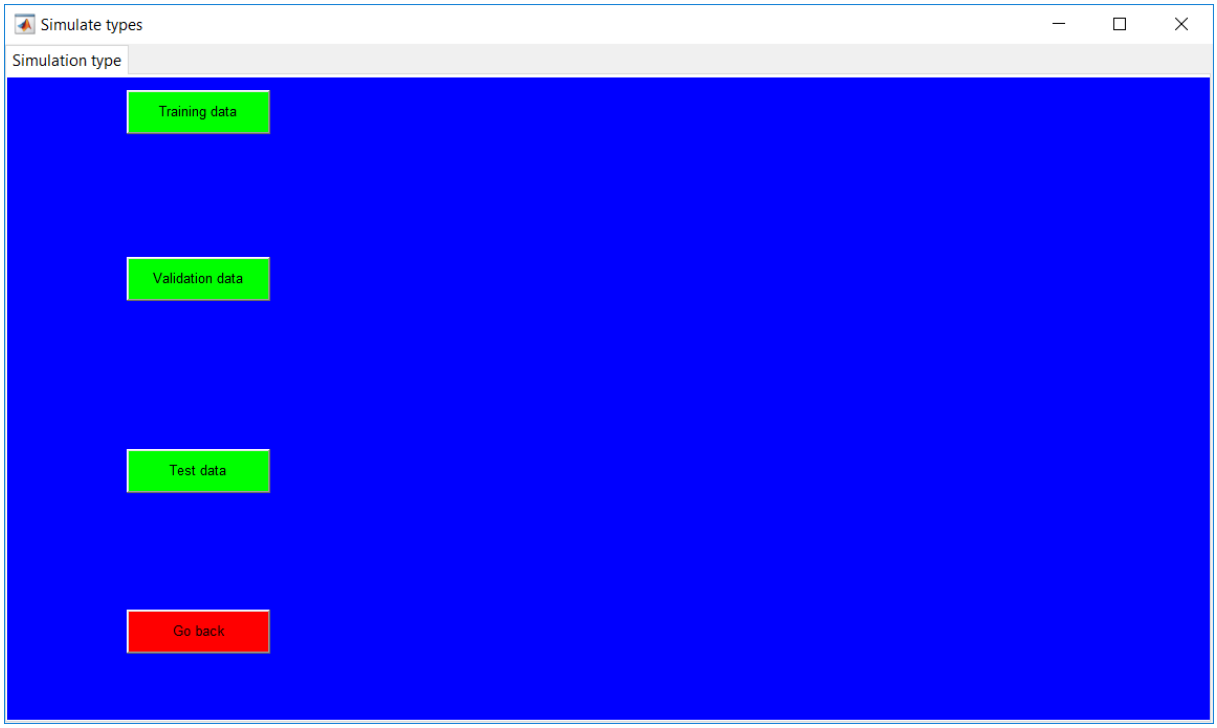


Figure A.9: Simulation type page.

A.4.5 Unimodal / multimodal data

The options to simulate a multimodal or unimodal process is provided in the drop-down list as shown in Figure A.10. Information about the mode types is passed by clicking on the appropriate mode selection from the list. A 'Simulation modes' message box associated with the pop-up menu provides information about the options in the drop-down list of the pop-up menu.

- The unimodal option opens the unimodal section (presented in Section A.4.6).
- The multimodal option also opens up the multimodal section (presented in Section A.4.7)
- The 'Go back' button returns the user to 'Simulate types' page.

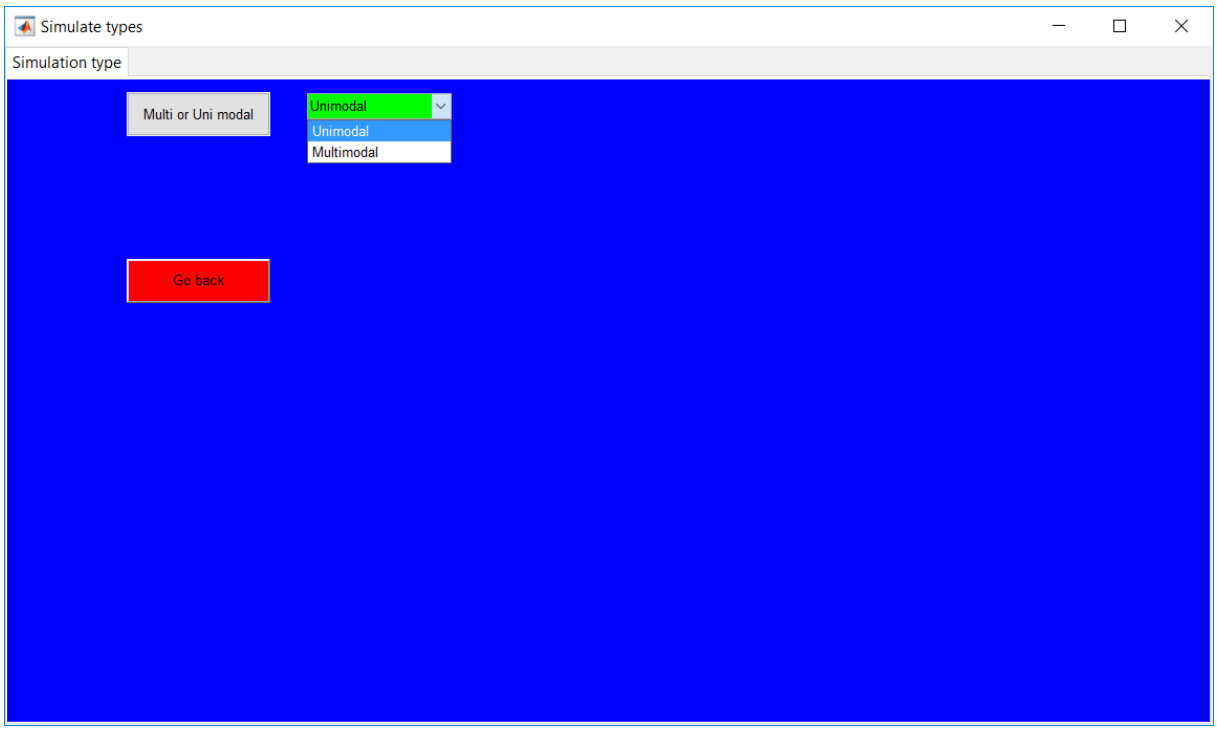


Figure A.10: Unimodal/multimodal pop-up menu.

A.4.6 Unimodal

Selecting the unimodal option from the drop down list opens the green button as shown in Figure A.11 which provides an option to continue.

- The green button proceeds to 'Simulation specifications' page for validation and training data (presented in Section A.4.10).
- The green button proceeds to the 'Fault specifications' page for test data (presented in Section A.4.14).
- The 'Go back' button returns the user to 'Simulate types' page.

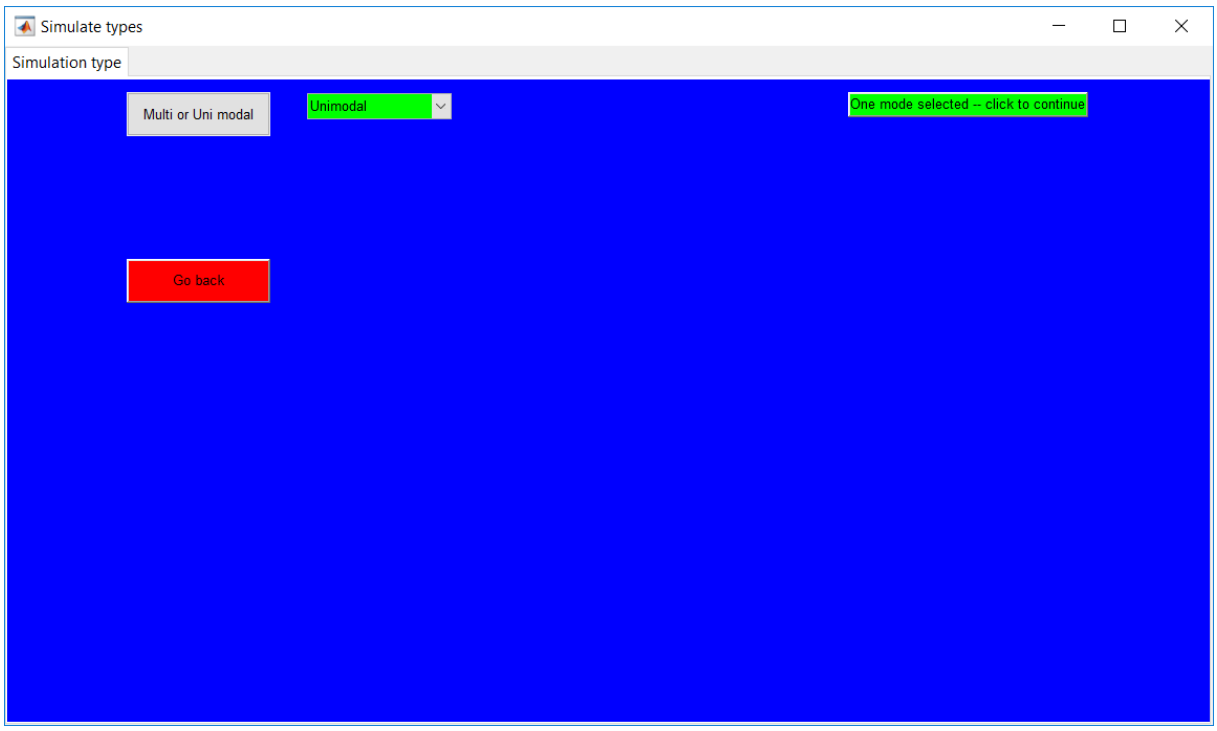


Figure A.11: Unimodal simulation selected page.

A.4.7 Multimodal

Selecting the multimodal option opens a pop-up menu that provides an option to select change in modes as concentration (C) controller or Temperature (C) controller set point change as shown in Figure A.12. A 'variable to change' message box also provides information about the options in the drop-down list of the pop-up menu.

- Selecting either option opens another pop-up menu that requires the number of modes to simulate (presented in Section A.4.8).
- As with the unimodal simulation, the 'Go back' button returns to the 'Simulation types' page.

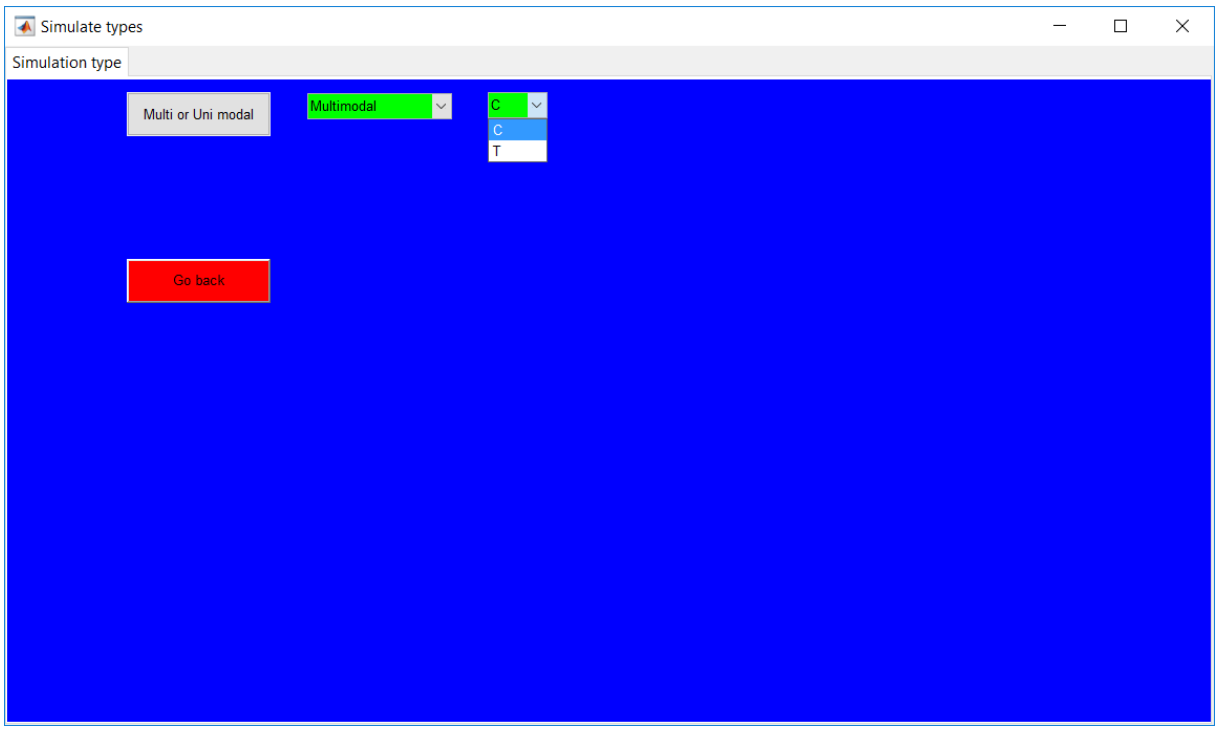


Figure A.12: Options of variables to change the controller set point.

A.4.8 Number of modes

A maximum of four modes can be simulated as shown in Figure A.13. Selecting any number of operating modes opens up the option to specify the set points of the modes (presented in Section A.4.9). A 'Number of modes' message box also provides information about the modes specification.

- The 'Go back' button returns to the 'Simulation types' page.

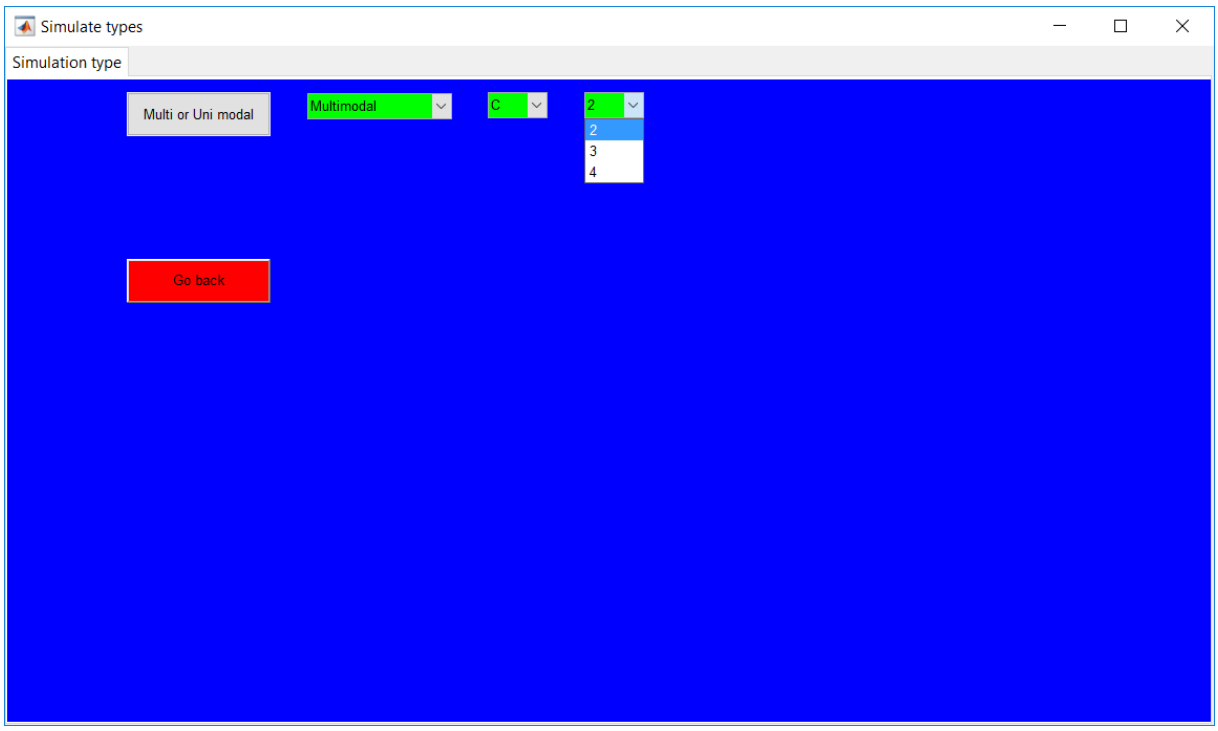


Figure A.13: Selection of number of modes to simulate.

A.4.9 Set points

The new set points of the controllers are required to be specified in percentages of the first set point value as shown in Figure A.14. To provide an idea of the current set point, an information box displays the current set point which is the steady state value of the variable. The other options with 'Enter mode set point' are to be cleared and a set point value to be entered. For example, to change the set point of the concentration controller from 0.8 to 1.2, the value of 150 is entered into the designated field. An 'Operating modes' message box also provides information about the modes specification.

- The 'Go back' button returns to the 'Simulation types' page.

Figure A.14: Option to enter new set point values.

An option to continue to the 'Simulation specifications' page for the multimode process (presented in Section A.4.11) is provided for training and validation data simulations once all fields are validated for numeric inputs as shown in Figure A.15.

Figure A.15: Option to continue to after entering new set points.

In contrast to the training and validation data, completing the required fields for the test data opens up the 'Fault specifications' page (presented in Section A.4.14).

A.4.10 Simulation specifications–Unimodal training/validation data

The 'Simulation specifications' page (in Figure A.16) for unimodal training and validation data provides options to specify the details of the simulation duration and the sampling time. The total number of data points generated is evaluated as the simulation duration / sampling time. A 'Complete required fields' message box associated with the page also provides information about the modes specification.

- Completing the required fields for the simulation and sampling times opens a button that provides a user with a 'Summary' page (presented in Section A.4.16).
- The times are specified by clearing the 'Enter value' part and inputting the appropriate numeric.
- 'Reaction upgrade/degrade' button provides options to activate/deactivate the reaction drift (presented in Section A.4.13).
- Buttons with 'Click here to turn on/off' provide options to activate/deactivate the PI controllers (presented in Section A.4.12).
- The 'Go back' button returns to the 'Simulation types' page.

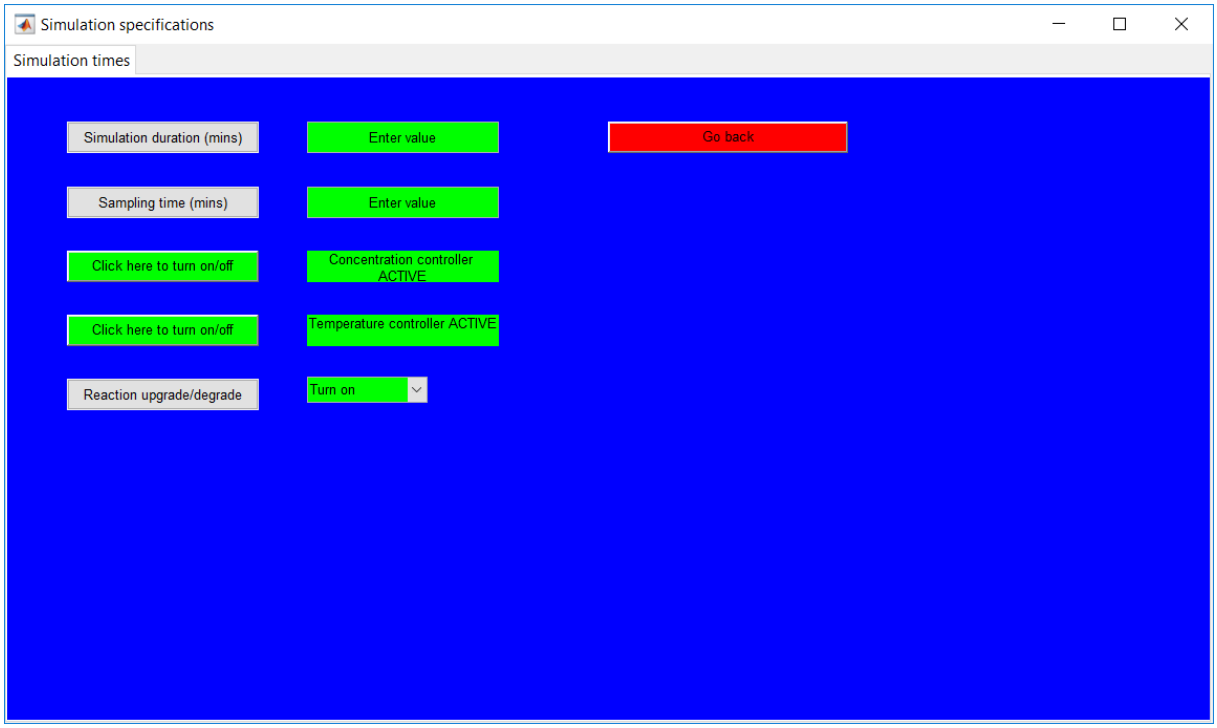


Figure A.16: Simulation specifications for time specifications – Unimodal.

A.4.11 Simulation specifications–Multimodal training/validation data

The 'Simulation specifications' page for multimodal training and validation data as shown in Figure A.17 provides the same requirements as the unimodal section with extra specifications for times to switch from one mode to another.

All mode change times and the required fields for the simulation and sampling times need to be completed before an option to the 'Summary' page (presented in Section A.4.16) is provided.

Figure A.17: Multimodal simulation specifications page for training/validation data.

A.4.12 PI controllers

All controllers can be activated/deactivated for all unimodal data simulation types as shown in Figure A.18. This is achieved by clicking on the buttons beside the respective controllers to change the current state. For multimodal simulation, on the other hand, a maximum of one controller can be deactivated as shown in Figure A.19. The controller fixed to achieve multimodal simulation cannot be deactivated. A 'Warning' message pops up when the fixed controller is clicked and remains unaffected.

Figure A.18: Unimodal simulation controller options.

Figure A.19: Multimodal controller activation/deactivation options.

A.4.13 Reaction drift

The reaction drift option, as described earlier, provides options to degrade/upgrade reaction performance. Options to turn on/off the drift is provided by selecting from the drop-down menu right to the 'Reaction upgrade/degrade' button as shown in Figure A.20.

- A positive drift is implemented by moving the slider originally at zero position to the right and the reverse for negative drift.
- The 'drift start time' and 'drift end time' provides options for times for implementation of the drift.

Figure A.20: Reaction drift selection and specification option.

Upon completion of the required fields for the drift, the current drift rate is provided in the text field below the slider as shown in Figure A.21. The drift rate is computed as magnitude/ (start time- end time).

- The magnitude is attained by moving the slider with the maximum positive and negative drifts as one.
- The 'Continue' button provides a link to the summary page (presented in Section A.4.16)

Figure A.21: Selected reaction drift rate at specified times.

A.4.14 Fault specifications

The 'Fault specifications' page collects information about the respective faults to be simulated for test data as shown in Figure A.22. The faults are activated/deactivated by clicking the respective on/off buttons.

- The sensor bias faults are to be specifed as percentages of the original sensor readings.
- As with the sensor bias faults, step input requires step values to be specified as a percentage of the initial value. A 'faults' message box also provides information about how to input respective faults.

The final step value/sensor bias values are computed as (1 + input value/100) × initial values/sensor readings.

While sensor drift and bias faults can be implemented in all variables, activating the input step in the manipulated variables pops up 'Fc warning' and 'Fa warning' message boxes respectively for the Fc and Fa step faults. This is as a result of the two controllers being active by default, and if not deactivated the respective step input faults would be overridden.

- The 'Go back' button returns to the 'Simulation types' page.
- The continue button opens up a link to the 'Simulation specifications' page for test data (presented in Section A.4.15).

Figure A.22: Fault specifications page for test data.

A.4.15 Simulation specifications–Multimodal/Unimodal test data

The 'Simulation specifications' page (shown in Figure A.23) for test data has the same features like that for the training/validation data (presented in Section A.4.10) with extra options to specify the time details for the selected faults.

- The selected 'fault on' and 'fault off' which require the times to activate and deactivate the faults. Completing all specified fields opens up the 'Summary' page (presented in Section A.4.16).

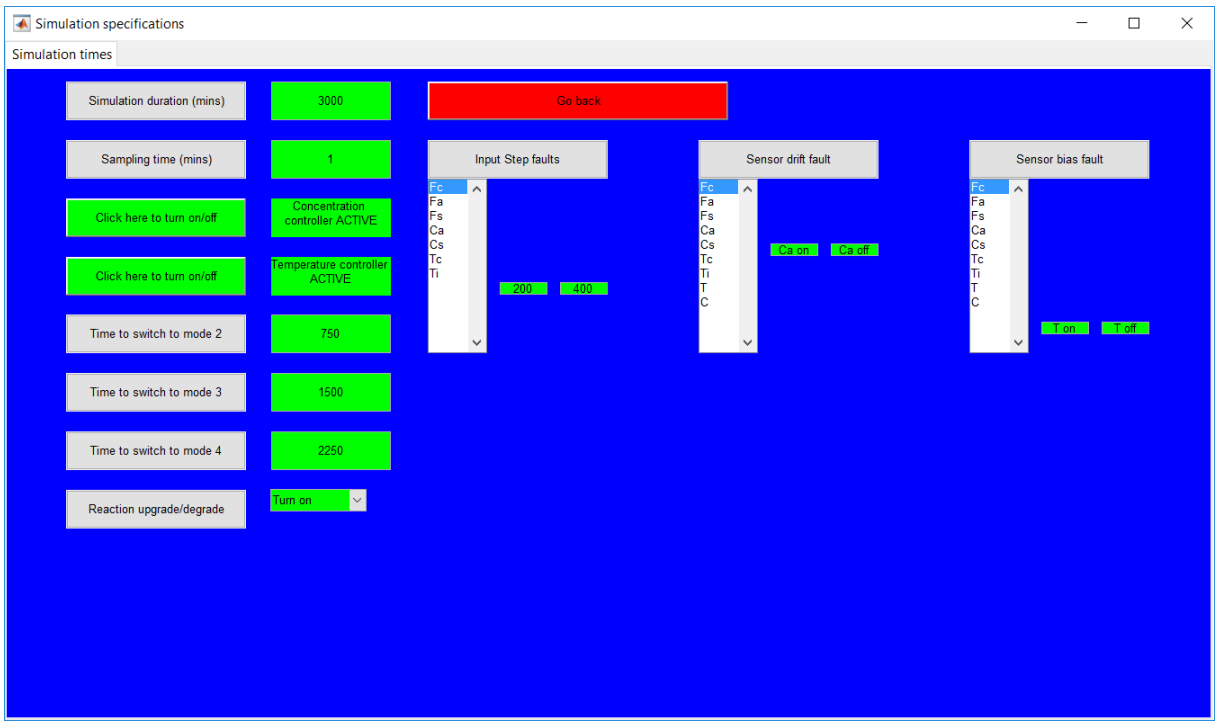


Figure A.23: Simulation specifications for test data.

A.4.16 Summary

The 'Summary' page provides a summary of all the information given by the user as shown in Figure A.24. Test data simulation provides additional information about the selected faults to be implemented at specific times and magnitudes.

- The 'Run' button simulates the non-isothermal CSTR model using the information displayed on the page and pops-up the 'Results' page (presented in Section A.4.17). A 'Simulating' message box that displays information about the current run is associated with it.

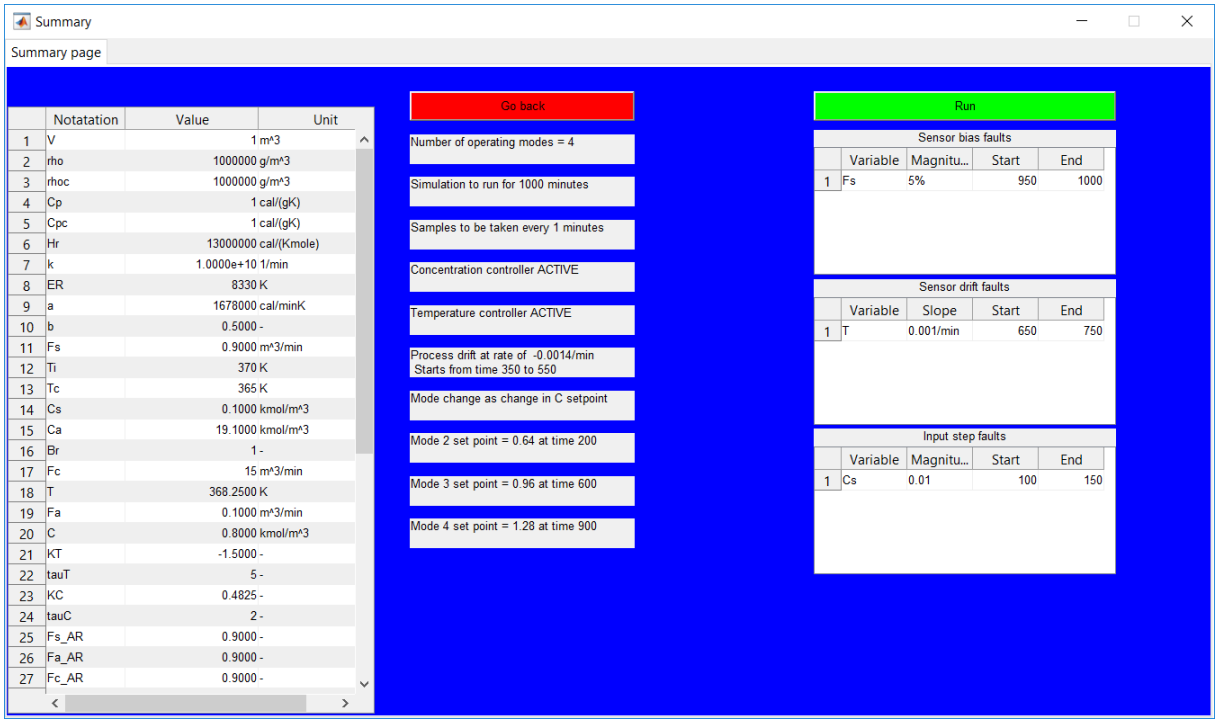


Figure A.24: Summary page for specified data.

A.4.17 Results

The ‘Results’ page as shown in Figure A.25 provides a preview of the generated simulation results for all nine process variables.

- The ‘Save and exit’ button pops up a ‘Simulation complete’ message box that displays information about the directory to which the data was saved (presented in Section A.4.18) and returns the user to the ‘Home’ page.
- The ‘Go back’ button returns the user to the ‘Summary page’ without saving the simulated data.

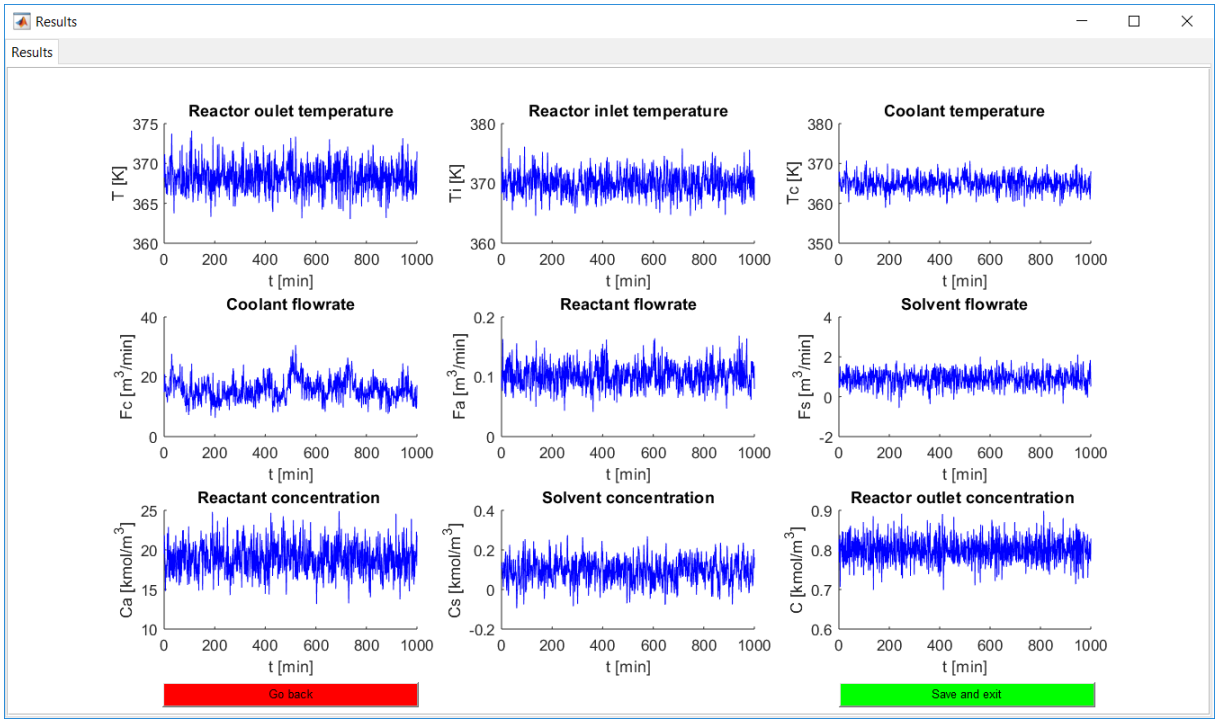


Figure A.25: Results preview of simulated data.

A.4.18 Saved data

The saved data as shown in Figure A.26 consists of the plots of the process variables and the generated simulation data as well as the simulation description.

- The graphs as shown, are plotted as subplots for the concentrations, temperatures, and flowrates respectively.
- The simulated data is saved in the 'Simulation_data' file and is made up of a table of the results with column names as the variable names.
- The seeds and specified reaction parameters by the user are saved in the 'Simulation_Parameters_and_seeds' file.
- The information captured on the 'Summary' page before the simulation is captured in the 'Summary' text file.
- The 'Faults' text file is generated as an additional file for the test data simulation. It contains information about the faults and it is tab delimited.

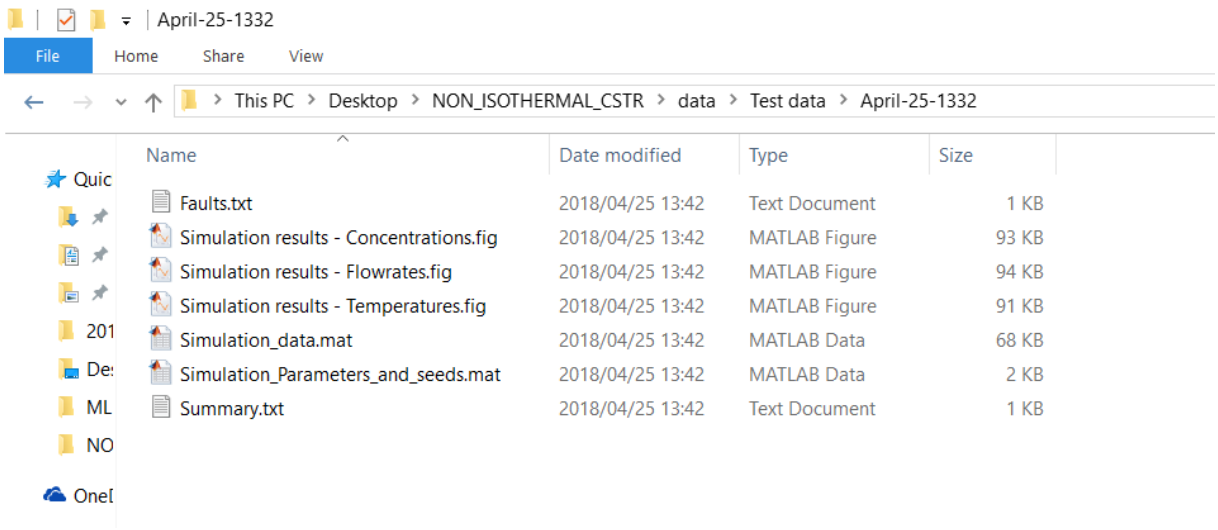


Figure A.26: Information about the saved simulation data.

APPENDIX B: COMPARISON OF SOME UNIQUE TE FAULTS

Appendix B provides information on the fault detection performance of some selected Tennessee Eastman process faults. APCA is compared with conventional PCA.

The performance of APCA and conventional PCA are analysed using specific TE faults (1and 21). The above-listed faults are selected due to the unique signature of the faults; fault 1 being an easily detectable (extreme) fault; and fault 21, which exhibits drift property.

B.1 Fault detection for Fault T01

The T^2 statistic for the APCA (shown in Figure B.1) and conventional PCA (shown in Figure B.2) achieves similar results for the alarm rates with a slight deviation in detection delay for which APCA performs better. Due to the extreme nature of the step fault, a zoomed-in image is provided in Figure B.3 which shows the model adaptation in APCA.

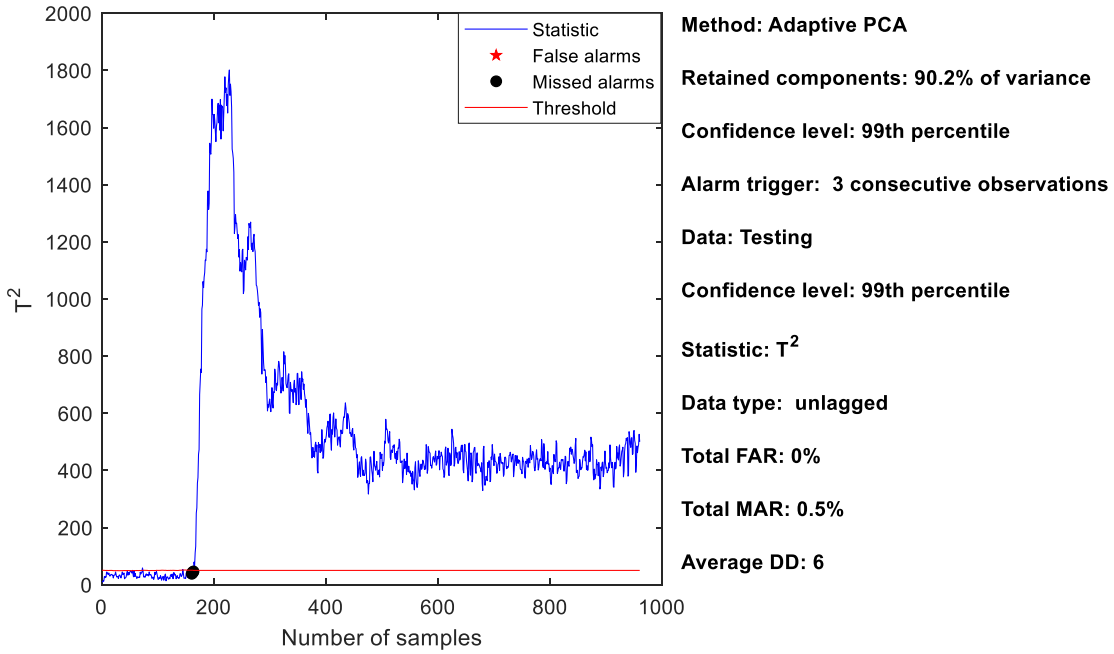


Figure B.1: APCA T^2 statistic for Fault T01.

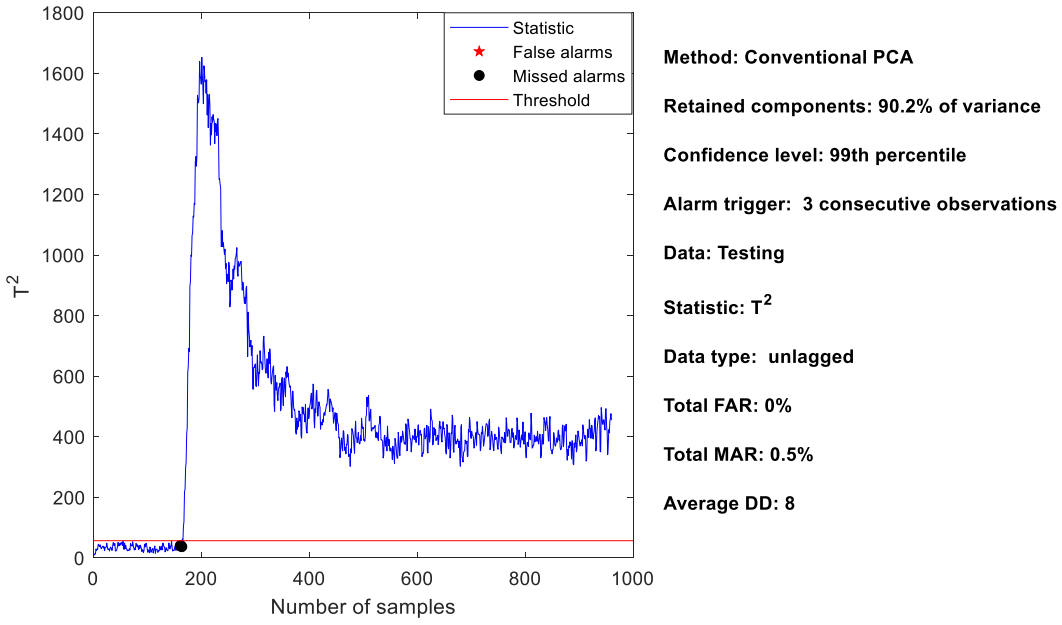


Figure B.2: Conventional PCA T^2 statistic for Fault T01.

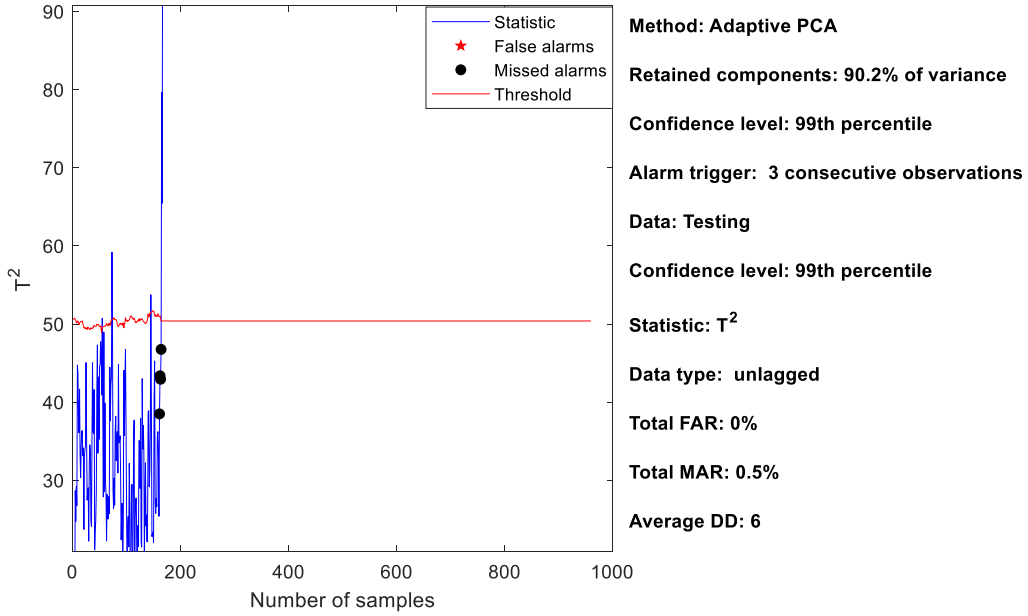


Figure B.3: APCA T^2 statistic for Fault T01 (zoomed-in).

A plot of the AUCs for conventional PCA and APCA (in Figure B.4) shows perfect AUCs for the T^2 statistic in both methods. The SPE statistic, however, shows a better AUC for APCA. Sample FAR and MAR evaluated at specified z values (shown in Figure B.4) shows similar results for both methods with APCA performing better in situations where there are slight differences.

Fault T01 is equally detectable by the two methods with no significant difference as shown in the results presented, a much more complicated situation (Fault T21) is considered in the next section.

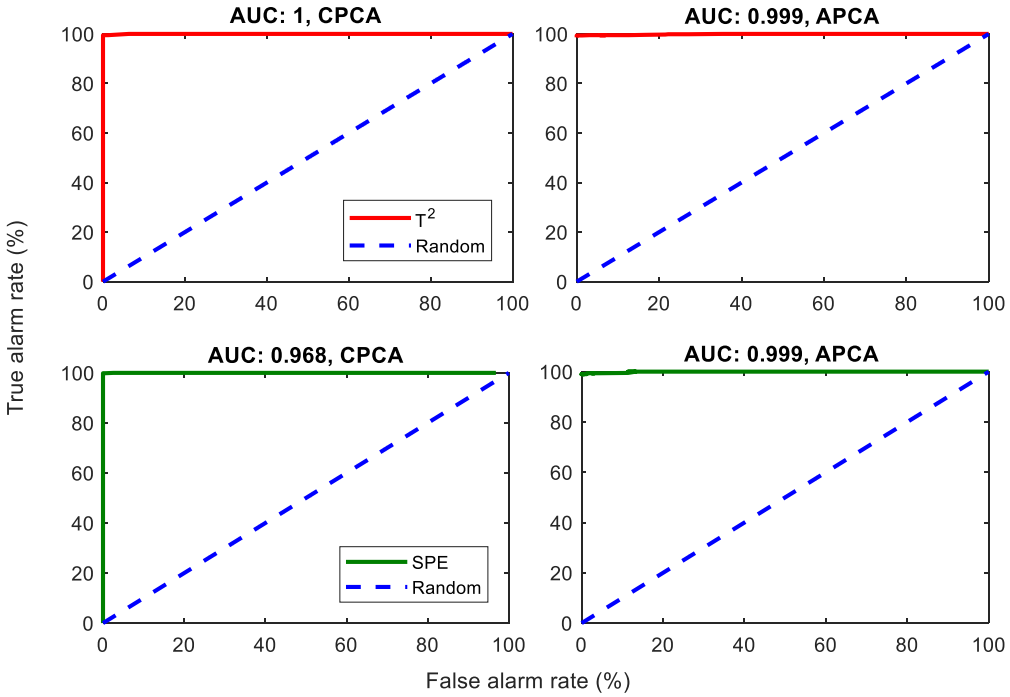


Figure B.4: Monitoring statistics ROC curve for Fault T01 evaluated at $z = 3$ and 90.2% retained variance.

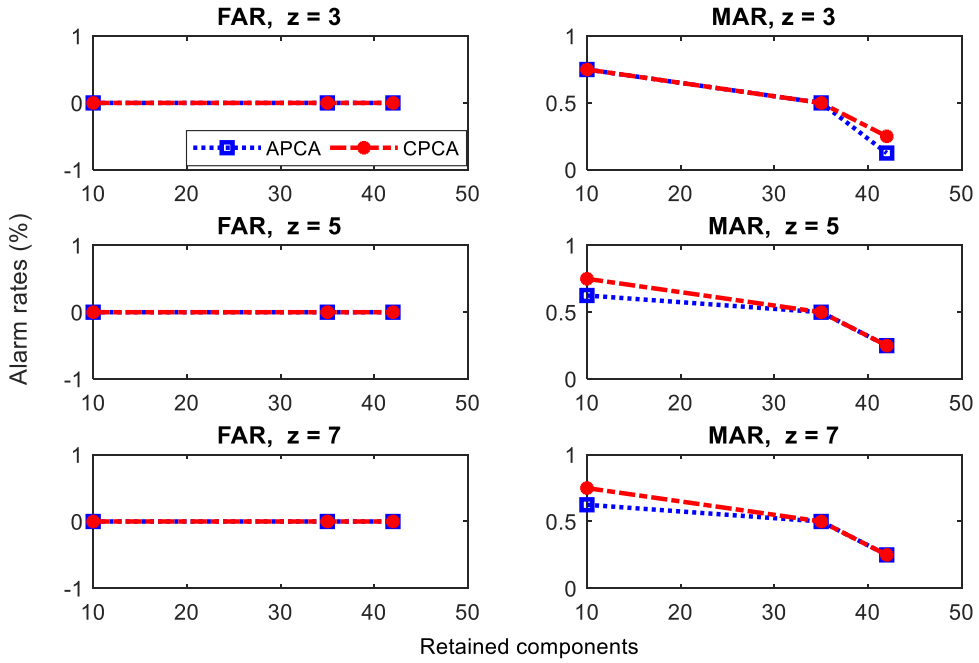


Figure B.5: T^2 for Fault T01 evaluated at specified z values, 90.2% retained variance and 99.5th percentile.

B.2 Fault detection for Fault T21

Fault 21 of the TE process is described as a 'constant position' fault type which was achieved by keeping a valve stream at the steady state position. Although not much of the faults simulation can be described, the signature of the fault (in Figure B.6) shows a drifting process. Initial stages of the fault show a constant process that drifts somewhere after sample 400.

Consequently, results of the APCA (in Figure B.7) shows adaptation to the process changes without detecting the 'fault'. To the question of whether the drift is an acceptable change or fault is open to interpretation, as this will depend on process knowledge.

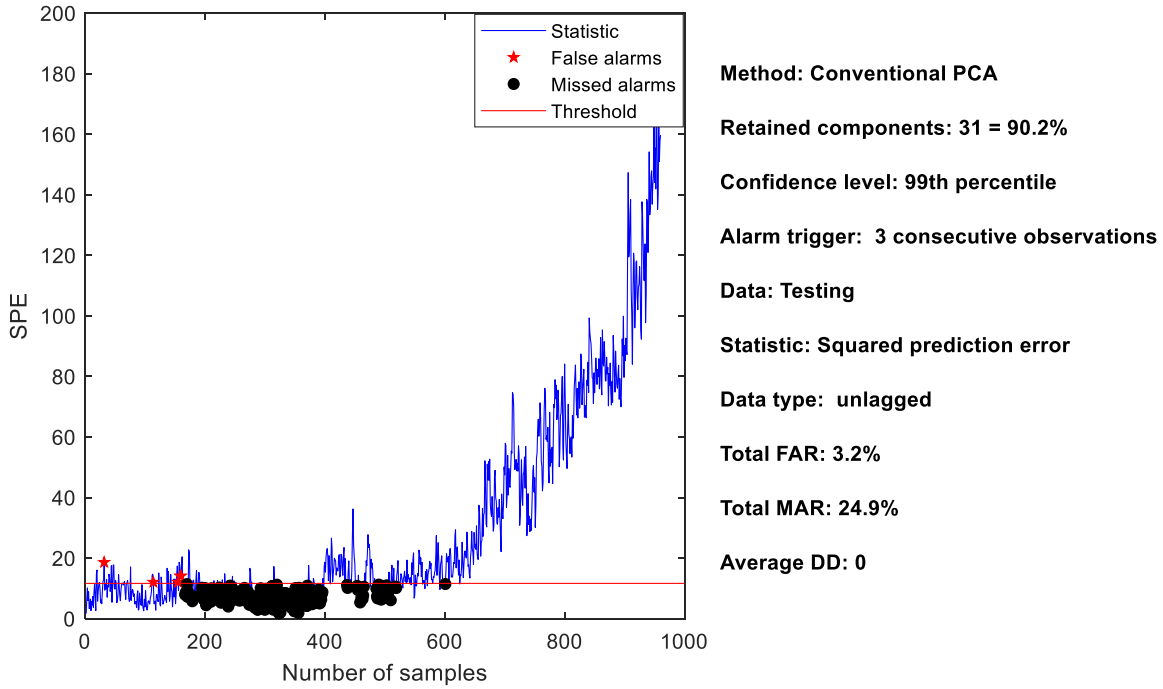


Figure B.6: Conventional PCA SPE statistic for Fault T21 evaluated at $z = 3$ and 90.2% retained variance.

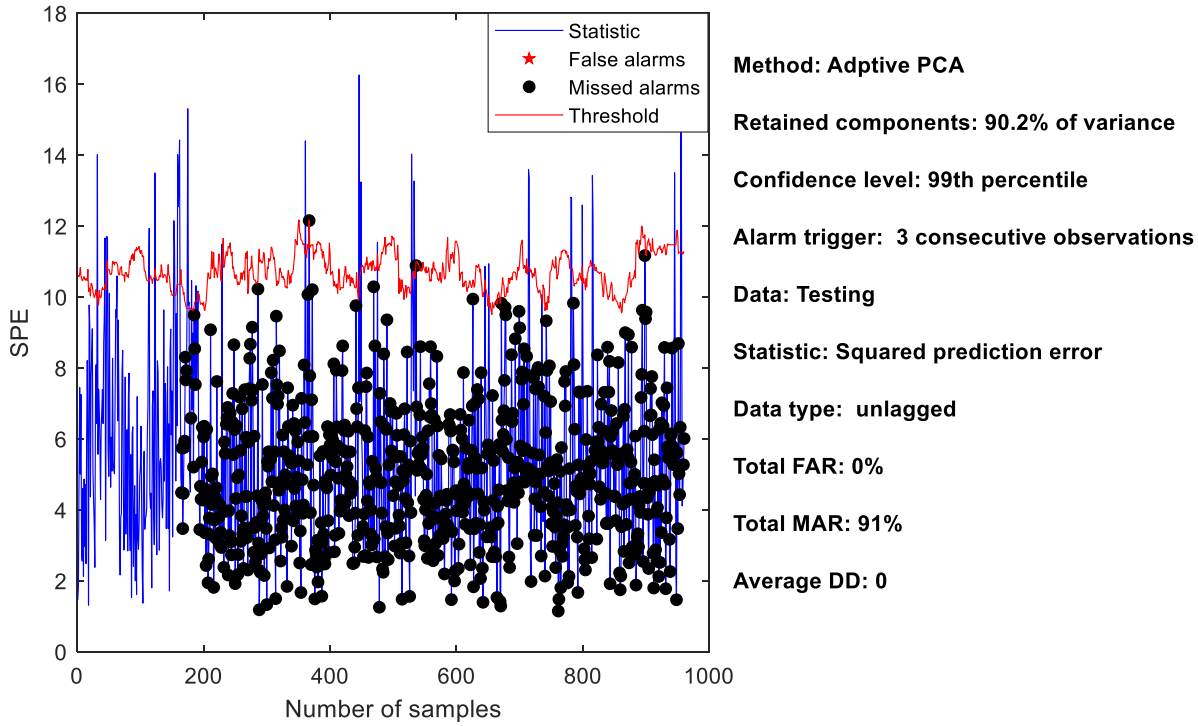


Figure B.7: APCA SPE statistic for Fault T21 evaluated at $z = 3$ and 90.2% retained variance.

APPENDIX C: COMPARISON OF DEVELOPED PCA AND APCA

Appendix C provides information on the fault detection performance for Tennessee Eastman process faults. Section C.1 compares the developed PCA (conventional) results with that achieved by Russell et al. (2000). Section C.2 provides the results for the performance of APCA compared with that for conventional PCA using the same retained PCs as that implemented in Section C.1.

C.1 Comparison of conventional PCA results for TE process

The method implemented by Russell et al. (2000) involves adjusting the thresholds (critical values) for the various faults as well as different retained PCs. Although there is information on the retained PCs, that on the implemented thresholds are somehow unclear and this is supported by no attempt by other works to reproduce the results after an exhaustive search. According to Russell et al. (2000), "In computing, the missed detection rates for Faults 1–21 of the testing set, the threshold for each statistic was adjusted to the tenth highest value for the normal operating condition of the *testing* set." The thresholds are further adjusted for faults with high FAR.

The closest approximation (CA) of the results based on the provided information (Russell, Chiang, and Braatz, 2000) involves the following steps:

1. Normalize the overall training data. Retain means and variances.
2. Compute the retained PCs and eigenvalues of the normalized data.
3. For a specific fault (for example Fault T01), normalize the NOC (training) data for the specific fault (which can also be termed a validation data for the overall training data). Compute the T^2 and SPE statistics using the retained PCs and eigenvalues. Note that for every fault dataset there is a corresponding NOC data as explained earlier.
4. Sort the respective statistic values and use the 10^{th} highest value as the critical value.
5. Compute the SPE and T^2 statistics of the test (fault) data using the retained means, variances, retained PCs and eigenvalues of the overall training data.

The best approximation (BA) of the results, however, involves computing the critical values by taking the 99.5^{th} percentile of the first 160 observations of a test (fault) data which is also NOC data.

Figure C.1 and Figure C.2 respectively show the results achieved for the SPE statistic of Fault T04 for the best approximation (BA) and the closest approximation (CA). Also, the T^2 statistic results for Fault T04 is presented in Figure C.3.

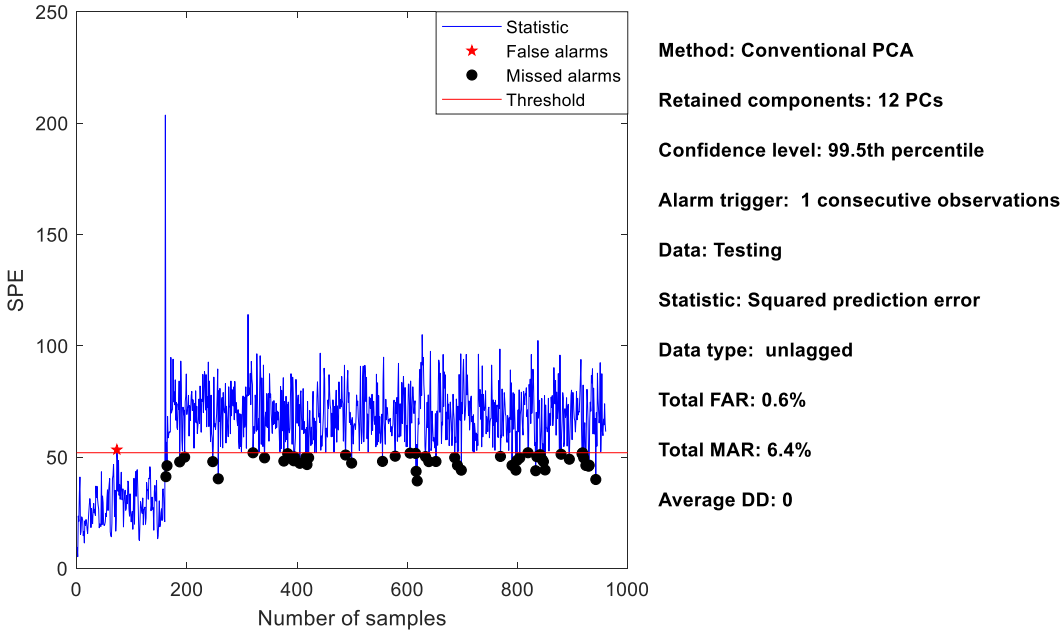


Figure C.1: SPE statistic for Fault T04 for BA. See Russell et al. (2000) for comparison.

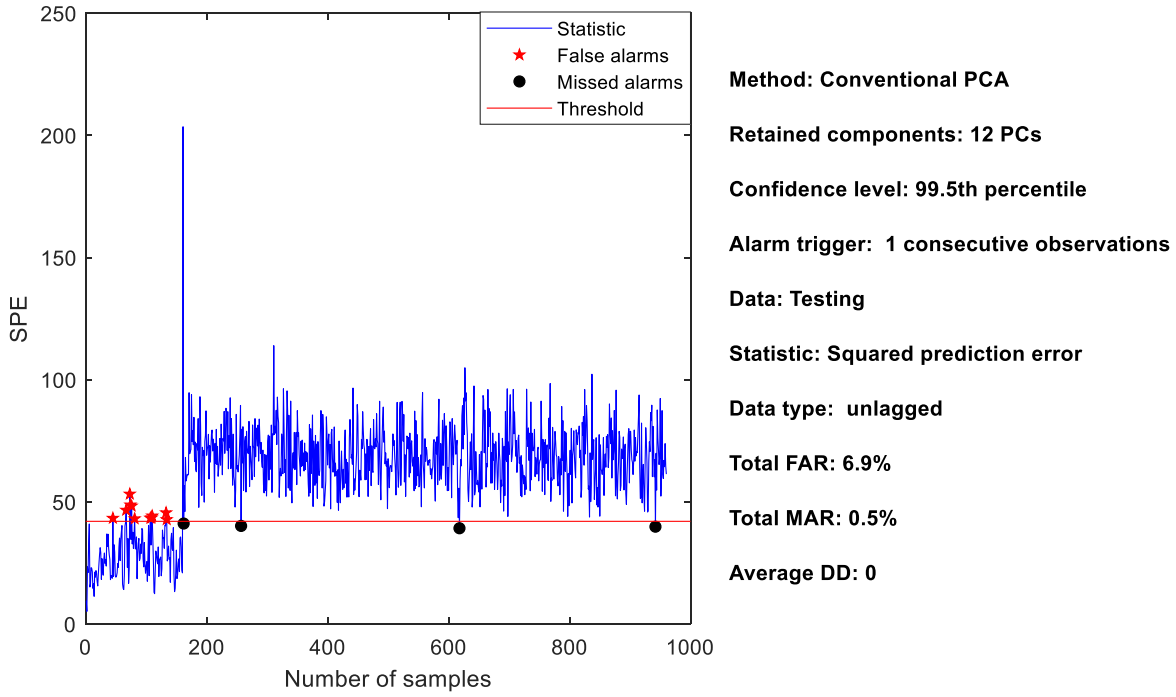


Figure C.2: SPE statistic for Fault T04 for CA. See Russell et al. (2000) for comparison

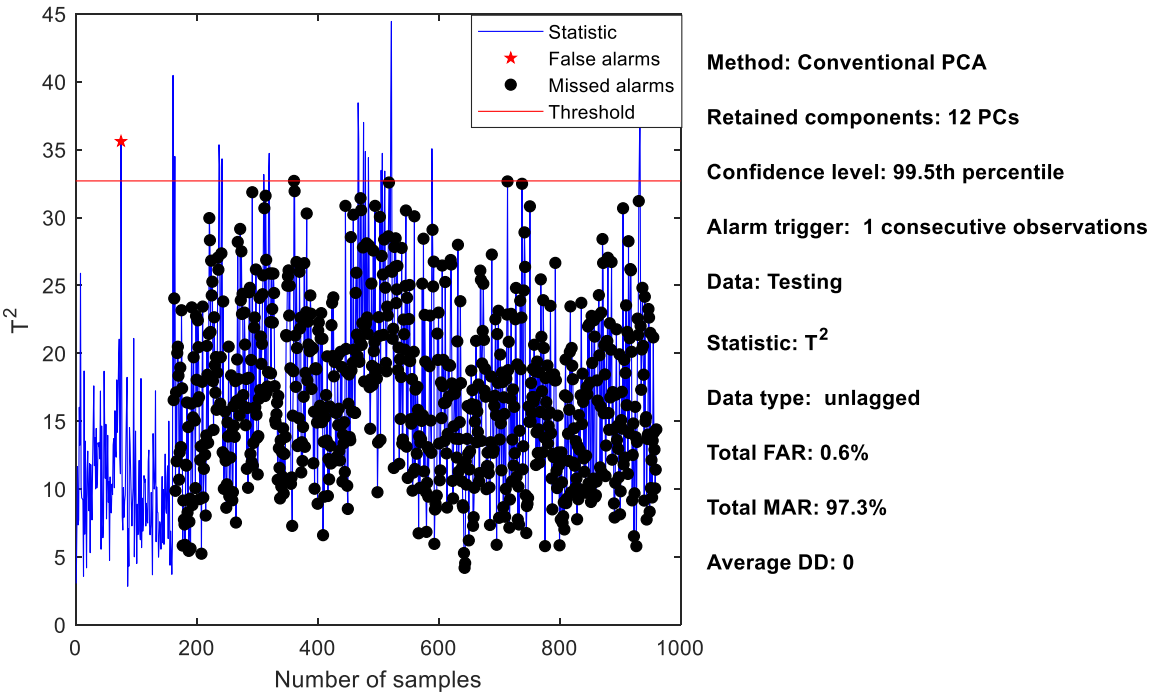


Figure C.3: T^2 statistic for Fault T04 for BA. See Russell et al. (2000) for comparison

Also, sample results for the T^2 statistic for Fault T05 for BA is provided in Figure C.4.

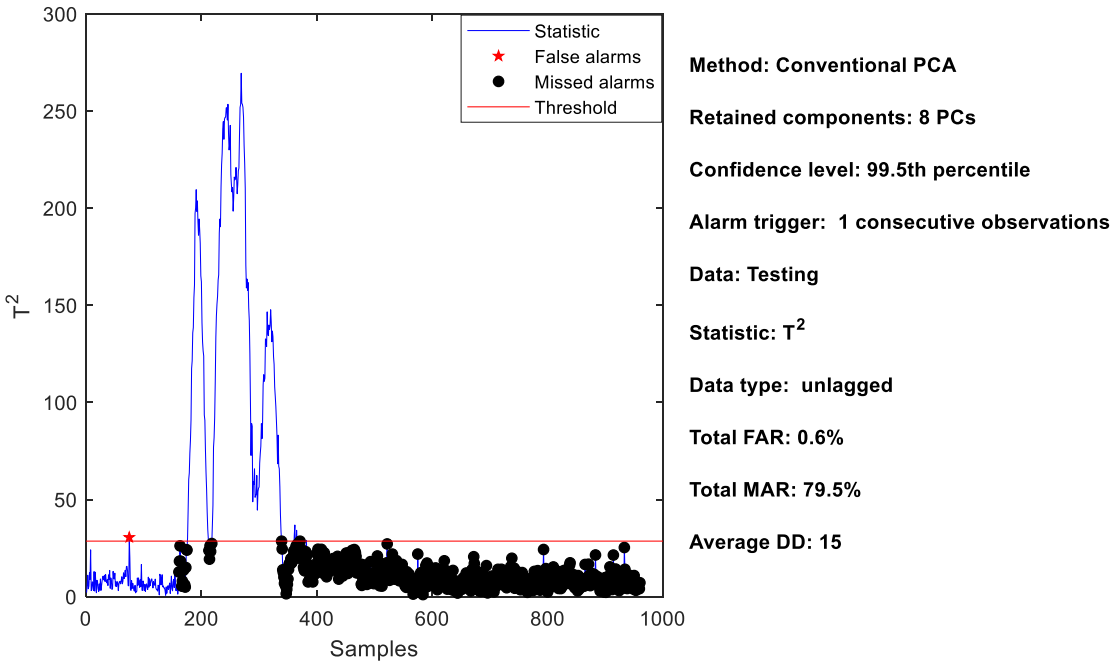


Figure C.4: T^2 statistic for Fault T05 for BA. See Russell et al. (2000) for comparison

The results for the BA without further adjustments produce closer results to that reported for Russell et al. (2000) as summarized in Table C.1 for the missed alarm rates (with $z = 1$).

Table C.1: Comparison of MAR for TE results of Russell et al. (2000) (RB) with that for the close approximation (CA) and best approximation (BA).

	Missed alarm rate					
Fault	**T^2 (RB)**	**T^2 (CA)**	**T^2 (BA)**	**SPE (RB)**	**SPE (CA)**	**SPE (BA)**
1	0.008	0.009	0.008	0.003	0.003	0.003
2	0.020	0.020	0.020	0.014	0.018	0.014
3	0.998	0.828	0.999	0.991	0.870	0.993
4	0.956	0.726	0.973	0.038	0.005	0.064
5	0.775	0.790	0.795	0.746	0.716	0.755
6	0.011	0.013	0.008	0.000	0.005	0.000
7	0.085	0.611	0.494	0.000	0.000	0.000
8	0.034	0.030	0.030	0.024	0.020	0.024
9	0.994	0.856	0.998	0.981	0.913	0.994
10	0.666	0.455	0.616	0.659	0.474	0.579
11	0.794	0.550	0.734	0.356	0.210	0.375
12	0.029	0.036	0.036	0.025	0.006	0.019
13	0.060	0.059	0.059	0.045	0.045	0.045
14	0.158	0.008	0.015	0.000	0.000	0.000
15	0.988	0.840	0.883	0.973	0.901	0.960
16	0.834	0.556	0.898	0.755	0.529	0.738
17	0.259	0.185	0.216	0.108	0.064	0.099
18	0.113	0.121	0.116	0.101	0.108	0.104
19	0.996	0.855	0.963	0.873	0.795	0.814
20	0.701	0.491	0.519	0.550	0.424	0.478
21	0.736	0.576	0.654	0.570	0.576	0.594

The results of the detection delays for the various aforementioned implementations are presented in Table C.2 (with $z = 6$ and undetected faults having infinity (Inf) value).

Table C.2: Comparison of DD for TE results of Russell et al. (2000) with that for the close approximation (CA) and best approximation (BA).

	Detection delay (samples)					
Fault	**T^2 (RB)**	**T^2 (CA)**	**T^2 (BA)**	**SPE (RB)**	**SPE (CA)**	**SPE (BA)**
1	7	12	11	3	7	7

Detection delay (samples)						
Fault	**T^2 (RB)**	**T^2 (CA)**	**T^2 (BA)**	**SPE (RB)**	**SPE (CA)**	**SPE (BA)**
2	17	21	21	12	19	16
3	Inf	45	Inf	Inf	Inf	Inf
4	Inf	64	Inf	3	7	9
5	16	20	20	1	5	5
6	10	15	11	1	9	5
7	1	5	5	1	5	5
8	23	30	30	20	22	24
9	Inf	7	Inf	Inf	Inf	Inf
10	96	20	75	49	52	53
11	304	192	198	11	10	15
12	22	27	27	8	11	11
13	49	53	53	37	41	41
14	4	8	8	1	5	5
15	Inf	585	670	740	724	744
16	312	40	317	197	23	200
17	29	32	32	25	29	29
18	93	102	100	84	91	91
19	Inf	Inf	Inf	Inf	Inf	Inf
20	87	89	89	87	91	91
21	563	467	561	285	289	289

The normalization of the NOC data of the specific faults (as implemented for the CA), however, involves doing that with the means and variances of the specific data and not that of the overall training data. The results obtained for the normalization using means and variances retained from the overall training data does not perform well and is reported as the closest approximation one (CA-1). Sample of such results is presented in Figure C.5 and Figure C.6.

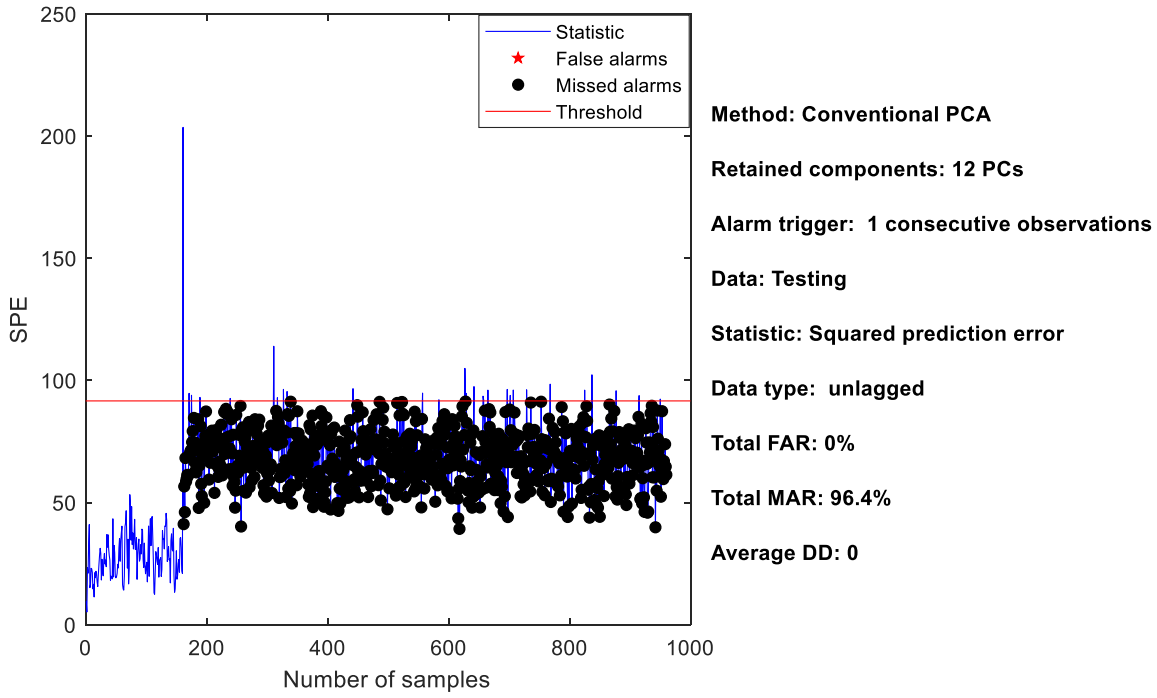


Figure C.5: SPE statistic for Fault T04 for CA-1.

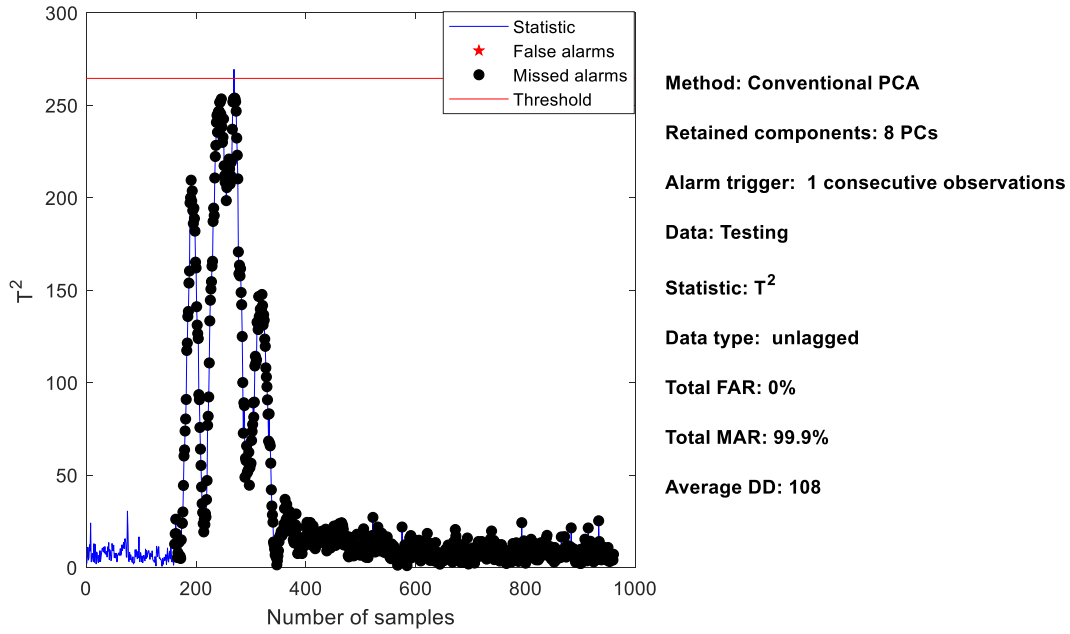


Figure C.6: T^2 statistic for Fault T05 for CA-1.

C.2 Comparison of conventional PCA and APCA results for TE process

Comparison of results for conventional PCA and APCA makes use of the traditional approach of computing threshold values. This involves evaluation at a specific confidence level of the

training data (overall). The results for the APCA and conventional PCA are summarized in Table C.3.

Table C.3: Comparison of alarm rates for APCA (A) and conventional PCA (C) evaluated at the 99th percentile and z =1.

	Missed alarm rate				False alarm rate			
Fault	**A(T^2)**	**C(T^2)**	**A-SPE**	**C-SPE**	**A(T^2)**	**C(T^2)**	**A-SPE**	**C-SPE**
1	0.008	0.008	0.001	0.001	0.000	0.000	0.056	0.056
2	0.018	0.018	0.013	0.013	0.013	0.013	0.038	0.038
3	0.969	0.979	0.929	0.904	0.031	0.013	0.075	0.081
4	0.859	0.920	0.004	0.001	0.006	0.006	0.081	0.113
5	0.745	0.744	0.688	0.685	0.013	0.013	0.031	0.031
6	0.008	0.008	0.000	0.000	0.006	0.006	0.013	0.013
7	0.519	0.518	0.000	0.000	0.000	0.000	0.013	0.013
8	0.035	0.035	0.021	0.021	0.000	0.000	0.050	0.050
9	0.973	0.971	0.919	0.919	0.050	0.056	0.056	0.056
10	0.605	0.605	0.450	0.449	0.006	0.006	0.044	0.044
11	0.681	0.706	0.271	0.255	0.006	0.006	0.031	0.044
12	0.035	0.035	0.011	0.011	0.013	0.013	0.025	0.031
13	0.060	0.060	0.044	0.044	0.000	0.000	0.019	0.019
14	0.024	0.023	0.000	0.000	0.000	0.000	0.069	0.069
15	0.950	0.950	0.880	0.880	0.000	0.000	0.044	0.044
16	0.763	0.774	0.523	0.499	0.144	0.100	0.044	0.063
17	0.224	0.244	0.069	0.064	0.000	0.000	0.013	0.013
18	0.118	0.118	0.099	0.099	0.000	0.000	0.031	0.031
19	0.974	0.981	0.646	0.585	0.000	0.000	0.088	0.088
20	0.643	0.641	0.403	0.403	0.000	0.000	0.031	0.031
21	0.678	0.678	0.444	0.440	0.000	0.000	0.150	0.150

The detection delay results for the alarm rates are summarized in Table C.4 and Table C.5 for z = 6 and z = 1 respectively.

Table C.4: Comparison DD for APCA (A) and conventional PCA (C) results implemented as well as that for Russell et al. (2000) (RB) and at the 99th percentile and z = 6.

Detection delay (samples)						
Fault	**C(T^2) (RB)**	**C-SPE (RB)**	**C(T^2)**	**A(T^2)**	**C-SPE**	**A-SPE**
1	7	3	11	11	6	6
2	17	12	19	19	15	15
3	Inf	Inf	Inf	Inf	Inf	Inf
4	Inf	3	Inf	361	5	5
5	16	1	18	18	5	5
6	10	1	11	11	5	5
7	1	1	5	5	5	5
8	23	20	30	30	24	24
9	Inf	Inf	Inf	Inf	Inf	Inf
10	96	49	75	75	52	52
11	304	11	197	197	14	15
12	22	8	27	27	11	11
13	49	37	53	53	40	40
14	4	1	8	8	5	5
15	Inf	740	679	679	724	724
16	312	197	312	312	23	23
17	29	25	32	32	29	29
18	93	84	100	100	89	89
19	Inf	Inf	Inf	Inf	135	135
20	87	87	90	90	86	86
21	563	285	561	561	260	262

Table C.5: Comparison DD for APCA (A) and conventional PCA (C) results at 99th percentile and z = 1.

Fault	**C(T^2)**	**A(T^2)**	**C-SPE**	**A-SPE**
1	6	6	1	1
2	14	14	10	10
3	40	40	20	20
4	0	0	0	0
5	2	2	0	0
6	6	6	0	0
7	0	0	0	0

Fault	C(T^2)	A(T^2)	C-SPE	A-SPE
8	9	9	15	15
9	2	2	0	0
10	22	22	24	24
11	6	6	5	5
12	6	6	2	2
13	48	48	35	35
14	1	1	0	0
15	572	572	90	90
16	33	33	3	3
17	27	27	24	24
18	5	5	14	14
19	10	10	9	9
20	78	78	81	81
21	256	256	1	1

APPENDIX D: PROCESS DRIFT CASE—CSTR PROCESS

Appendix D provides information on the simulated data from the CSTR process and also the retained model parameters.

D.1 Simulated data

To achieve a process drift, a process reaction change β_r with the value of -0.001 min^{-1} is introduced into the reaction rate with the resultant reaction rate in Equation **D-1**. This is to demonstrate slow reaction performance changes that could be attributed to catalyst deactivation or heat exchanger fouling, among other factors.

$$r = \beta_r k_o e^{-E/RT} C \qquad \textbf{D-1}$$

Test data C01 is simulated by introducing a step fault of magnitude 3% in the reactor inlet temperature Ti from time 702 to 1001. The test data C02 is simulated in the same way as C01 but with the introduction of the process drift from time 202 to 702 after which the step fault was introduced from time 702 to 1001. The training data and validation data for the analysis are Ct00 and Cv00 respectively. All simulations have different random seeds. Figure D.1 shows the simulation results for C02 which shows a process drift and step change with a response from manipulated variables Fc and Fa. Table D.1 shows a summary of the simulated data.

Table D.1: Summary of simulated CSTR data.

Data	Description	Samples
Ct00	Training data	1001
Cv00	Validation data for Ct00	501
C01	Step in Ti	1001
C02	Process drift and step in Ti	1001

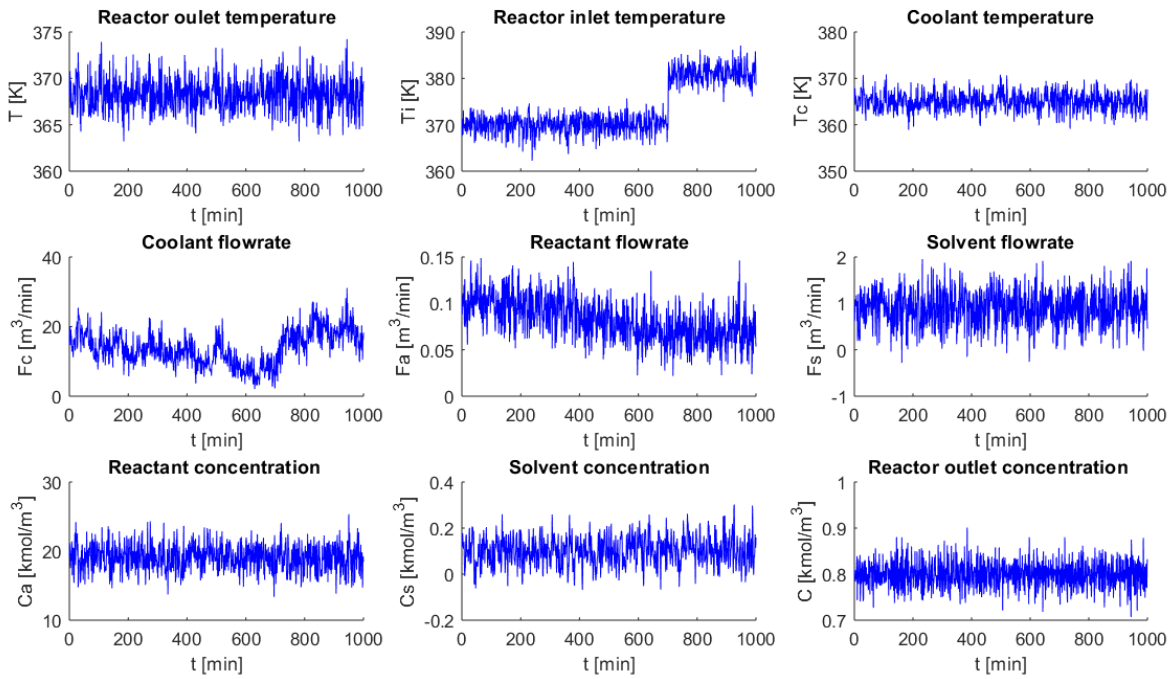


Figure D.1: Simulation results for process drift and step in Ti with a response from manipulated variables Fc and Fa.

D.2 Retained PCs

The scree plots and Pareto charts (in Figure D.2) shows that at least 4 and 5 PCs are respectively required to account for 80% and 90% variance in the NOC data.

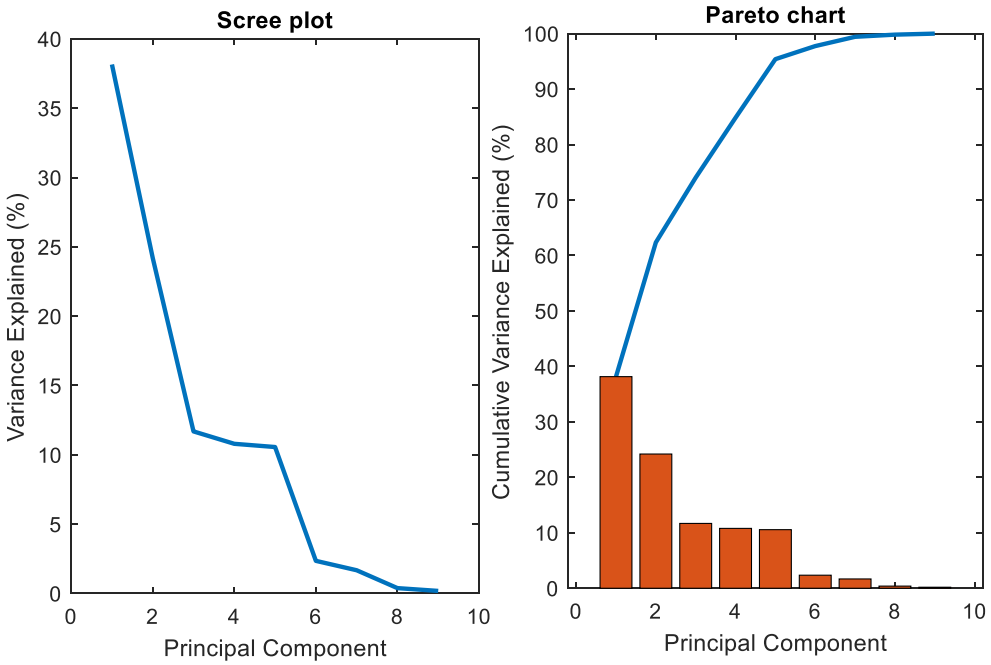


Figure D.2: Scree plot (left) and Pareto chart (right) of CSTR process training data Ct00.

APPENDIX E: COMPARISON OF REPCA AND MWPCA

Appendix E provides information on the comparison of RePCA and MWPCA by considering the effect of different hyperparameter values in fault detection.

E.1 CSTR process Fault C03

The CSTR process training data Ct00 and validation data Cv00 are used in this case. Test data C03 (shown in Figure E.1) is generated by introducing the process drift (discussed earlier) from time 202 to 702 after which a step fault of magnitude 3% is introduced in reactor inlet temperature Ti from time 702 to 1001, the process then returns to NOC from time 1001 to 1301.

Table E.1: Summary of simulated CSTR data.

Data	Description	Samples
Cs03	Process drift and step in Ti and back to NOC	1301

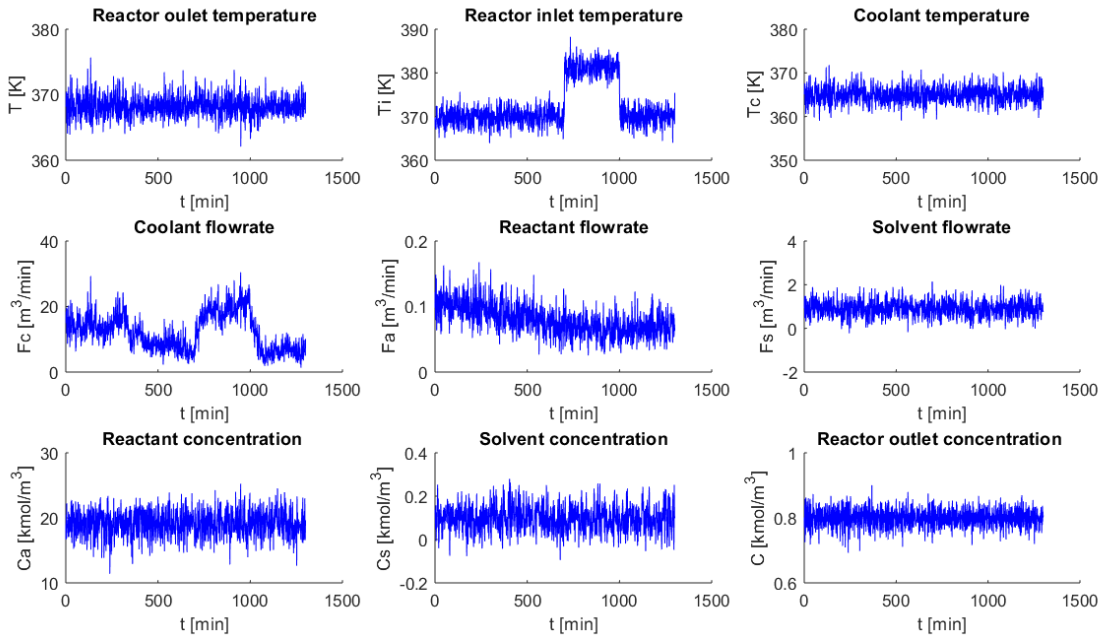


Figure E.1: Simulation results for C03 showing process drift and step in Ti with a response from manipulated variables Fc and Fa.

E.2 Fault detection for Fault C03

SPE results for MWPCA and RePCA are shown in Figure E.2 and Figure E.3 respectively.

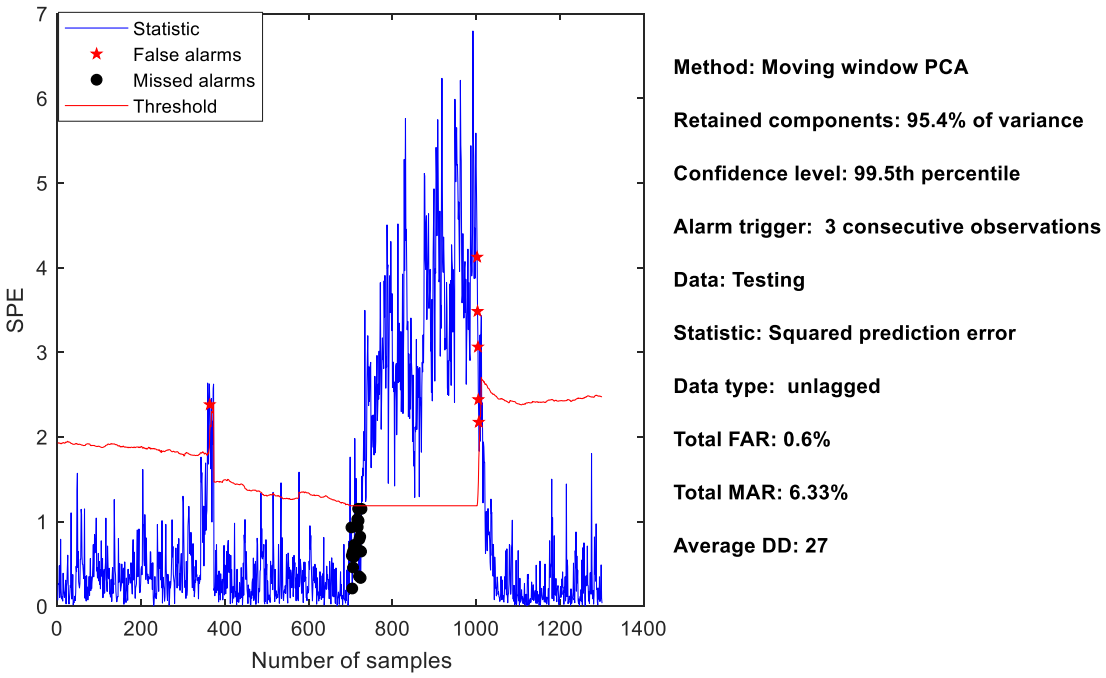


Figure E.2: MWPCA SPE statistic performance for Fault C03.

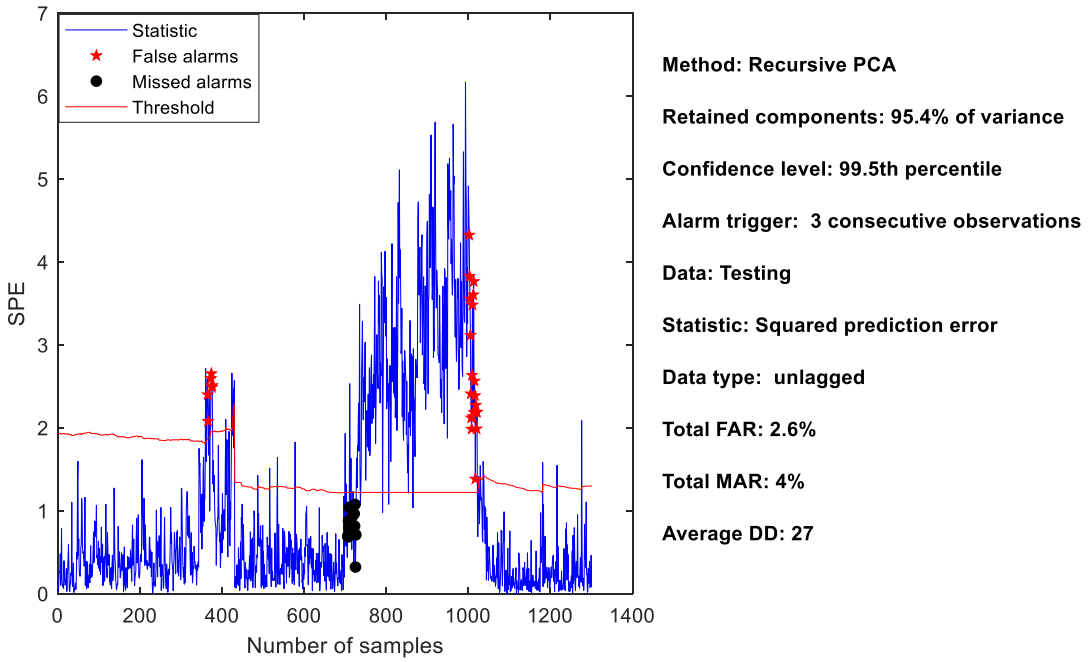


Figure E.3: RePCA SPE statistic performance for Fault C03.

Figure E.4 provides an insight into the performance which shows slightly better performance for MWPCA in the SPE detection delay statistic; the T^2 values are however equal in all regards. This shows that the SPE statistic tends to suffer more from the process drift than the T^2 in this case.

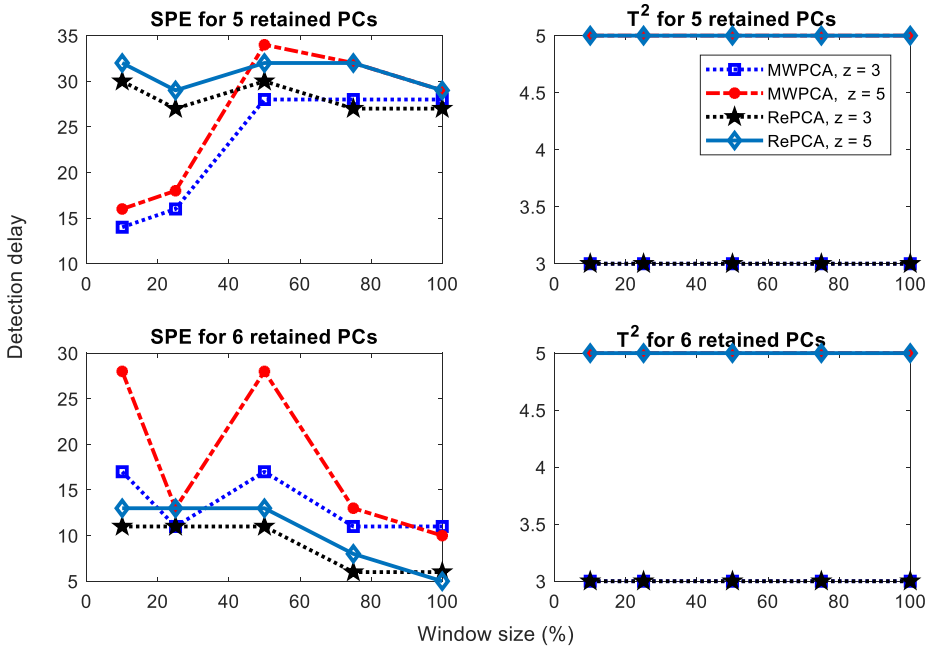


Figure E.4: Detection delays of MWPCA and RePCA for different retained PCs, window sizes, and z values at 99.5 percentile.

E.3 CSTR process Fault C04

Test data C04 (shown in Figure E.5) is generated by introducing the process drift at time 3202 to 3702 after which a step fault of magnitude 3% is introduced in the reactor inlet temperature Ti from time 4502 to 5001. This is to evaluate the impact of increased observations on the performance of the recursive and moving window MWPCA.

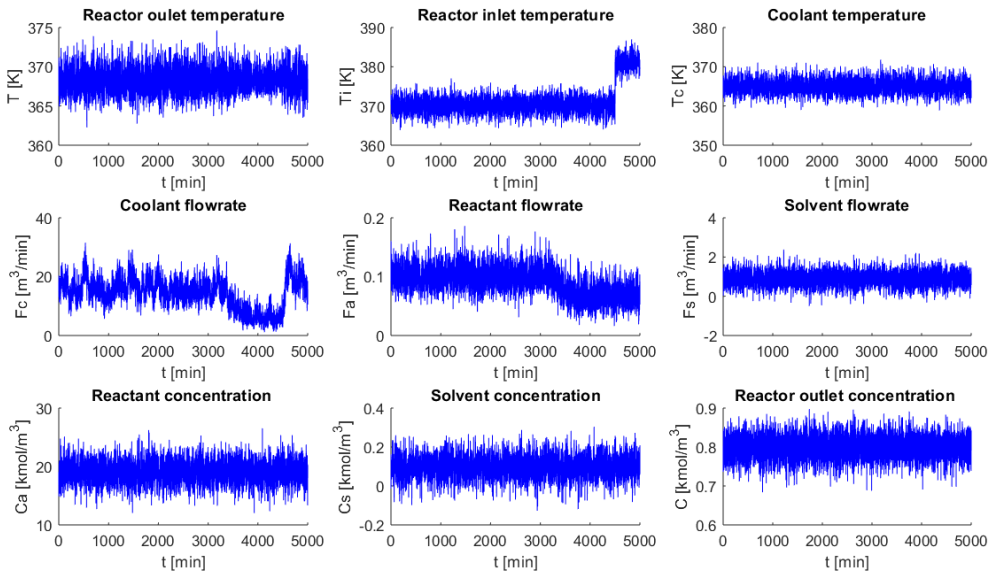


Figure E.5: Simulation results for C04 showing process drift and step in Ti with a response from manipulated variables Fc and Fa.

E.4 Fault detection for Fault C04

Sample T^2 statistic results show a better performance of MWPCA (shown in Figure E.6) than that for RePCA (shown in Figure E.7).

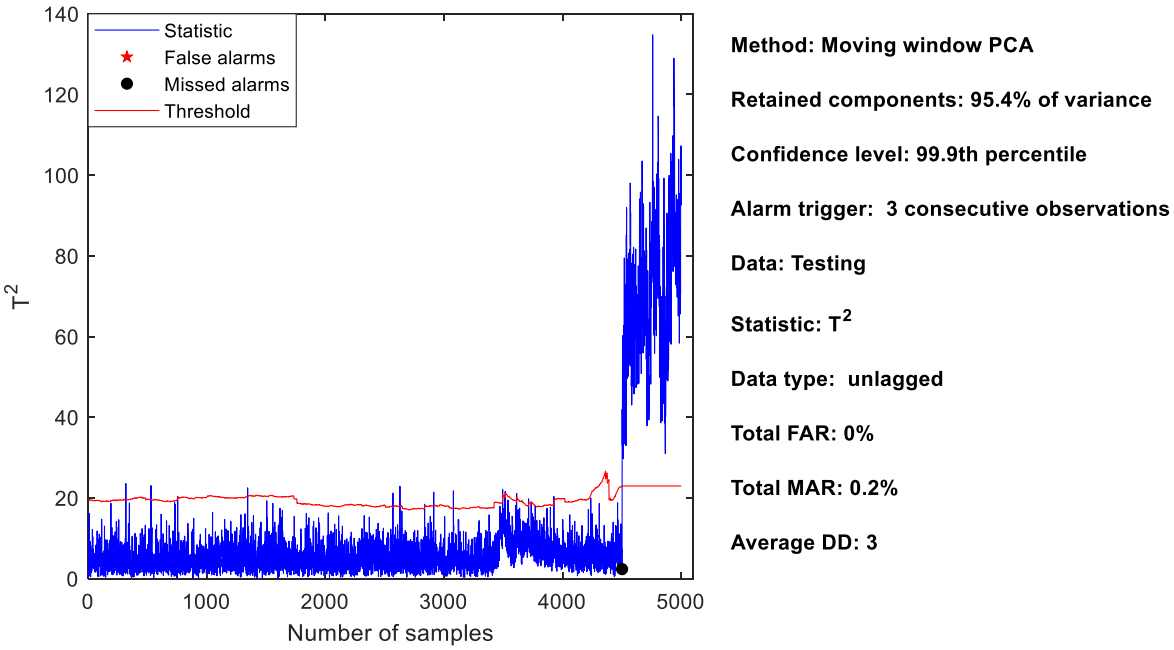


Figure E.6: MWPCA T^2 statistic for Fault C04.

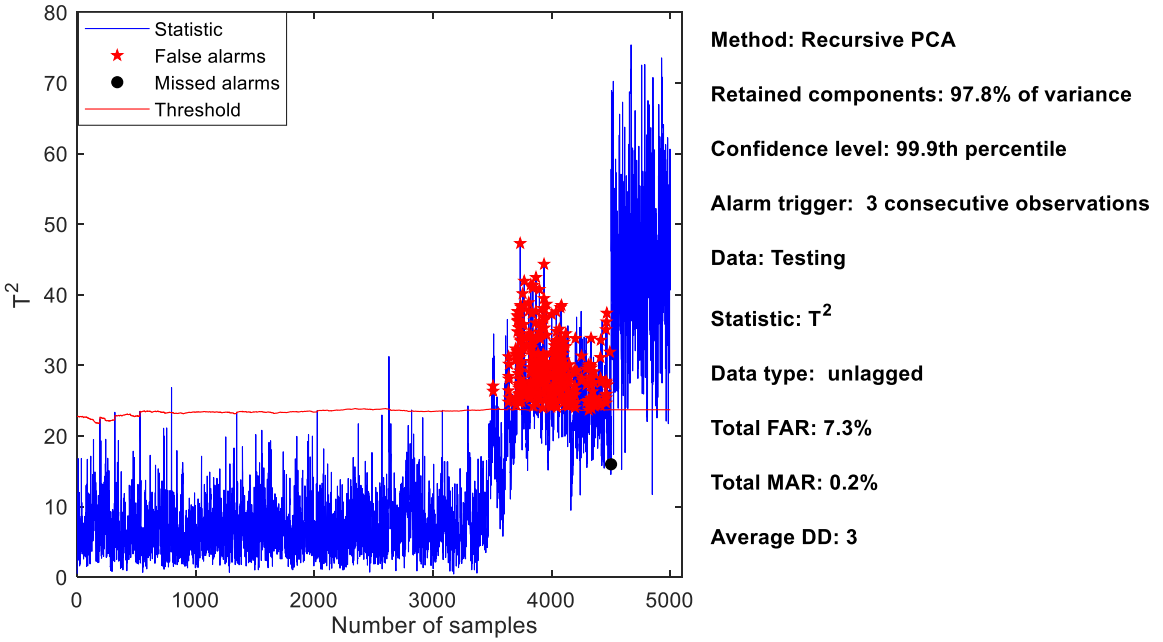


Figure E.7: RePCA T^2 statistic for Fault C04.

Figure E.8 shows the FAR results for MWPCA and RePCA SPE statistics. General trend shows MWPCA having a better performance overall. Performance of RePCA tend to suffer more as

the window size increases; increase in retained PCs, as well as z values, provides a reduced amount of alarm rates. MWPCA seems to suffer less from the window size changes. However, there is an obvious reduction in alarm rate with large z values and six retained principal components.

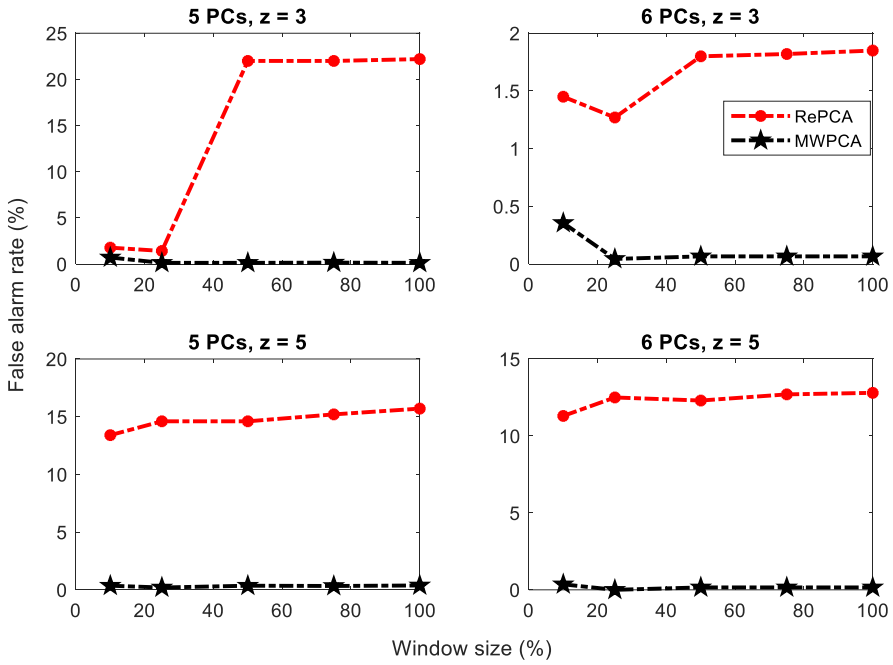


Figure E.8: Comparison of MWPCA and RePCA SPE statistic for Fault C04, evaluated at specified window sizes, retained PCs, and z values at 99.5th percentile.

Figure E.9 shows the MAR results for MWPCA and RePCA SPE statistic. The overall trend shows MWPCA having worse values in almost all cases apart from specified values of 5 retained PCs and z = 3. The performance, however, gets better at the latter end of the window size. RePCA maintains an almost flat value of zero in most cases because the model would have failed to cope with the drift and start flagging false alarms and eventually all the subsequent faults are detected. This is because the process drifts sets as a precedent which moves the statistics for the observations before the fault to high values (close to and above the threshold) and therefore makes the fault easily detectable.

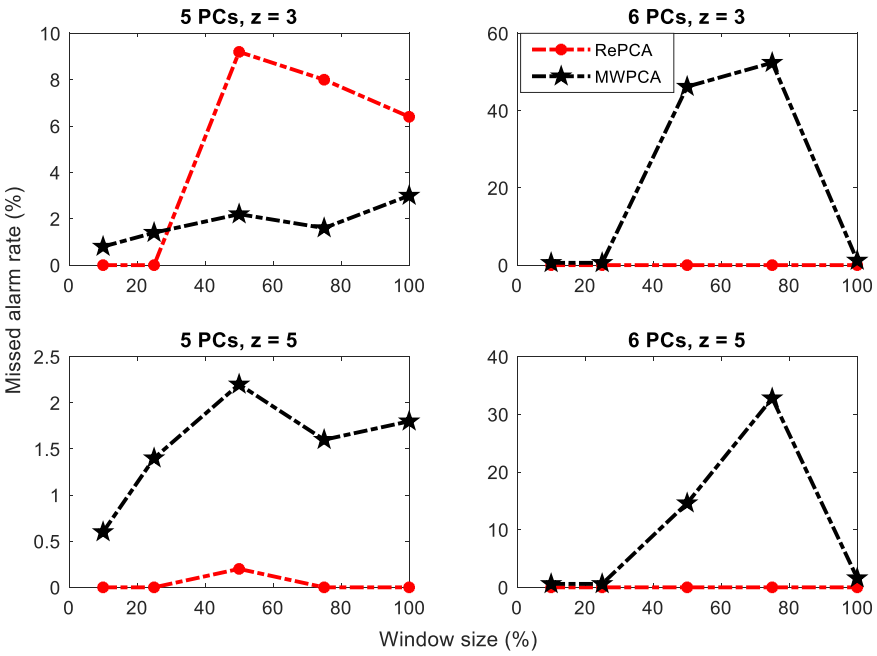


Figure E.9: Comparison of MWPCA and RePCA SPE statistic for Fault C04, evaluated at specified window sizes, retained PCs, and *z* values at 99.5th percentile.

FAR for the T^2 values for MWPCA and RePCA (in Figure E.10) shows an overall better performance for MWPCA with little influence from changes in window size, retained PCs and *z* values. RePCA once again performs poorly, however, the FAR tends to decrease with an increase in window size as opposed to the SPE statistic.

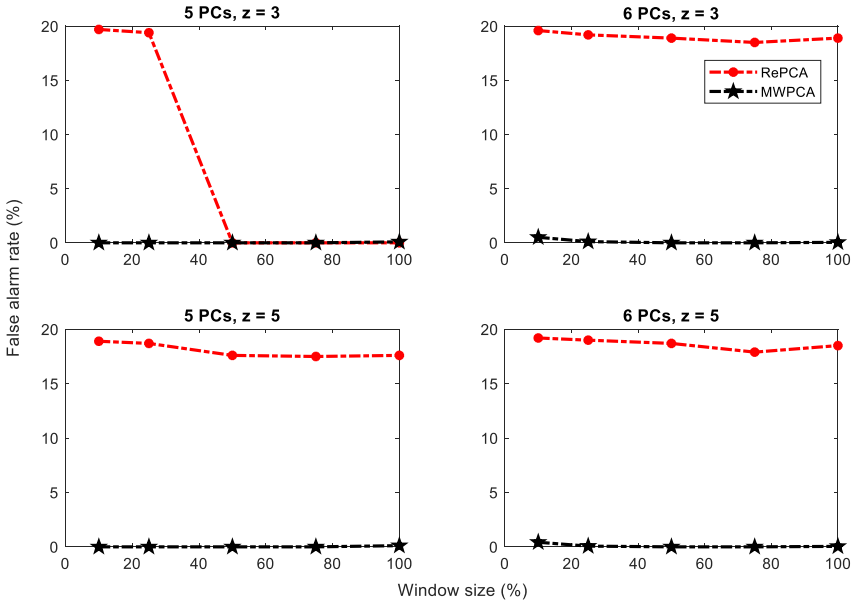


Figure E.10: Comparison of MWPCA and RePCA T^2 statistic for Fault C04, evaluated at specified window sizes, retained PCs, and *z* values at 99.5th percentile.

Detection delay results for the T^2 values of MWPCA and RePCA (in Figure E.11) shows almost-constant values with a change in window size; except for few instances where there is deviation before 50% of the initial window size. Results for thresholds computed at 99.5th and 99.9th percentile confidence levels (cl) are comparable for both 5 and 6 retained PCs; with few instances of deviation when the initial window size is less than 25%. Detection delay values for greater *z* values are higher.

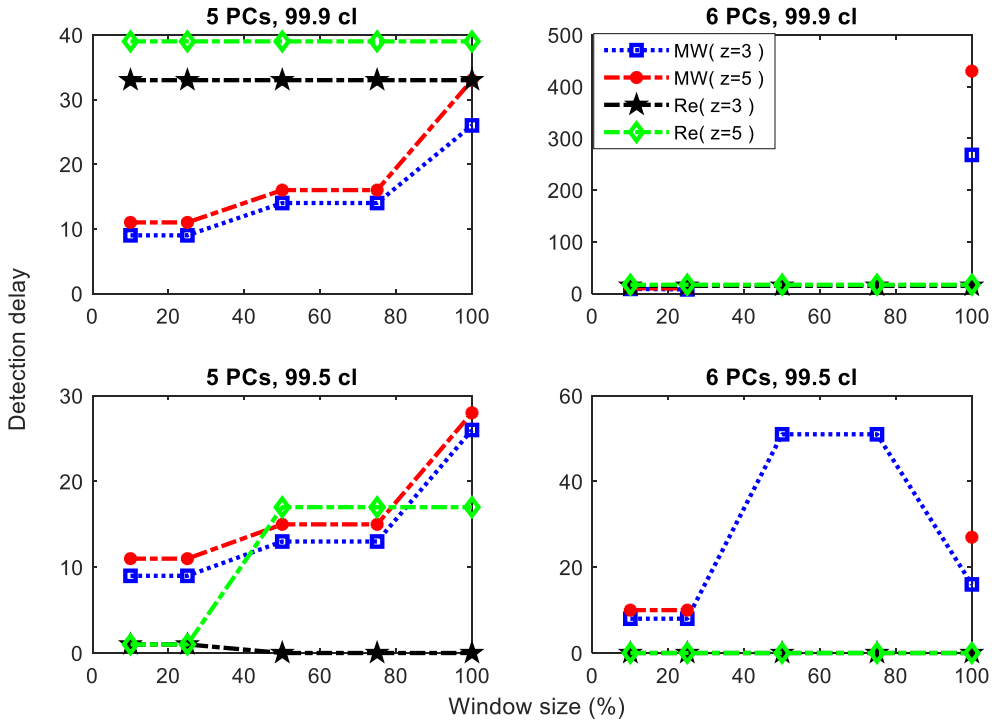


Figure E.11: DD of MWPCA (MW) and RePCA (Re) T^2 statistic for Fault C04, evaluated at specified window sizes, PCs, and *z* values.

Detection delay results for the SPE statistic of MWPCA and RePCA (in Figure E.12) shows higher DD values for RePCA than MWPCA at 5 retained PCs and 99.9th percentile; while the values for RePCA seem not to change much with initial window size, the values for MWPCA increases with the increase in window size. This could be interpreted as the increase in window size effect on the model responding quickly to changes, for larger initial window sizes, it takes some time for the model to sense changes in observations as unusual.

At the same retained PCs and reduced threshold (99.5%), MWPCA results show a general increase as window size increases. Values for RePCA, however, show almost-zero DD values for higher window size values and $z = 3$. This is a result of the thresholds for the RePCA being

challenged at the initial stage of the drift; with the initial wrong flagged drift values serving as a precedent for the quick detection in those cases.

The DD values reported for the retained PCs value of 6 and 99.9% percentile (confidence level) showed almost-constant zero values for all RePCA values because of the drift results in a lot of false alarms which serve as a precedent for the quick detection. The MWPCA, however, fails to detect the fault in the instances where there are blank values (DD is infinity). The best results achievable for 6 retained PCs, in this case, is at the 99.5th percentile.

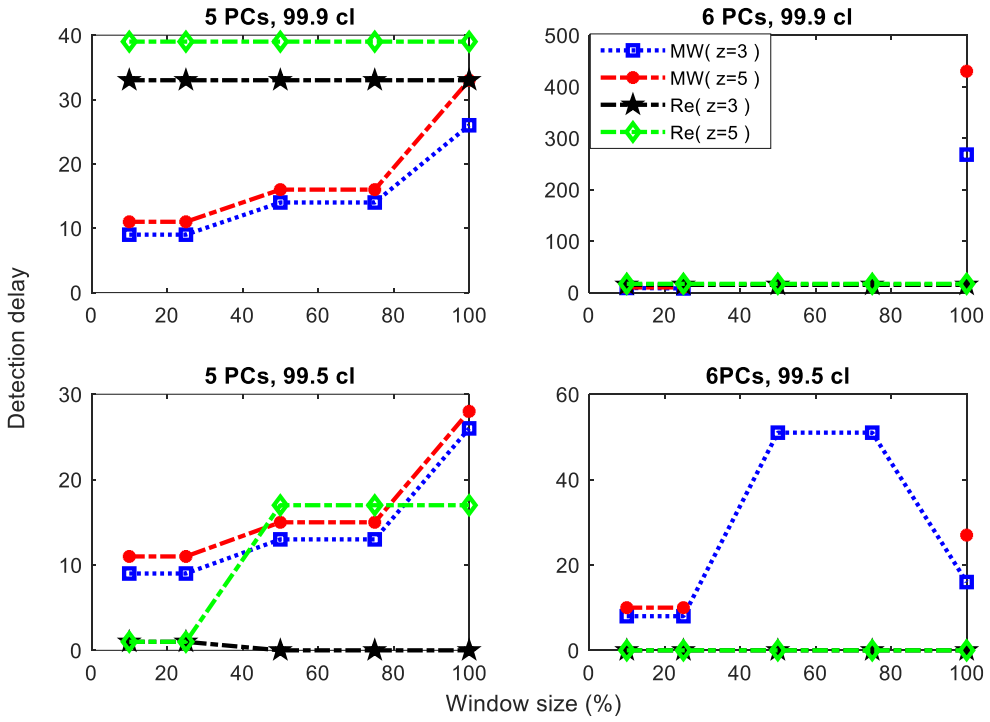


Figure E.12: DD of MWPCA (MW) and RePCA (Re) SPE statistic for Fault C04, evaluated at specified window sizes, PCs, and *z* values.

APPENDIX F: ADAPTATION TECHNIQUES FOR MODEL UPDATES

Appendix F provides information on the comparison of the different model update techniques for APCA and the fault detection performance at different hyperparameter values.

F.1 Fault detection for Fault C03

The subsequent figures show the performance of the respective approaches. The poorest performance at this stage is that for monitoring UM-3 mainly due to a relatively low threshold value.

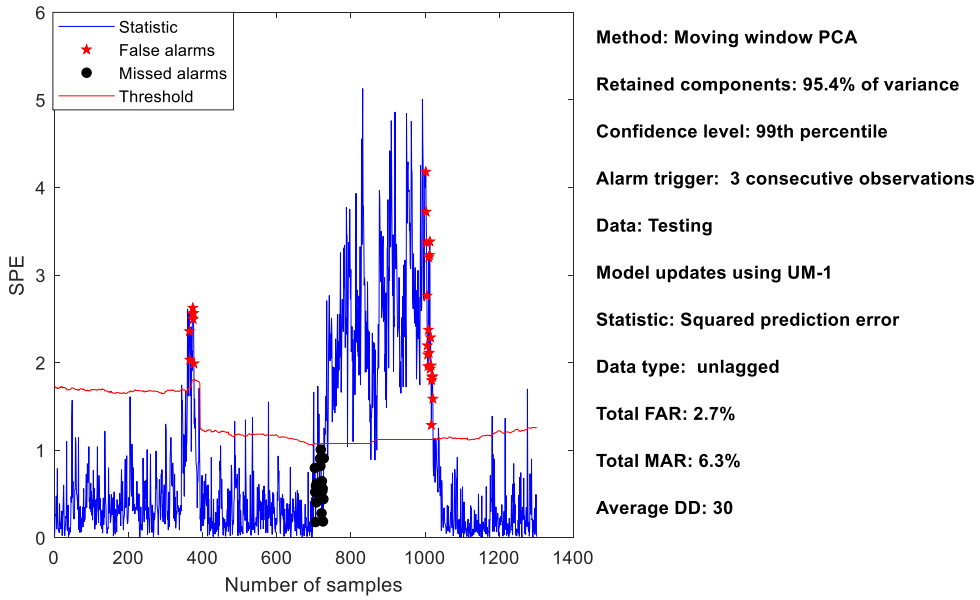


Figure F.1: SPE statistic for UM-1.

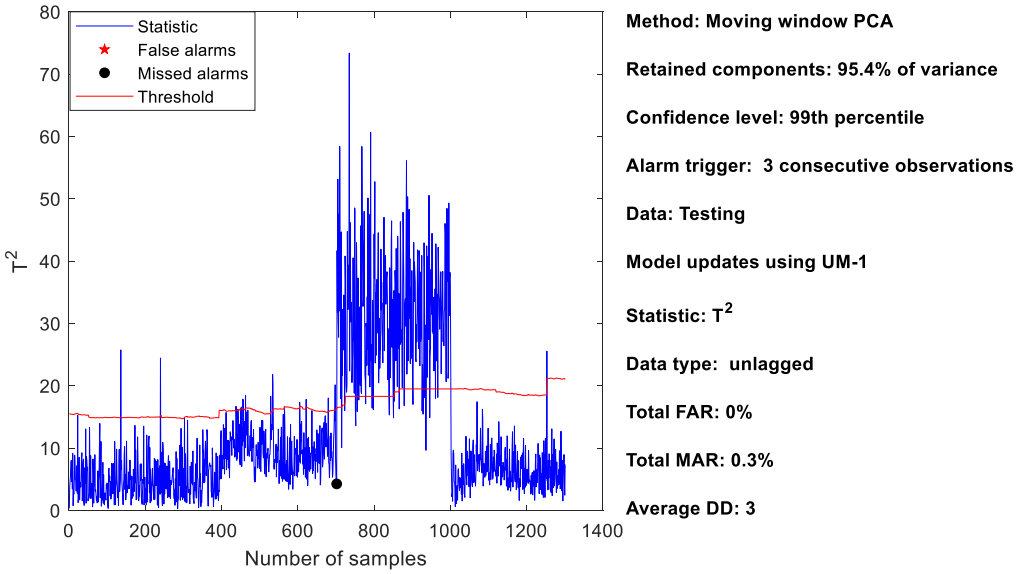


Figure F.2: T^2 statistic for UM-1.

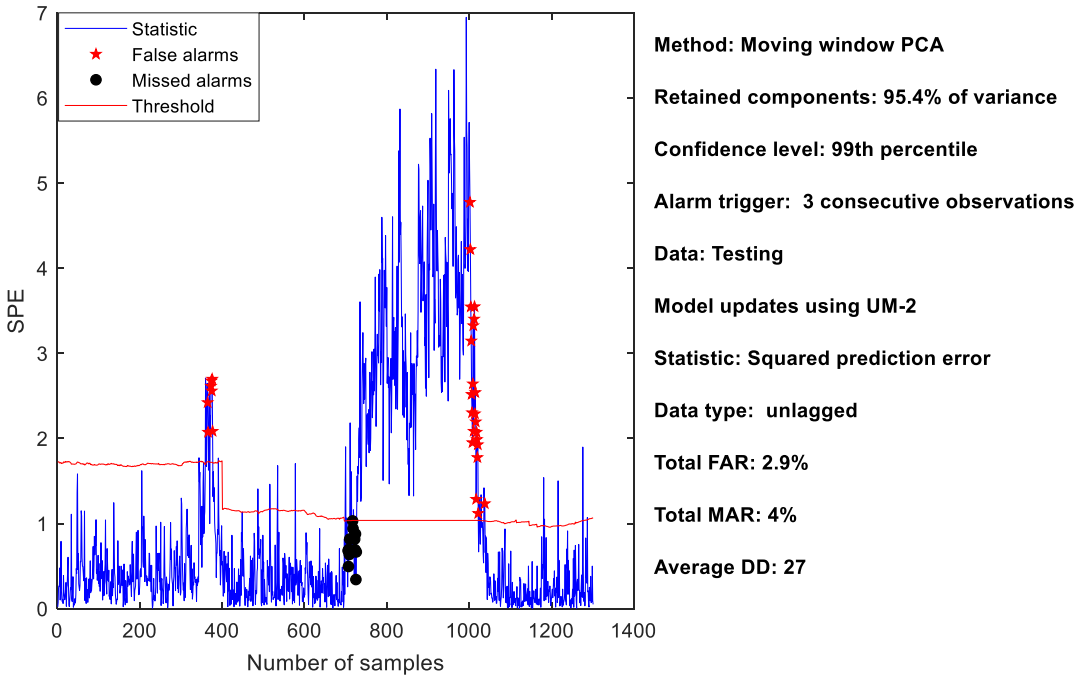


Figure F.3: SPE statistic for UM-2.

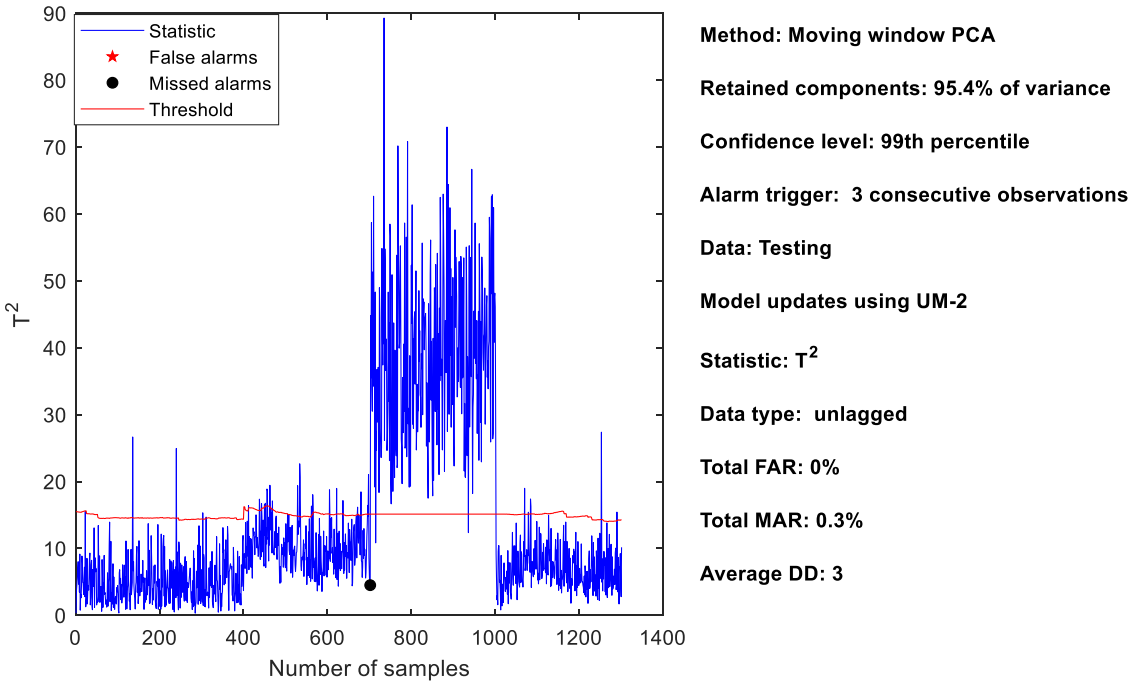


Figure F.4: T^2 statistic for UM-2.

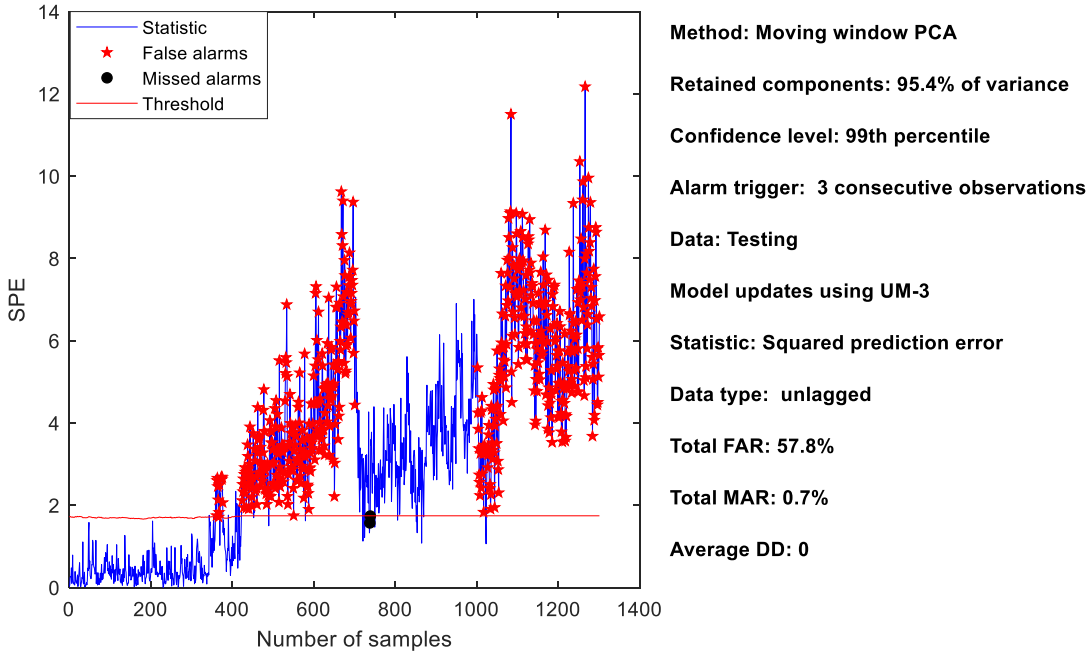


Figure F.5: SPE statistic for UM-3.

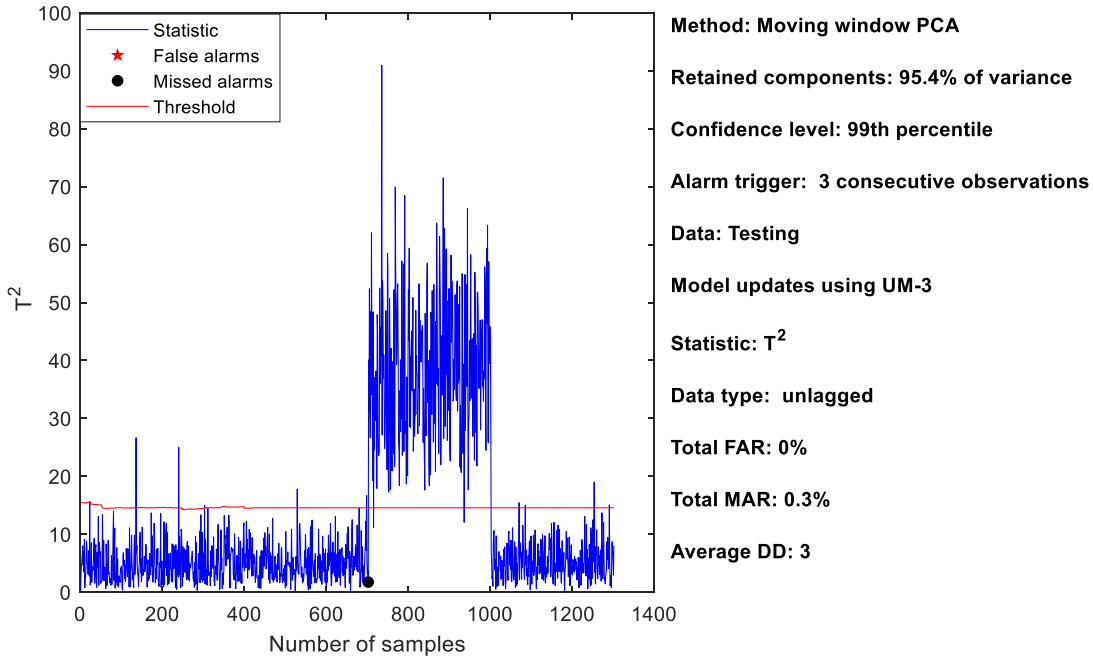


Figure F.6: T^2 statistic for UM-3.

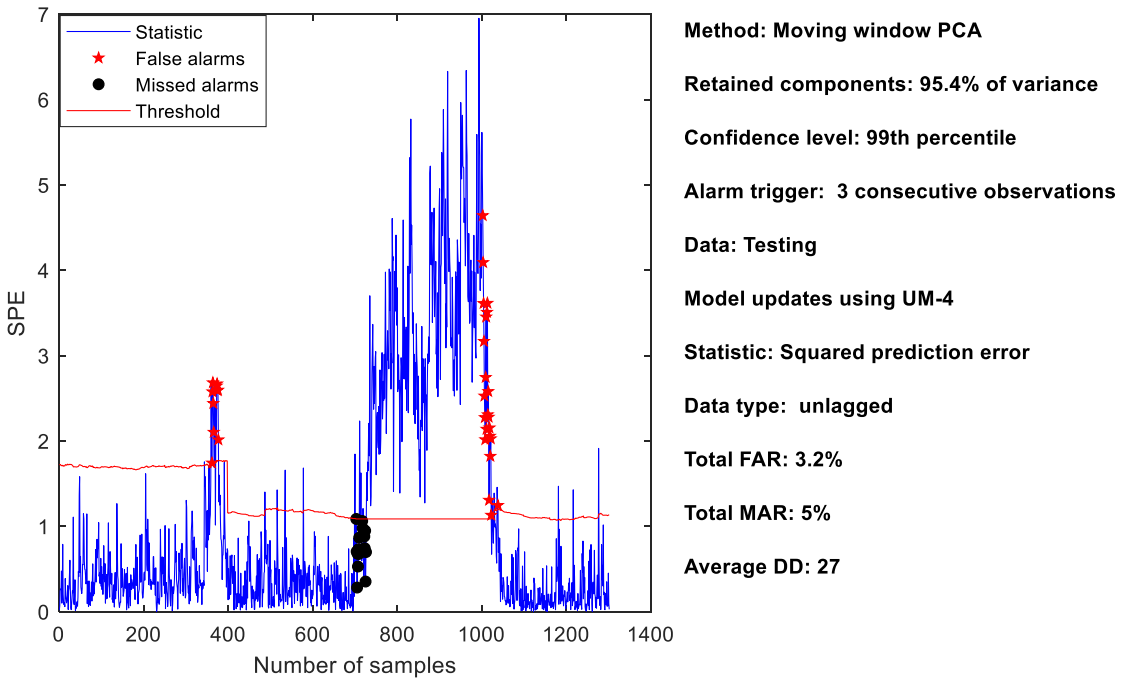


Figure F.7: SPE statistic for UM-4.

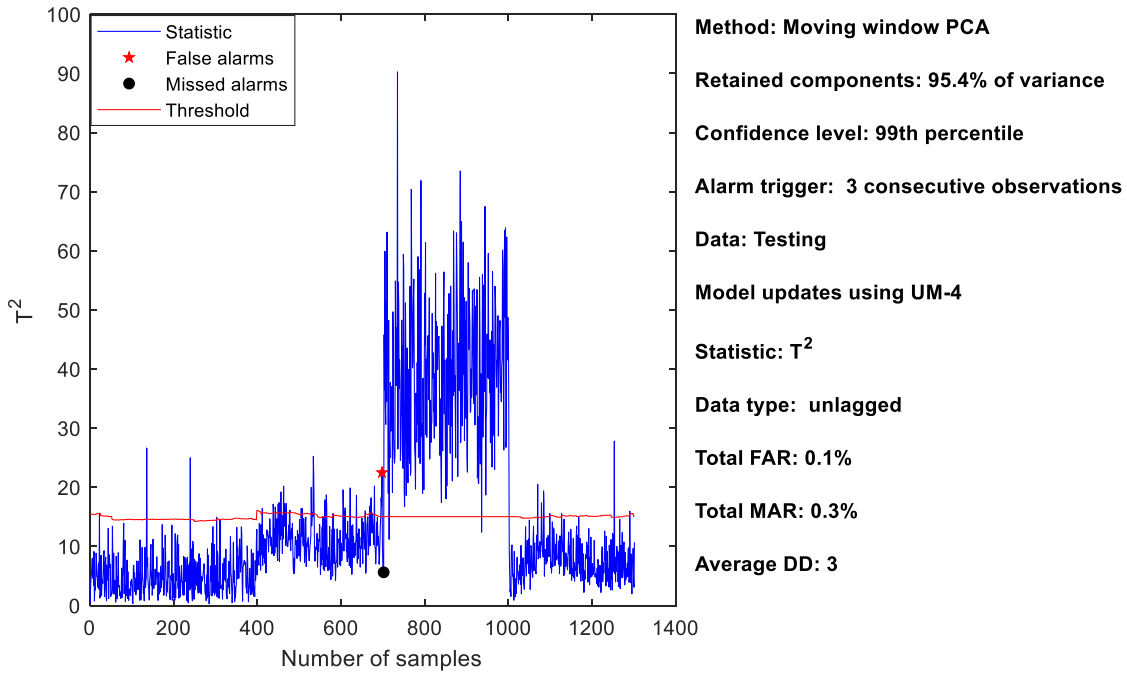


Figure F.8: T^2 statistic for UM-4.

However, increasing the threshold to the 99.5th percentile makes the UM-3 capable to respond to the drift. The improved results for the SPE statistic are shown in Figure F.9. UM-1 is the method that suffers in this case. The method continues to update until an alarm is triggered; the

consequence of the observations' statistics not being consecutive enough to cause an alarm trigger results in the inability to detect the fault as shown in Figure F.10.

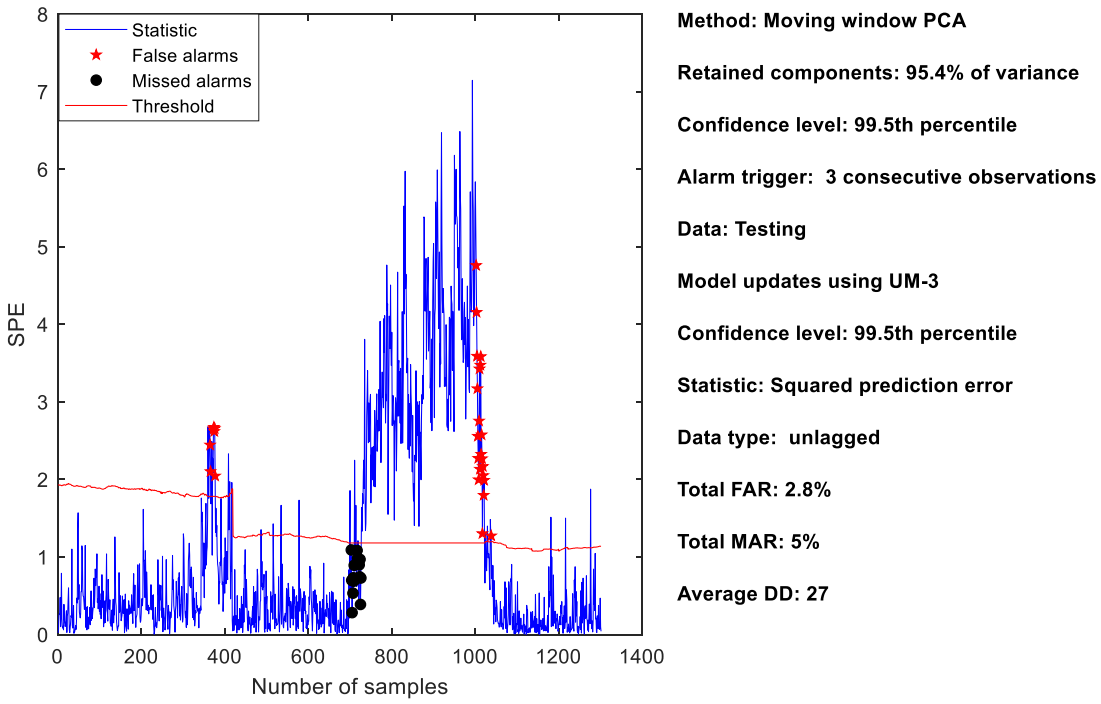


Figure F.9: SPE statistic for UM-3.

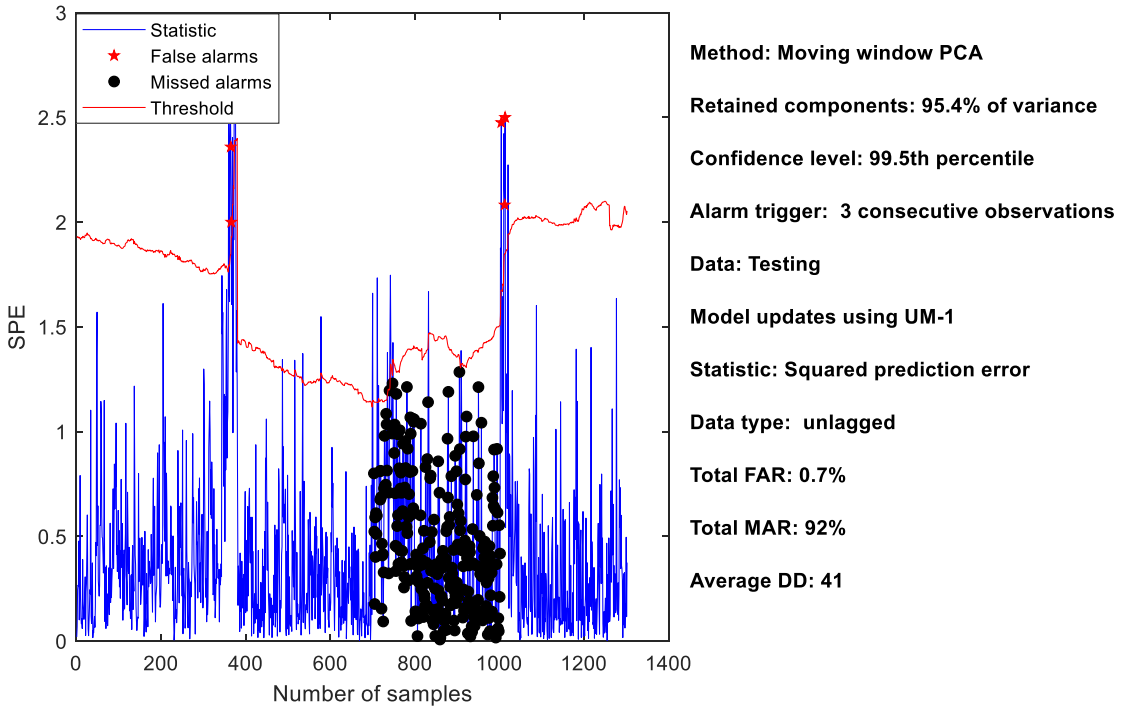


Figure F.10: SPE statistic for UM-1.

The MAR and FAR results (shown in Figure F.11 and Figure F.12 respectively) show a satisfactory performance for all update methods apart from UM-1. All FAR for other update methods produced the same results. Consequently, UM-2 and UM-3 performed slightly better when the total alarm rates for the T^2 statistic are considered.

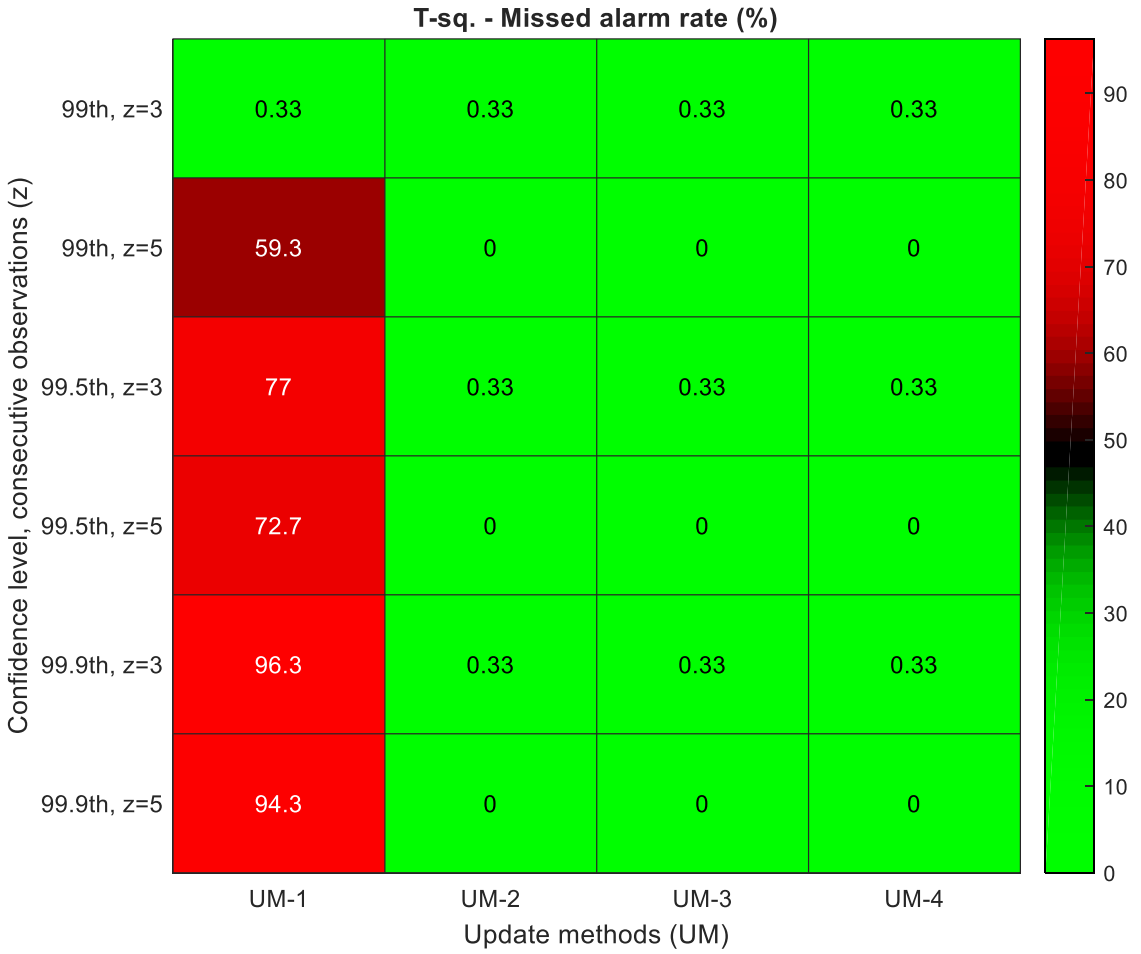


Figure F.11: Missed alarm rates for T^2 statistic evaluated at specified levels.

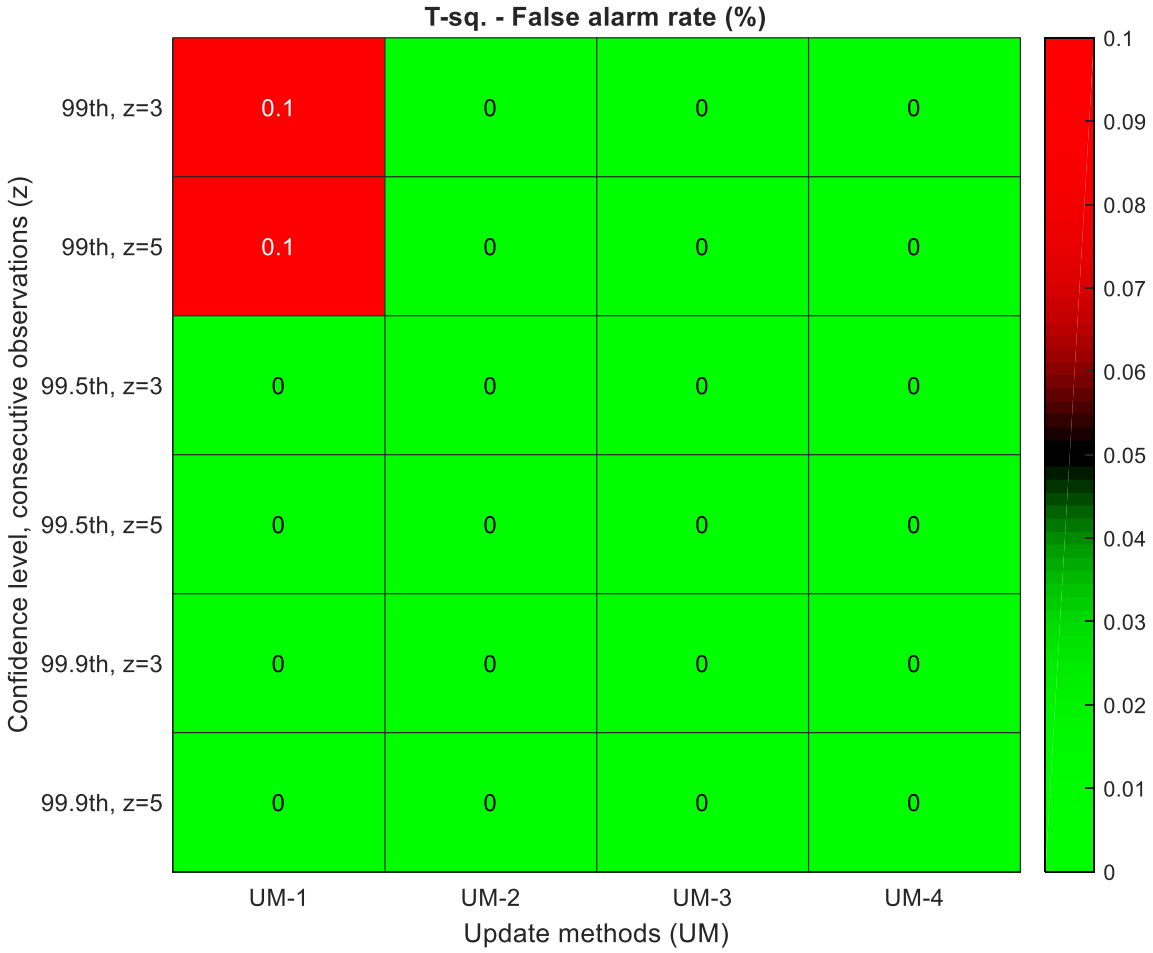


Figure F.12: False alarm rates for T^2 statistic evaluated at specified levels

F.2 CSTR process Fault C05

Ct00 and Cv00 are used as training and validation data respectively. The test data C05 is generated by introducing a drift from time 102 to 602 after which 4 intermittent faults are introduced from times (602 - 607, 617 - 622, 632 - 637, 647 - 651), and a step fault from time 702 to 1001, after which the process returns to NOC. All the step and intermittent faults occur as 3% change in reactor inlet temperature Ti. Figure F.13 shows the simulation results for C05 with periods of the drift and faults, and a response from the manipulated variables Fc and Fa.

Table F.1: Summary of simulated CSTR data.

Data	Description	Samples
C05	Process drift, intermittent faults, and step in Ti, and back to NOC	1301

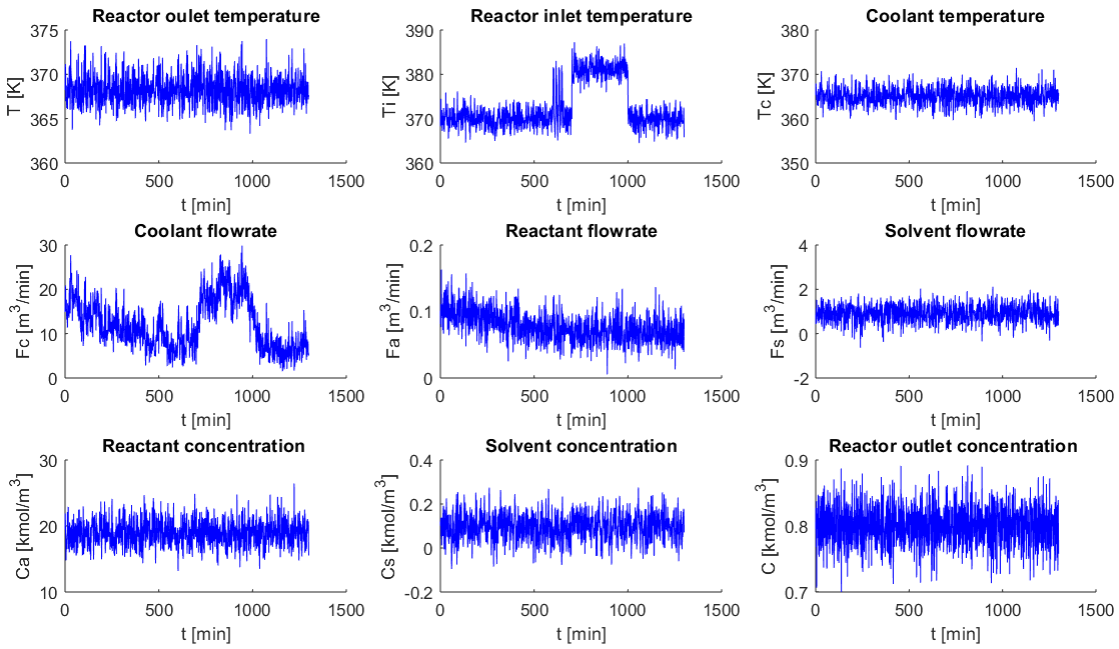


Figure F.13: Simulation results for process drift, intermittent and step in Ti with the response from manipulated variables Fc and Fa.

F.3 Fault detection for Fault C05

Fault C05 as discussed earlier is a step of difficulty level above that for Fault C03 which is due to the intermittent faults before a permanent fault (step fault). In a non-adaptive case, when such intermittent faults occur and do not contribute to an alarm-trigger case as they may be short-lived, it may not a major concern. However, in the adaptive case, the ability of the model to judge that although no alarm is triggered, the observation is unfit to cause an update is key. The consequence of the model incorporating the observations when an alarm is not triggered impacts its ability to correctly judge a similar fault of that nature. It, however, depends on the period between the intermittent faults and permanent fault, the effect of incorporating the intermittent fault observations could be neutralized if more new observations are incorporated before a permanent failure.

Sample results for the various approaches are presented in the subsequent figures. UM-1 is the worst performer at this stage. This is mainly due to UM-1's inability to sense the initiation of the intermittent faults and then with the continuous update, the permanent failure (step fault) remained undetected. This is observable at the threshold and retained PCs level where it could have usually detect a fault if not for the intermittent faults.

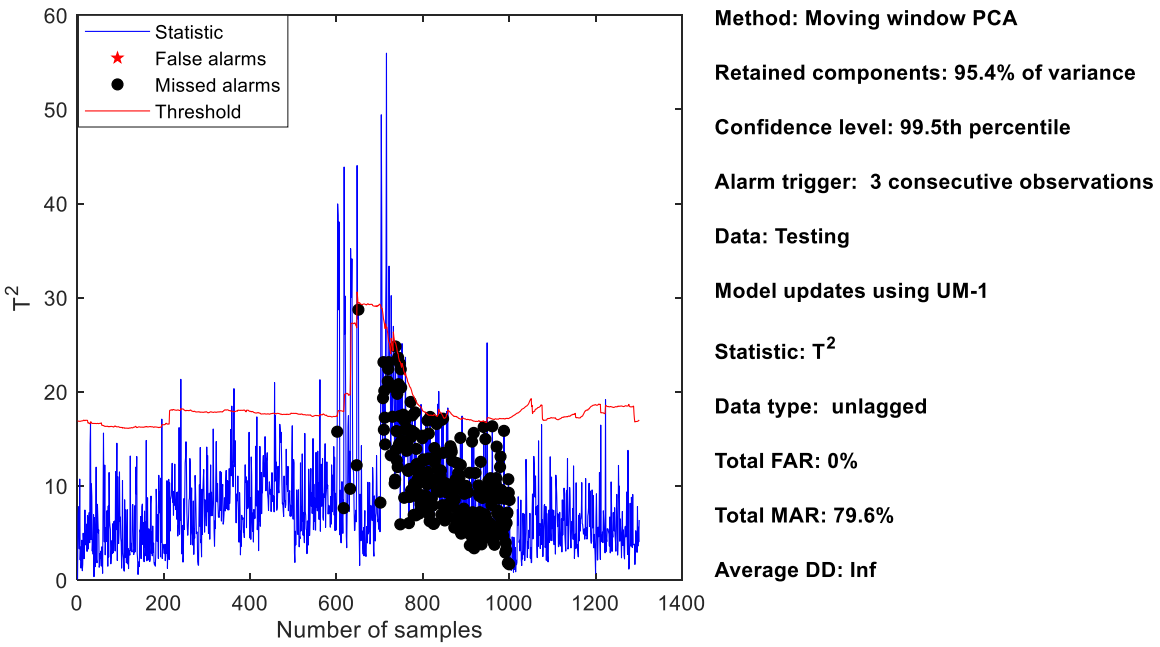


Figure F.14: T^2 statistic for UM-1.

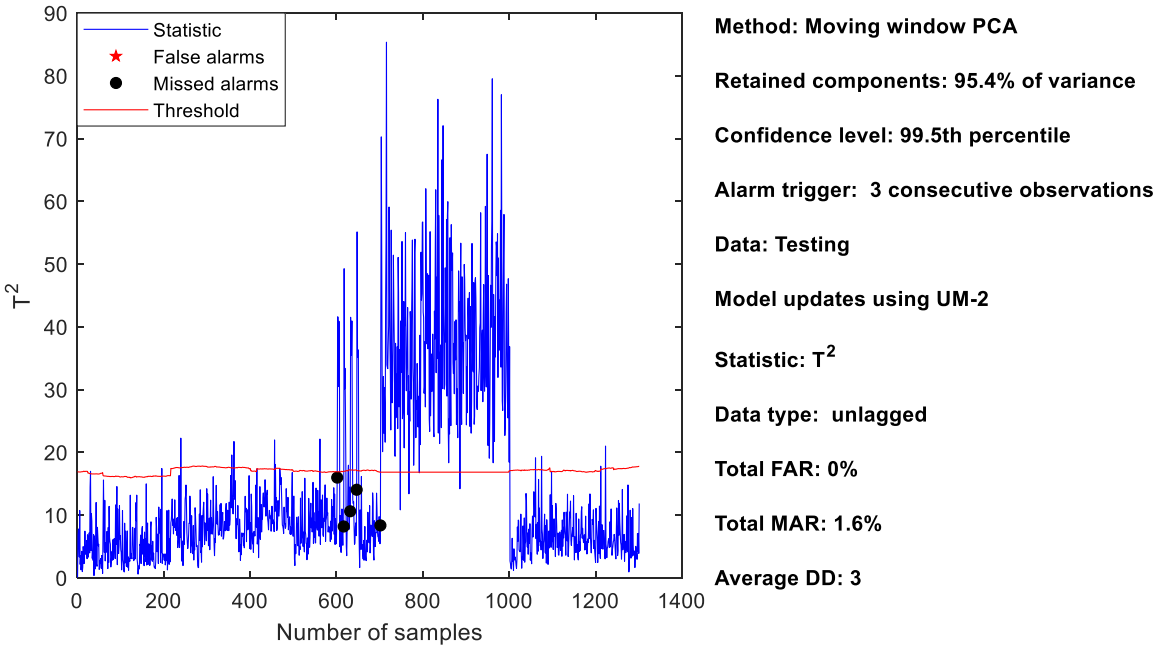


Figure F.15: T^2 statistic for UM-2.

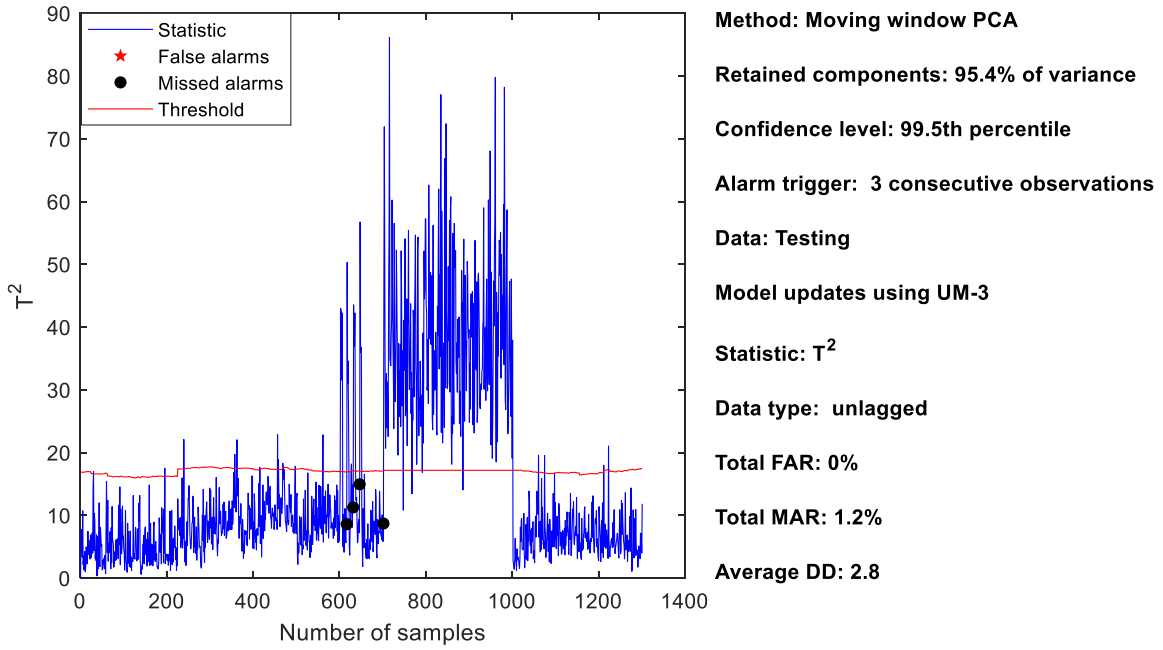


Figure F.16: T^2 statistic for UM-3.

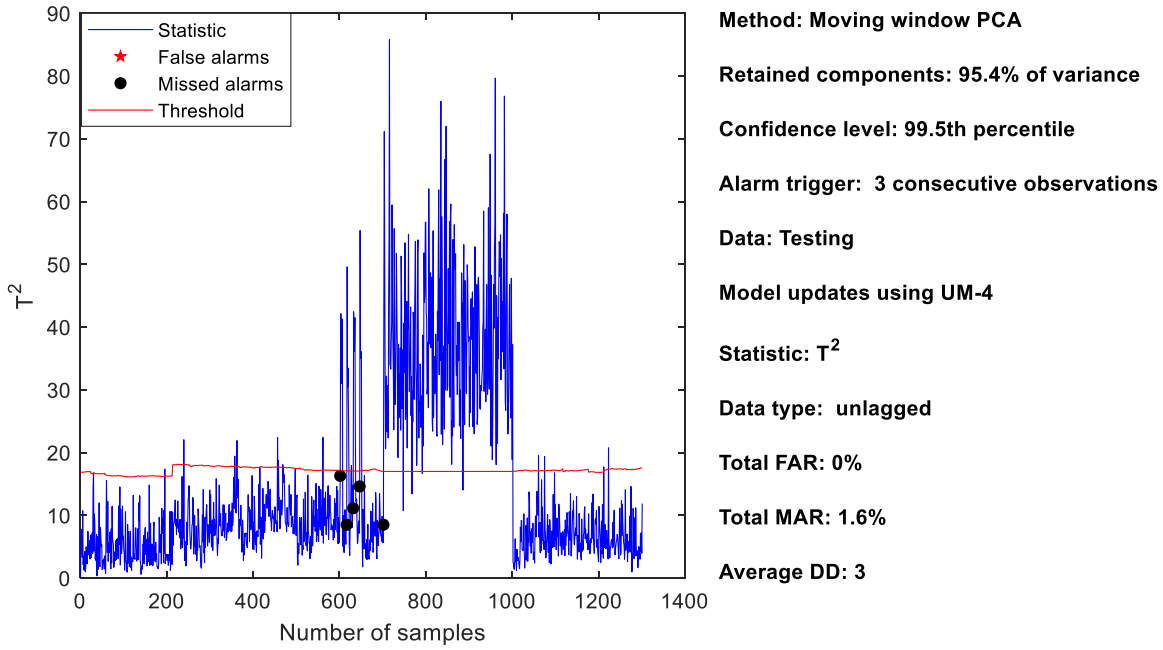


Figure F.17: T^2 statistic for UM-4.

Figure F.18 shows the SPE statistic alarm rates for update methods two to four. The results depict expected decrease in FAR and increase in MAR as threshold increases. UM-3 produced the lowest missing alarm rates and consequently the highest FAR in all cases. UM-2 and UM-4 produces overlapping results for the MAR; UM-4 however produced reduced FAR in all cases. This makes UM-4 and UM-2 similar performers with UM-4 having a slight edge.

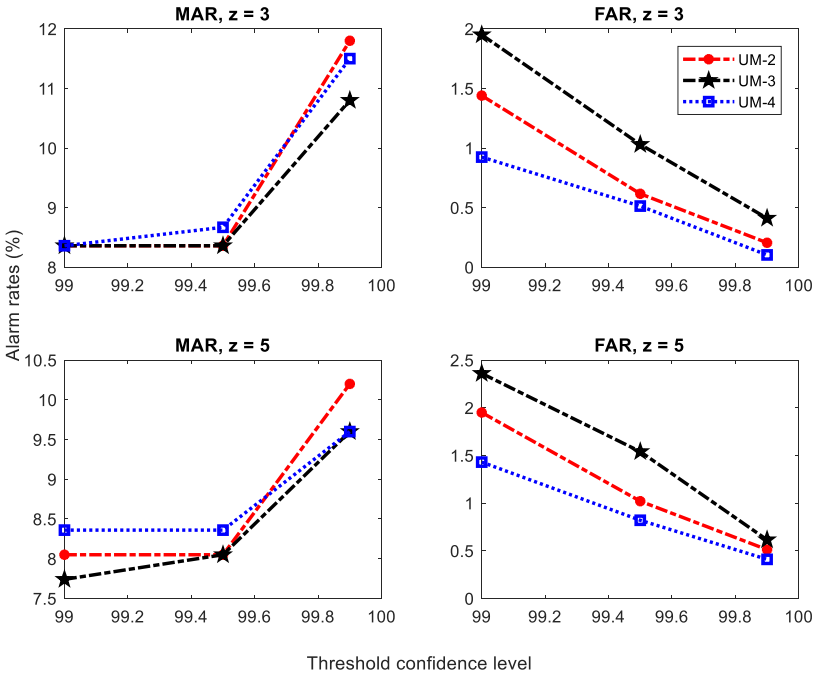


Figure F.18: Alarm rates of SPE statistic for the update methods.

The T^2 statistic (shown in Figure F.19) presents similar results to that achieved for the SPE statistic with UM-2 and UM-4 achieving close results once more. The 99.5th percentile is however seen to provide an overall balance between the alarm rates at each instance.

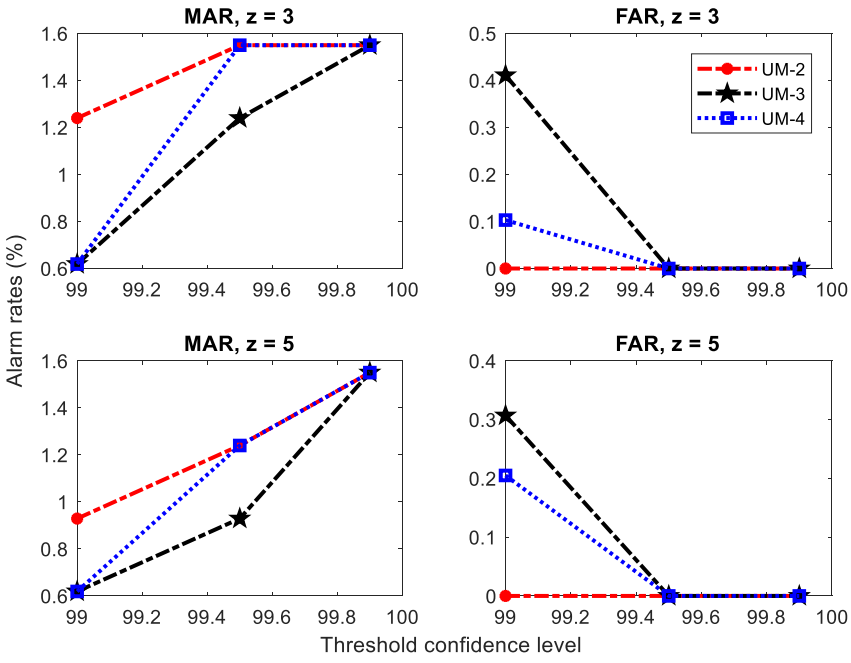


Figure F.19: Alarm rates of T^2 statistic for the update methods.

Overall, all the update methods considered in the analysis produce good results with UM-2 and UM-4 providing significantly better results than UM-3. The ability to detect a process which

involves process drift with intermittent and step fault was quite handled well for the step change in Ti. The last analysis in this comparison involves that for Fault C06 which is presented in the next section.

F.4 CSTR process Fault C06

Test data C06 is similar to C05 but with the faults expressed in the concentration of reactant *a*, Ca. The magnitude of the fault was a 50% increase in the concentration value. Figure F.20 shows the simulation results for C06. Summary of the simulated CSTR data is shown in Table F.2.

Table F.2: Summary of simulated CSTR data.

Data	Description	Samples
C06	Process drift, intermittent and step in Ca, and back to NOC	1301

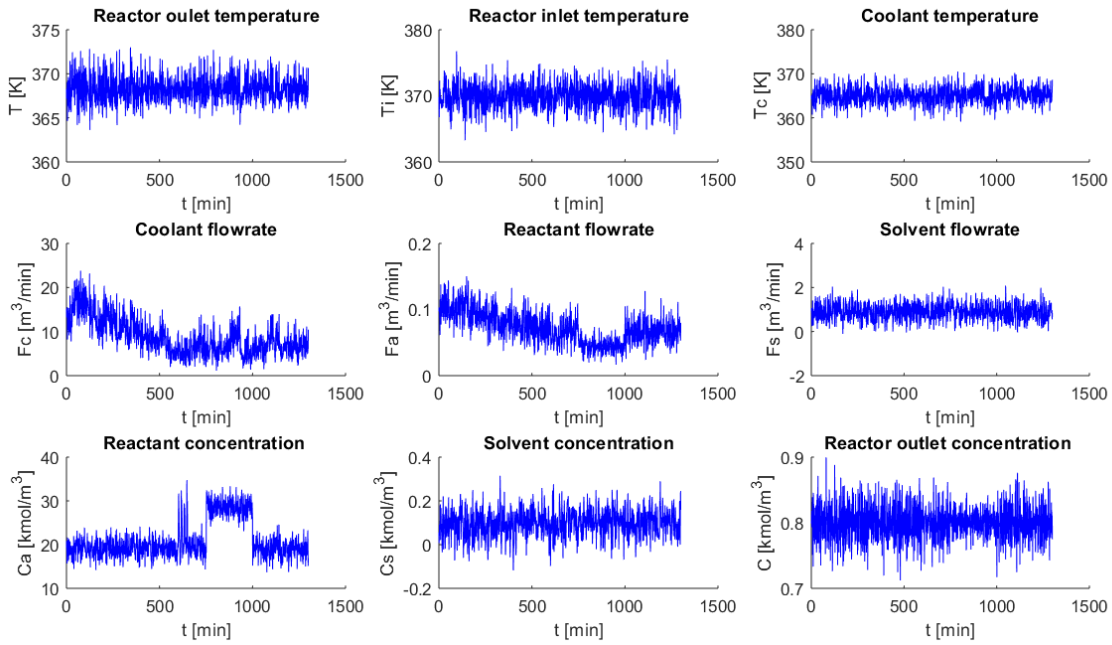


Figure F.20: Simulation results for process drift, intermittent and step in Ca with the response from manipulated variables Fc and Fa.

F.5 Fault detection for Fault C06

Fault C06 as discussed is similar to Fault C06 but with different fault type (step in Ca). The obvious challenge is to see the performance of the update methods for various runs and fault types.

Sample results for the various methods are presented in the subsequent figures. UM-3 shows the inability of the model to update due to the sensitivity of the SPE statistic to the drift resulting in a fixed threshold for the T^2 statistic as well. Results for UM-4 shows slightly better results than UM-2; which are the two best performing methods in this instance. UM-1, as expected had no challenge which models updating but rather with detecting when not to update.

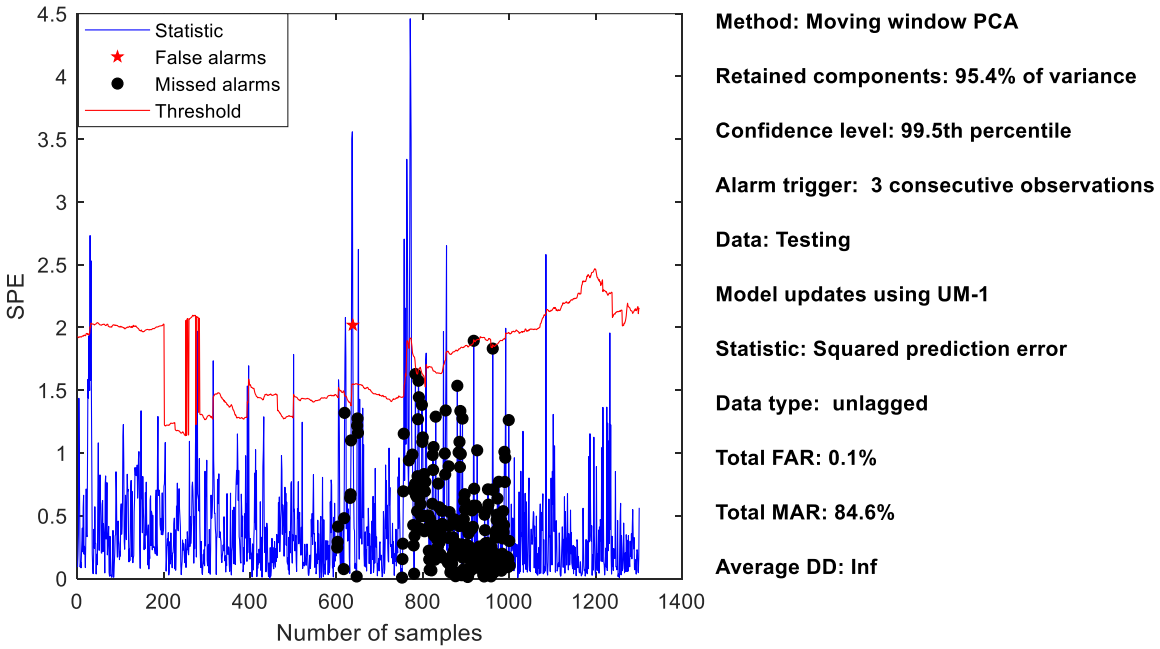


Figure F.21: SPE statistic for UM-1.

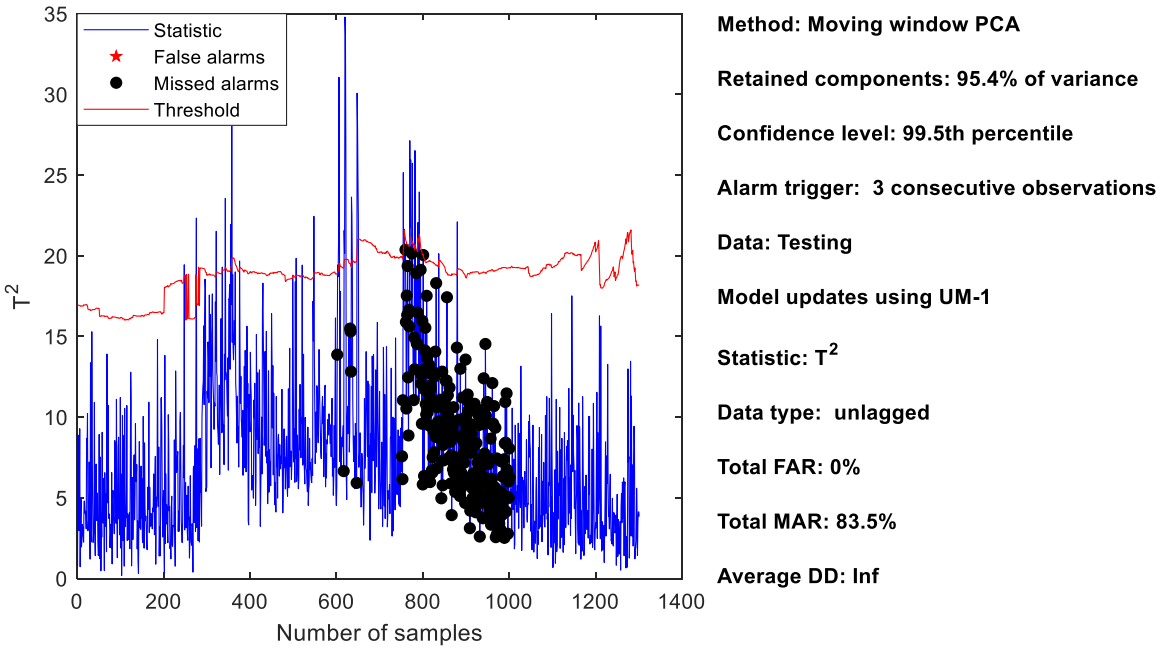


Figure F.22: T^2 statistic for UM-1.

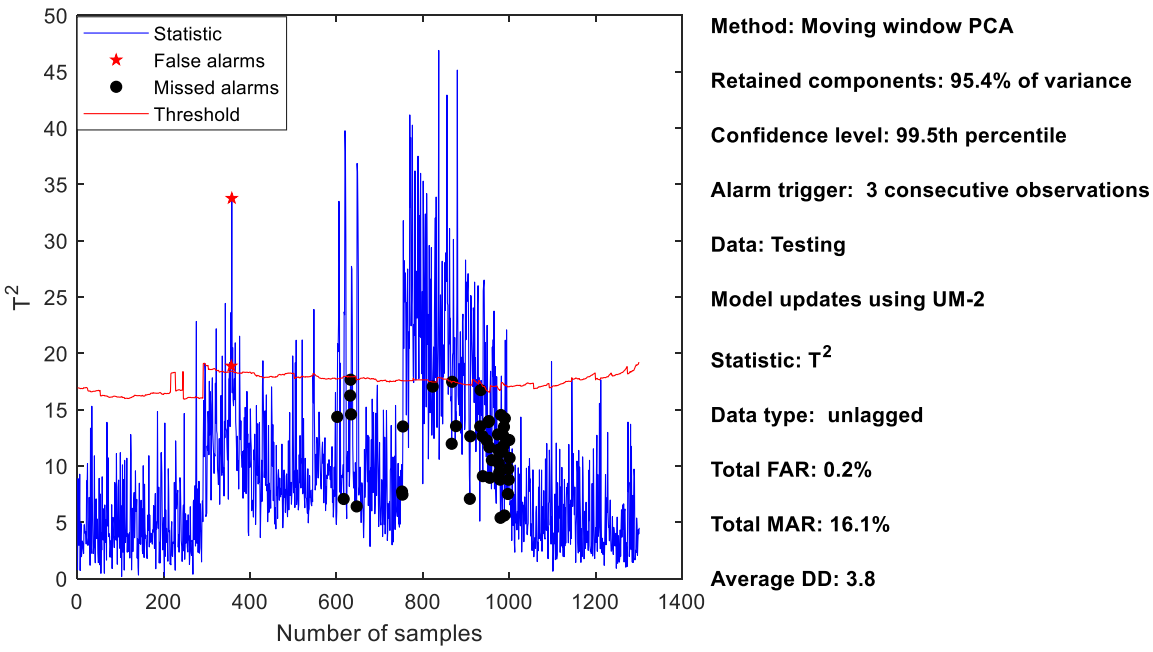


Figure F.23: T^2 statistic for UM-2.

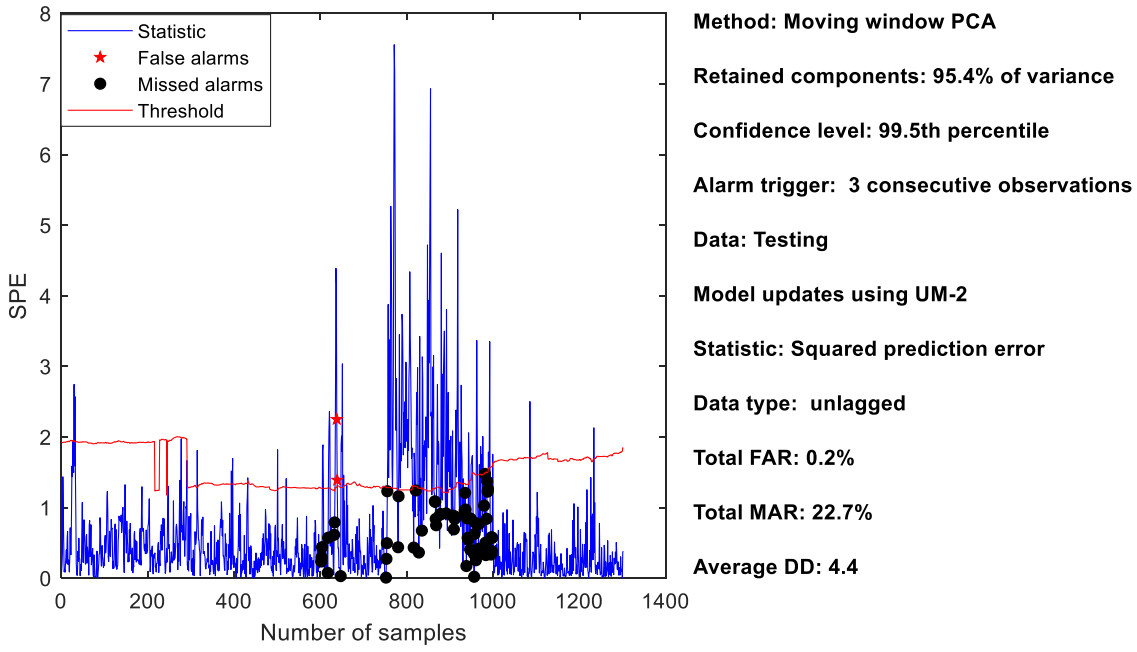


Figure F.24: SPE statistic for UM-2.

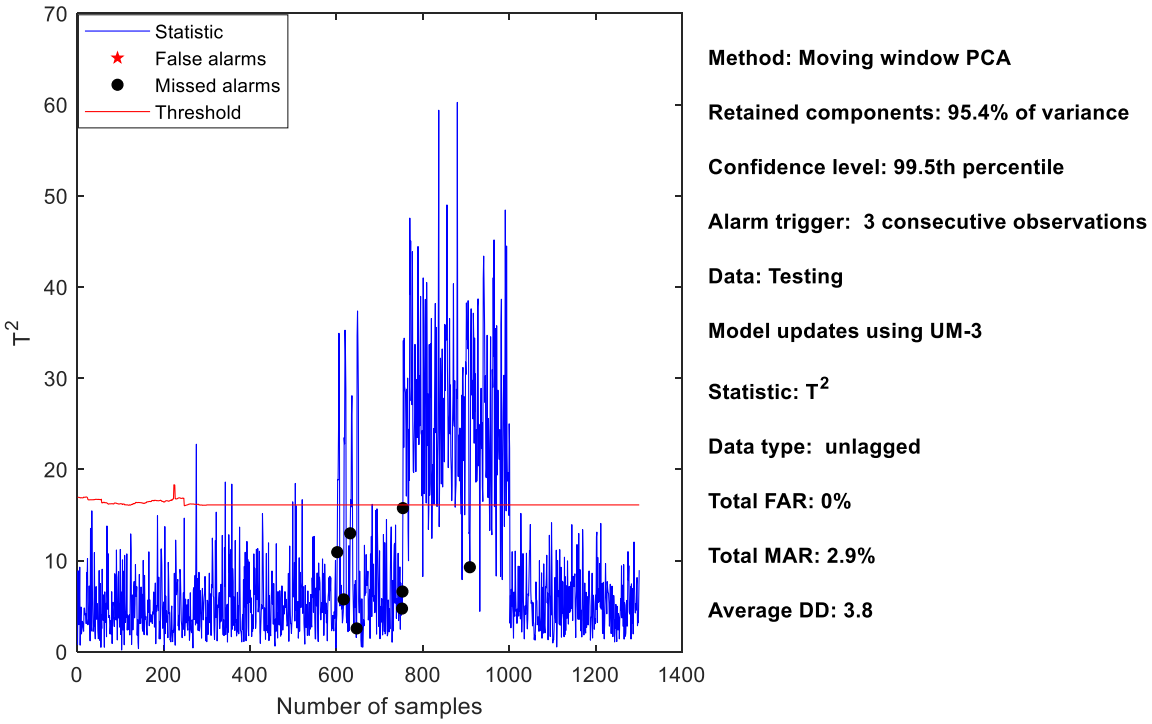


Figure F.25: T^2 statistic for UM-3.

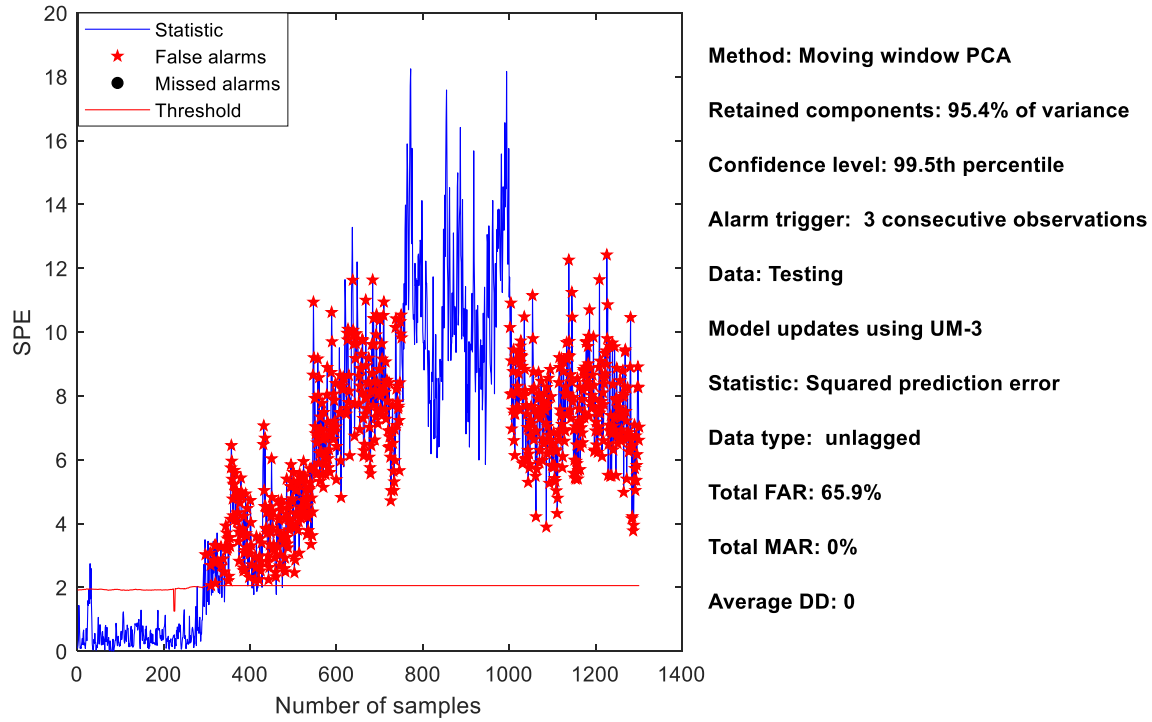
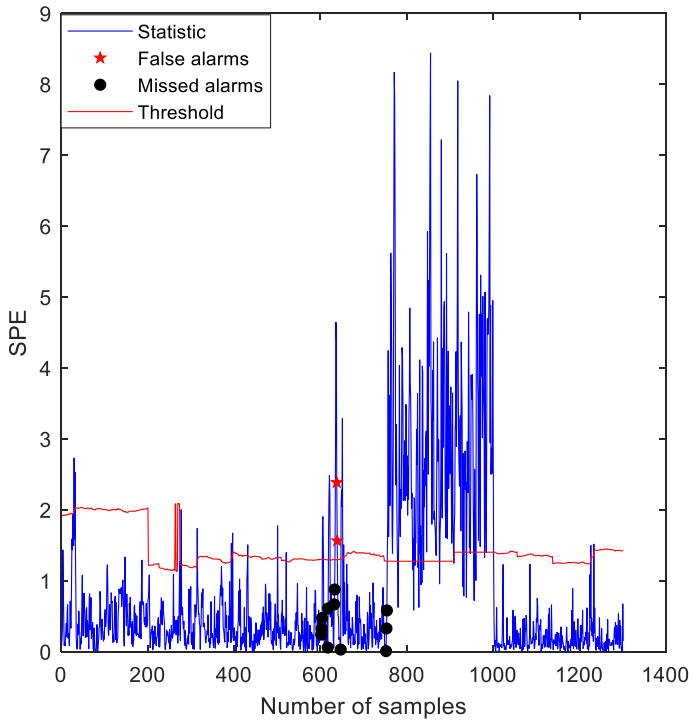


Figure F.26: SPE statistic for UM-3.



Method: Moving window PCA

Retained components: 95.4% of variance

Confidence level: 99.5th percentile

Alarm trigger: 3 consecutive observations

Data: Testing

Model updates using UM-4

Statistic: Squared prediction error

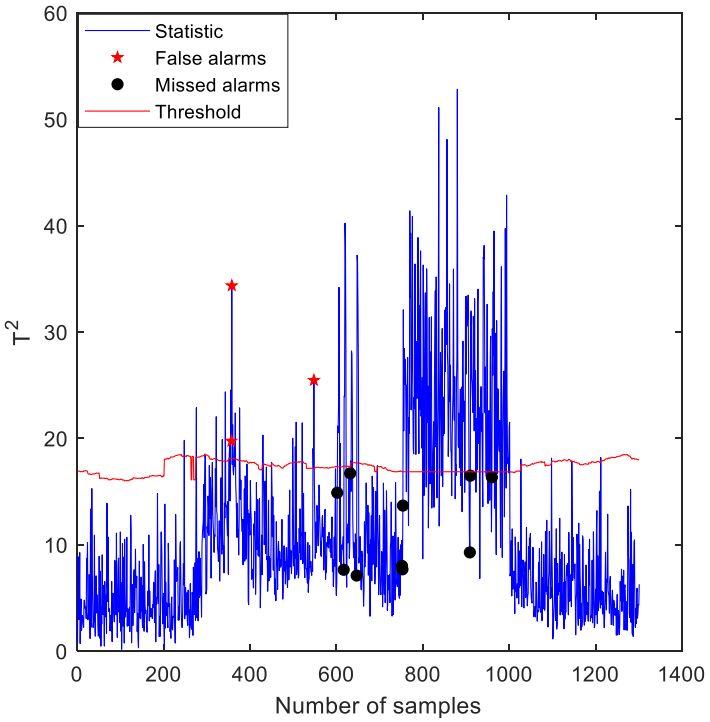
Data type: unlagged

Total FAR: 0.2%

Total MAR: 4%

Average DD: 4.2

Figure F.27: SPE statistic for UM-4.



Method: Moving window PCA

Retained components: 95.4% of variance

Confidence level: 99.5th percentile

Alarm trigger: 3 consecutive observations

Data: Testing

Model updates using UM-4

Statistic: T^2

Data type: unlagged

Total FAR: 0.3%

Total MAR: 3.7%

Average DD: 3.8

Figure F.28: T^2 statistic for UM-4.

MAR and FAR for T^2 statistic are respectively shown in Figure F.30 and Figure F.31 for all update methods. Similar to the SPE statistic, UM-1and UM-2 produced a high amount of missed alarms at elevated thresholds; with UM-1 showing total performance failure. UM-4 produced a high amount of MAR for the same instance as that encountered for the SPE statistic. All FAR recorded for all instances are low apart from that for 99th percentile with a *z* value of 5 for UM-4.

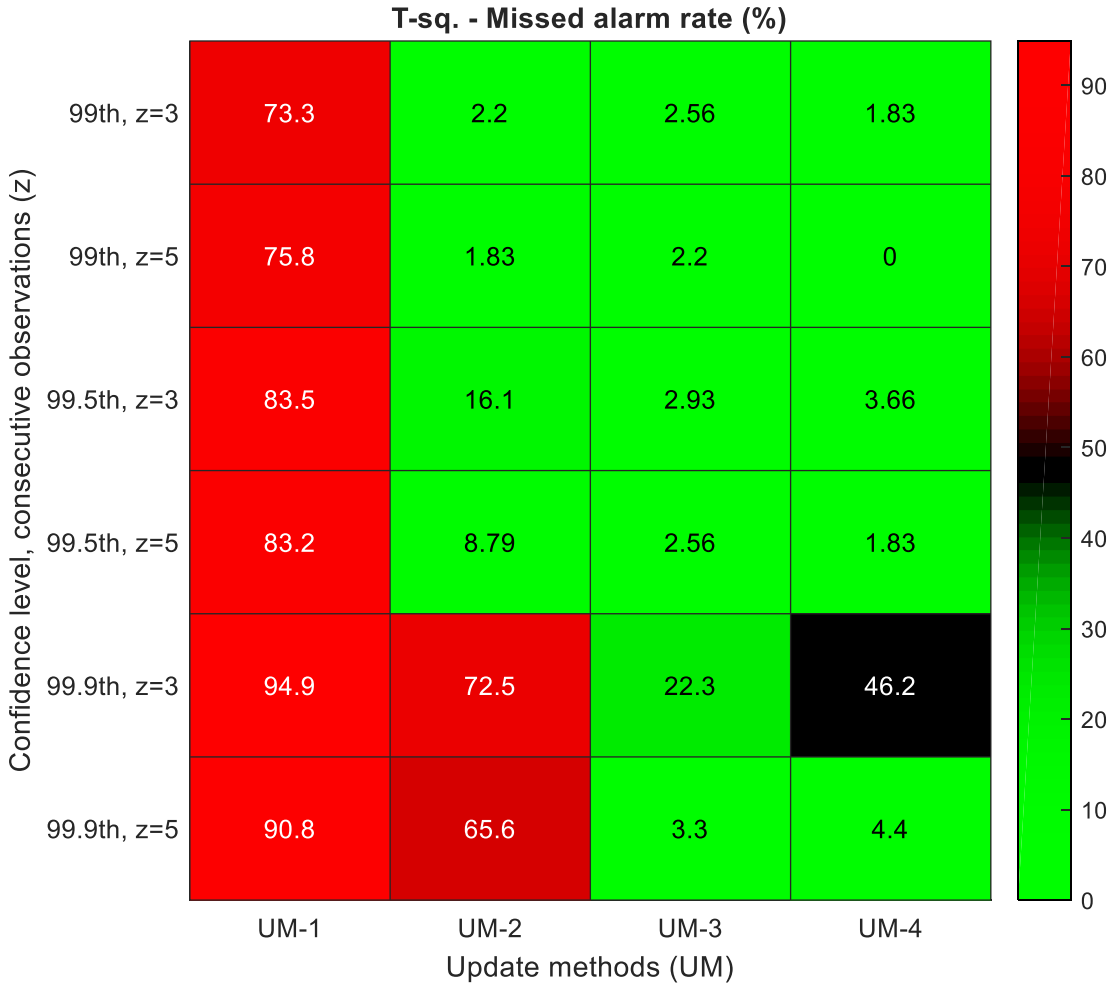


Figure F.29: T^2 statistic MAR for specified update methods.

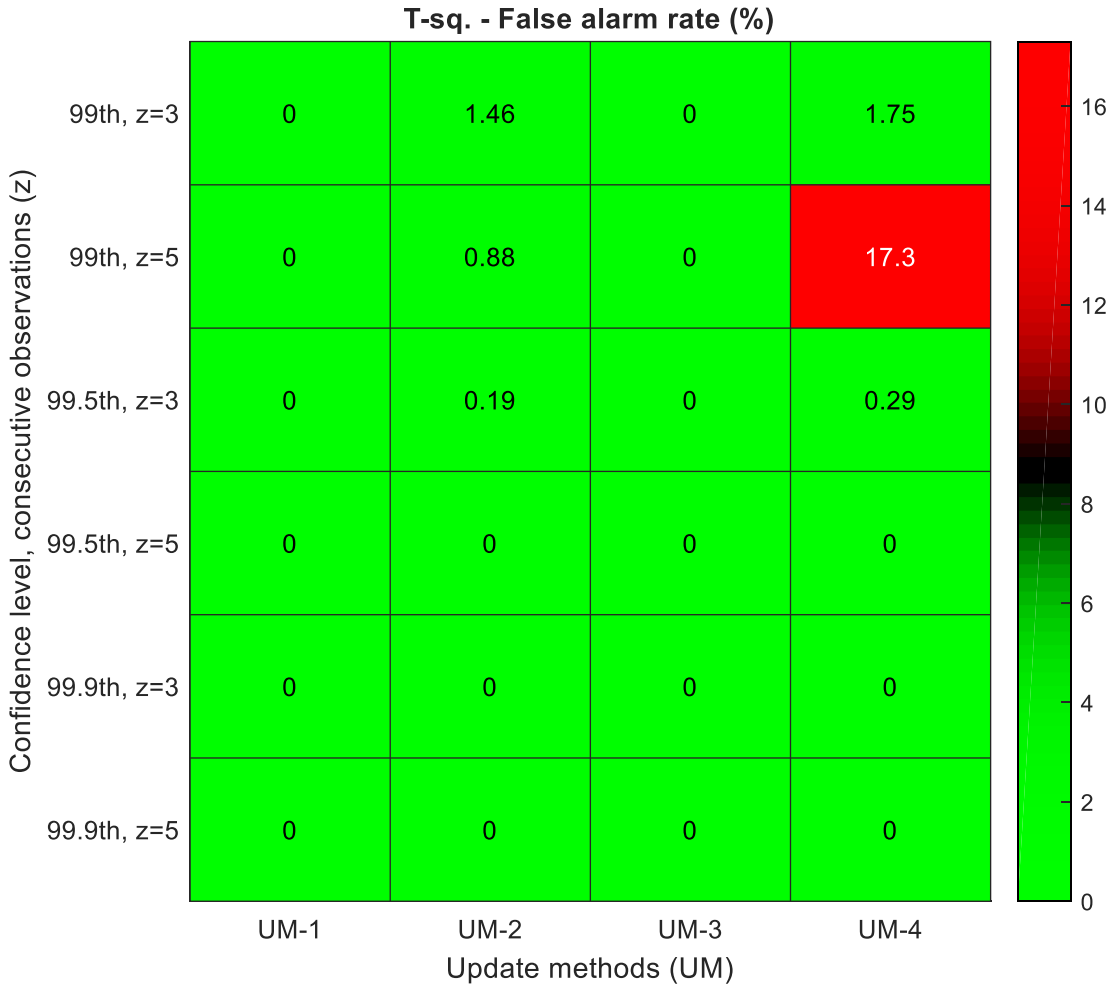


Figure F.30: T^2 statistic FAR for specified update methods.

Overall performance for UM-2 shows low alarm rates for all situations excluding the 99.9th percentile threshold values. UM-3 also seem to perform well for 99.9th percentile threshold overall. UM-4 provides great results across all the thresholds with a considerable amount of alarm rates in two instances of high threshold (99.9th percentile) with lower z value, and low threshold (99th percentile) with high z value.

F.6 Effect of fixed number of PCs and fixed amount variance

Another important factor applicable to all update methods is the general improvement of maintaining a fixed amount of variance as opposed to a fixed number of PCs. As the model updates with time, a fixed number of PCs is very unlikely to account for the same amount of the desired variance. Either the amount of retained variance increases or reduces which impacts the monitoring performance. Results for the SPE statistic of UM-2 and UM-4 respectively shown in Figure F.31 and Figure F.32 show degraded performance (compared to Figure F.15 and Figure F.17) for retaining a fixed number of PCs at each instance. The results are

comparable with the ones analyzed in the previous section which shows a better performance for maintaining a fixed amount of variance at each instance (the amount of variance is predetermined by the number of PCs retained from the training stage). The T^2 statistic for the two methods (shown in Figure F.33 and Figure F.34) in this case is slightly affected by the condition.

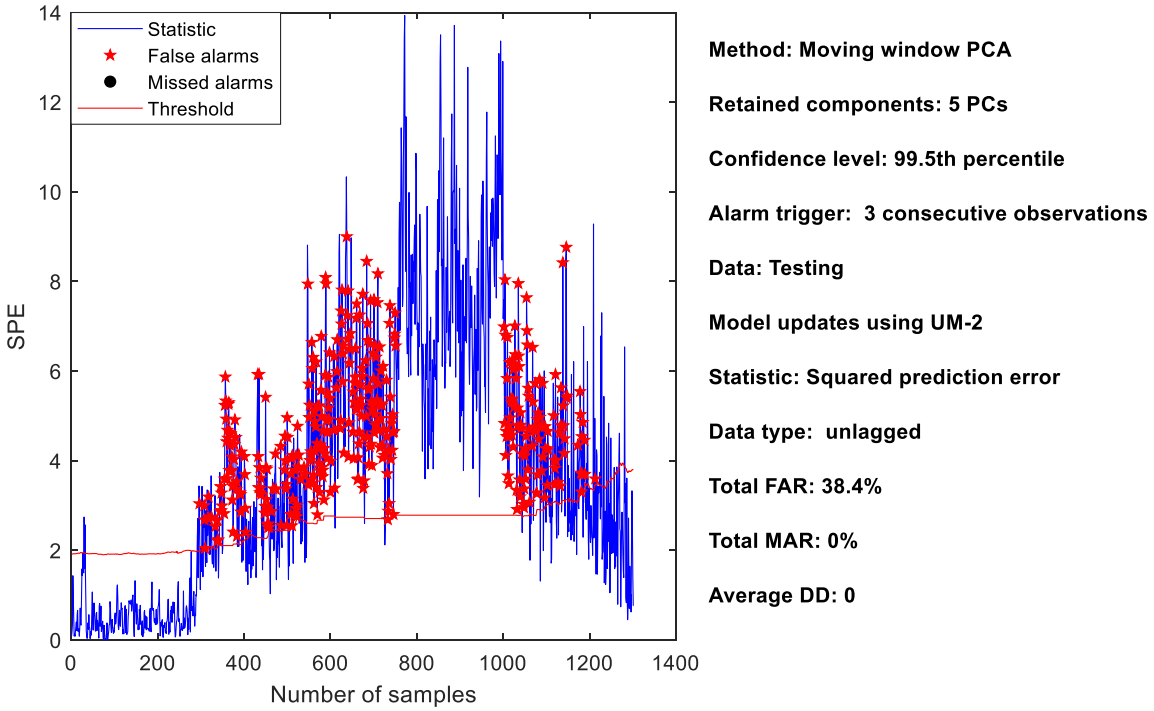


Figure F.31: SPE statistic for the fixed number of retained PCs for UM-2 for Fault C05.

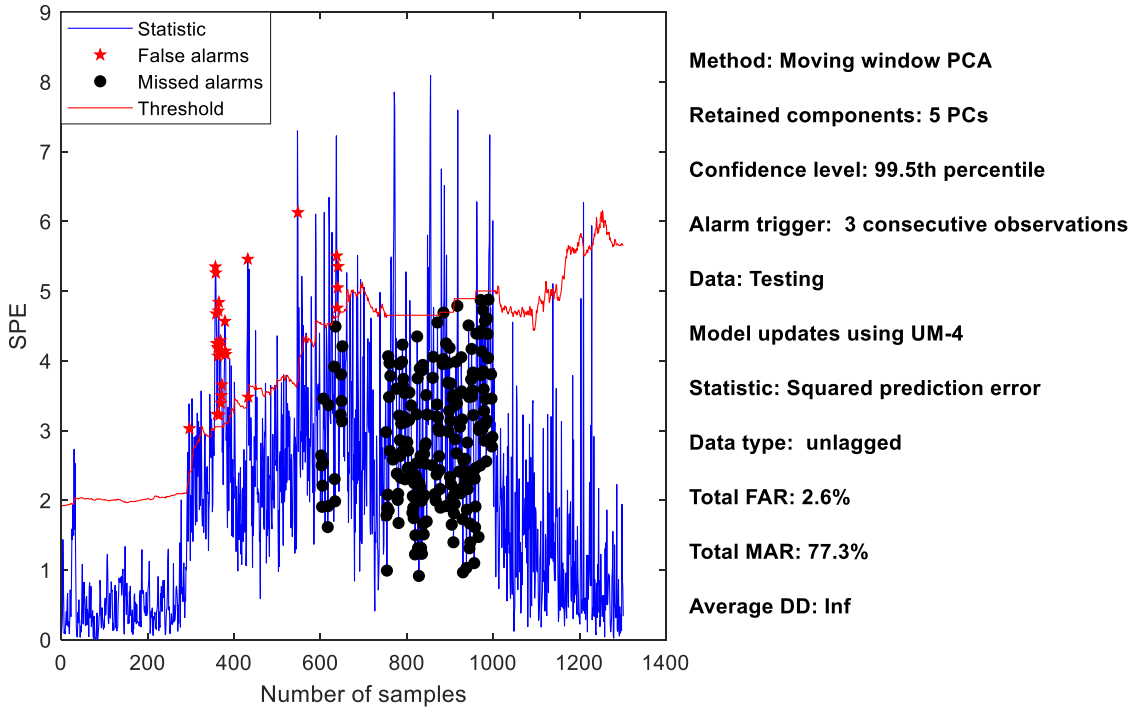


Figure F.32: SPE statistic for the fixed number of retained PCs for UM-4 for Fault C05.

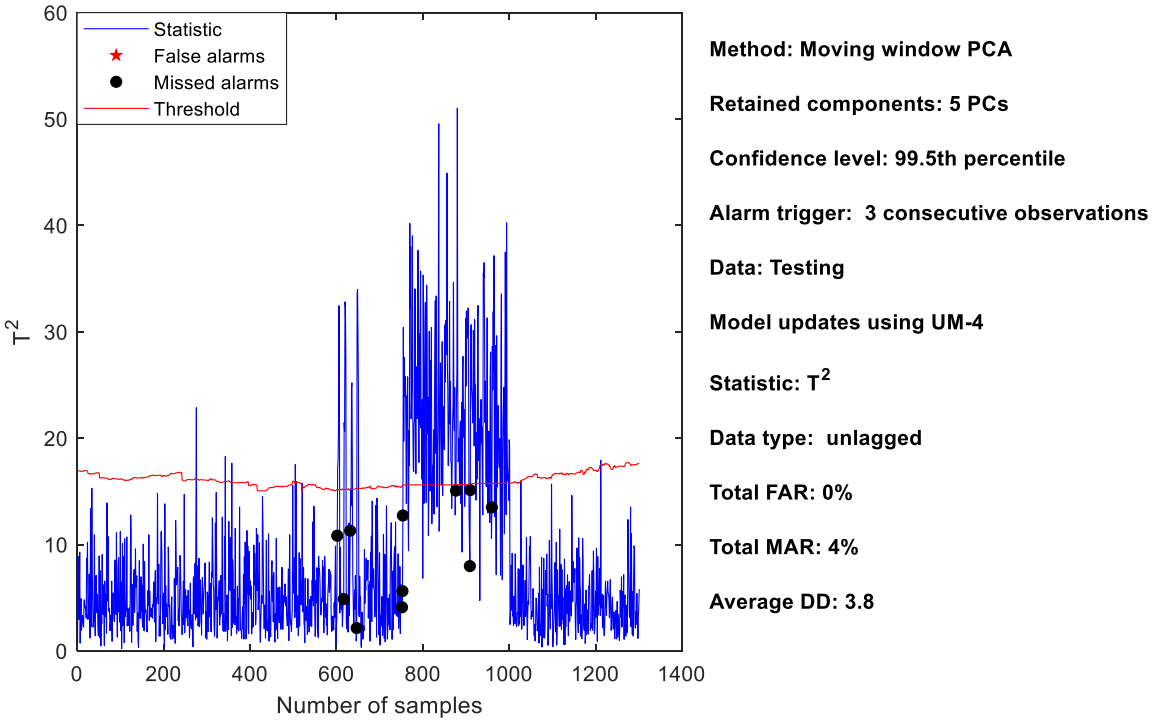


Figure F.33: T^2 statistic for the fixed number of retained PCs for UM-4 for Fault C05.

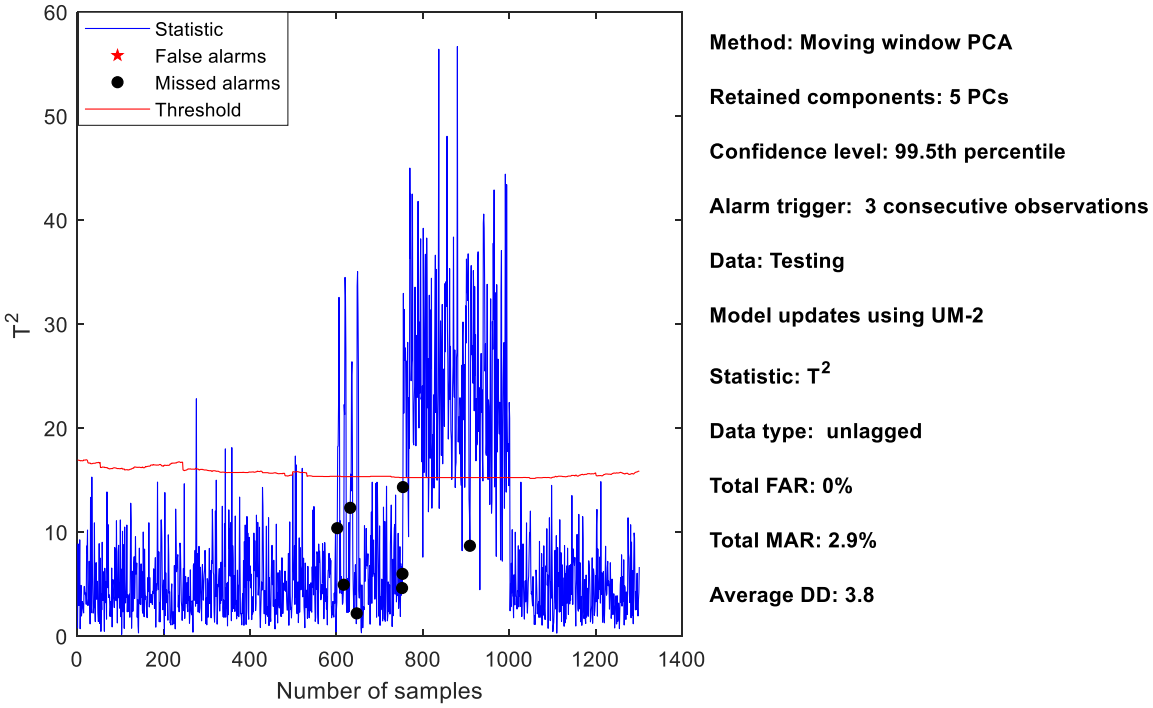


Figure F.34: T^2 statistic for the fixed number of retained PCs for UM-2 for Fault C05.

F.7 Pseudo-update and non-pseudo-update of models

Application of the proposed update method requires the computation of the pseudo-updated model parameters to achieve maximum performance. This is however not ascertained for the other update methods (because pseudo-updating is not involved). Figure F.35 and Figure F.36 respectively show the worsened performance for detecting Faults C05 and Fault C03 (more for Faults C05) by implementing the proposed technique without computing the pseudo-updated model parameters. Figure F.37, on the other hand, shows a much worse performance for coupling the effect fixed number of PCs and not computing the pseudo-updated model parameters.

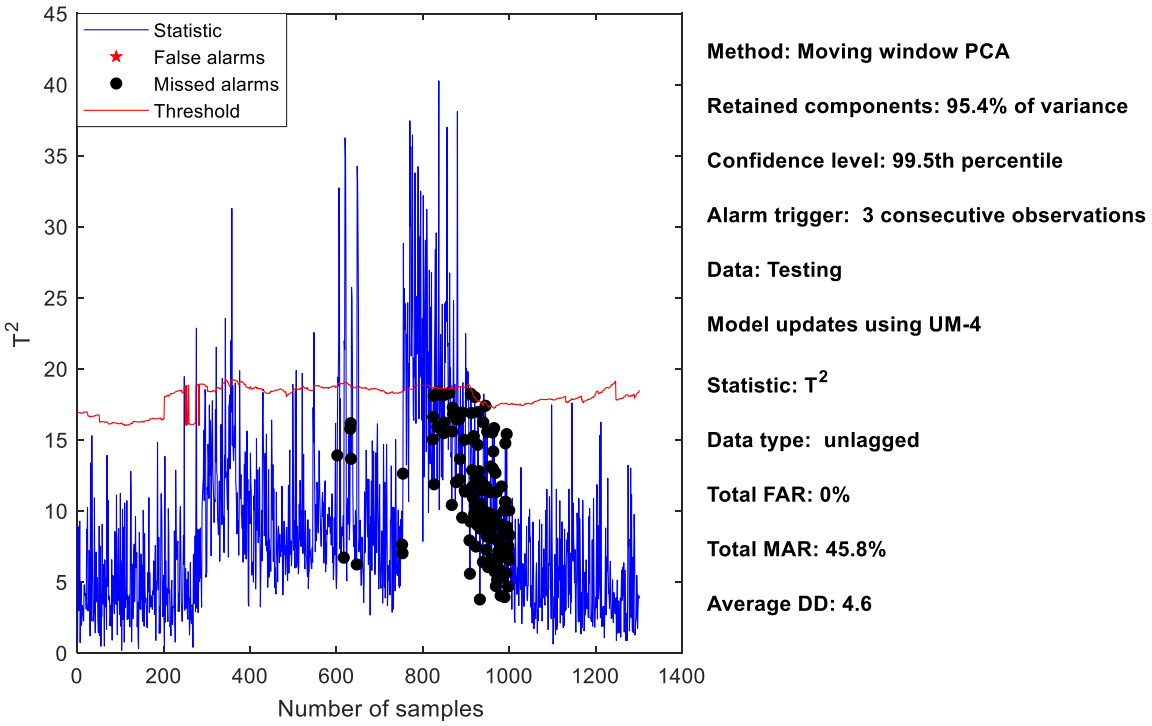


Figure F.35: T^2 statistic for UM-4 using the unperturbed model for Fault C05.

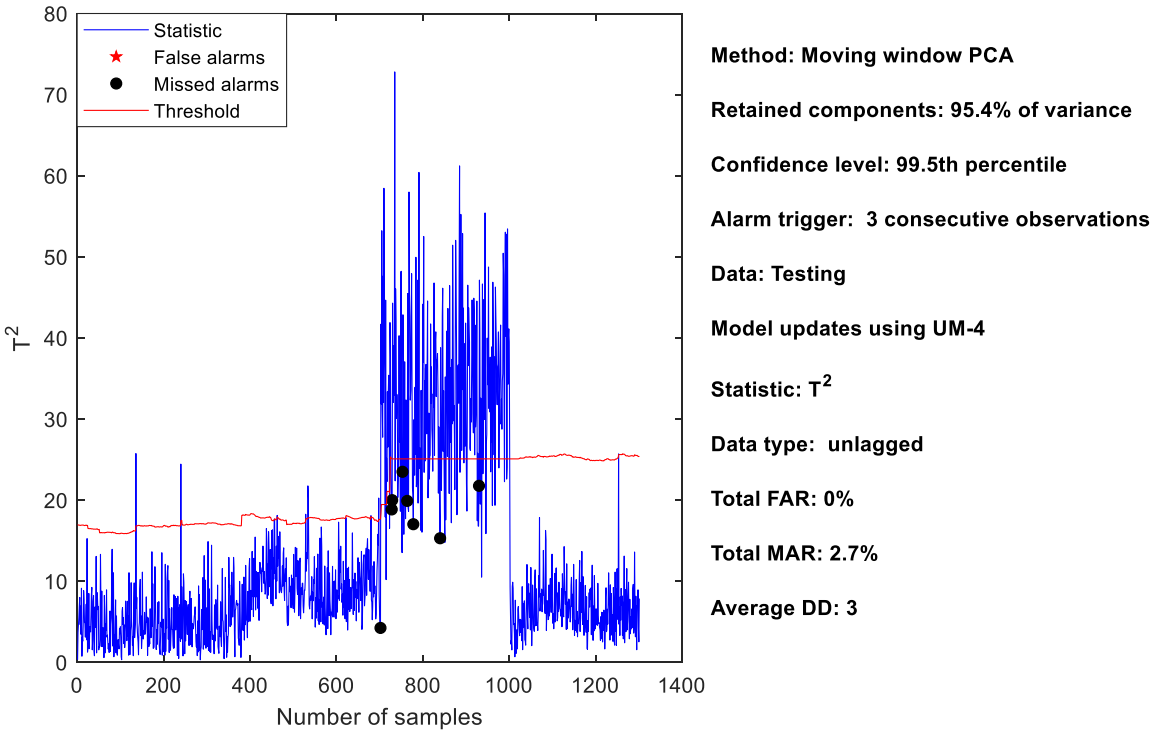


Figure F.36: T^2 statistic for UM-4 using the unperturbed model for Fault C03.

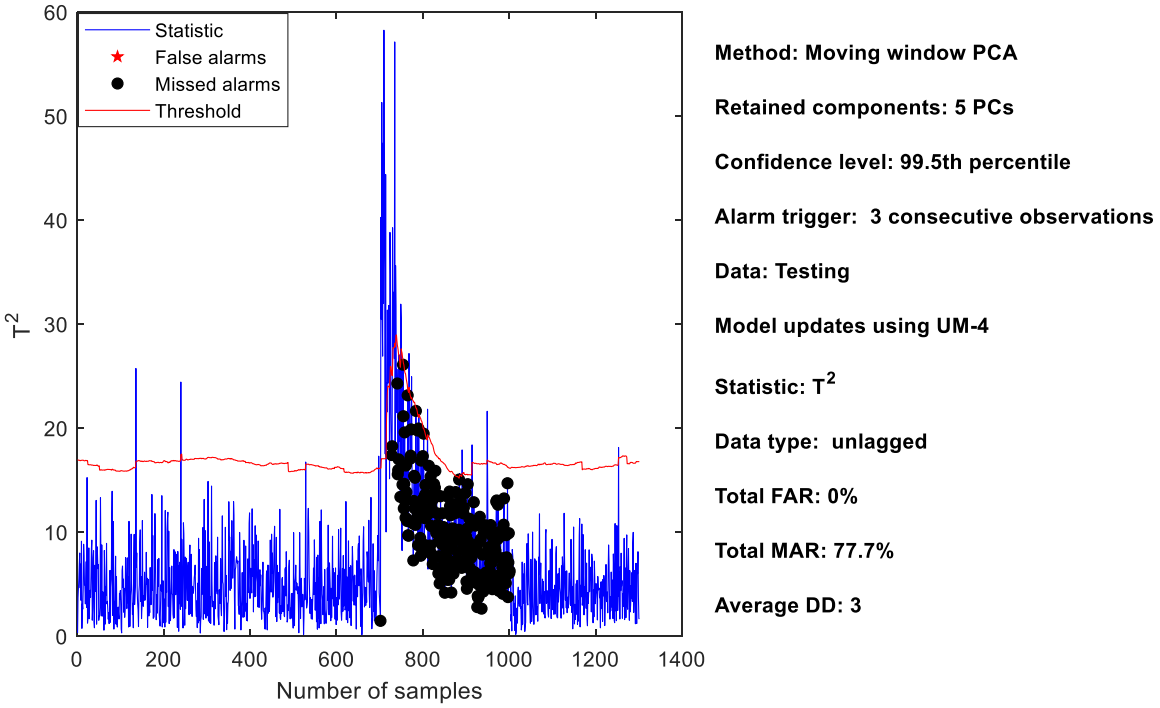


Figure F.37: T^2 statistic for UM-4 using the unperturbed model coupled with a fixed number of PCs for Fault C03.

This confirms the requirement of the pseudo-model update when using the update method four (UM-4). All other update methods do not rely on the pseudo model update.

APPENDIX G: MULTIMODAL PROCESSES

Appendix G provides extra information on the development of GMM and the implications of the various monitoring approaches as well as model selection criteria on fault detection.

G.1 Comparison of unimodal and multimodal approaches

G.1.1 CSTR process Fault C07

For the multimodal process, a 4-mode CSTR process data is generated. Training data Ct01 is generated by setting the concentration controller set point to 120%, 150% and 200% of the original value to produce modes 2, 3 and 4, respectively. A total of 2001 samples are sampled at each minute interval. Mode changes occurred at times 502 and 1002 and 1502.

The test data C07 is generated by introducing the same mode changes like that for the C01 but with the mode changes occurring at times 1002, 2002, and 3002. The total number of samples generated is 4001. Four faults are simulated for C07: a 3% sensor bias in coolant temperature Tc from time 502 to 1002, a 3% sensor bias in reactor inlet temperature Ti from time 1502 to 2002, step of 1.5% magnitude in the reactor inlet temperature Ti from time 2502 to 3002, and a 50% step in concentration of reactant a Ca from time 3502 to 4001. Data Cv01 is validation data with 1201 samples with mode changes occurring at times 302, 602 and 902 respectively for modes 1, 2 and 3. Figure G.1 shows the simulation results for C07 which shows the stated faults with a response from manipulated variables Fc and Fa. Table G.1 shows a summary of the simulated data.

Table G.1: Summary of the simulated CSTR data.

Data	Description	Samples
Ct01	Training data	2001
Cv01	Validation data for Ct01	1201
C07	Tc sensor bias, Ti sensor bias, 1.5 % step in Ti, 50% step in Ca	4001

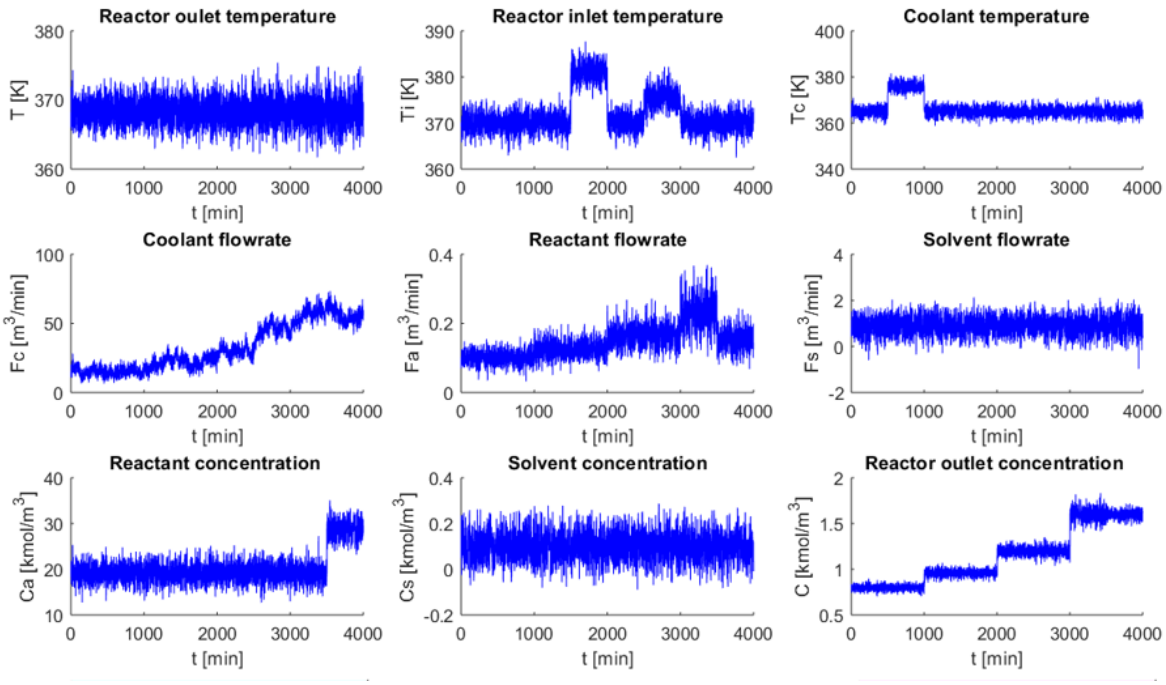


Figure G.1: Simulation results for Tc and Ti sensor bias followed by steps in Ti and Ca with the response from manipulated variables Fc and Fa.

G.1.2 Fault detection for Fault C07

Results for NLPDF statistics (in Figure G.2) and SPE statistic (in Figure G.3) provides a better performance by GMM at an elevated number of retained PCs.

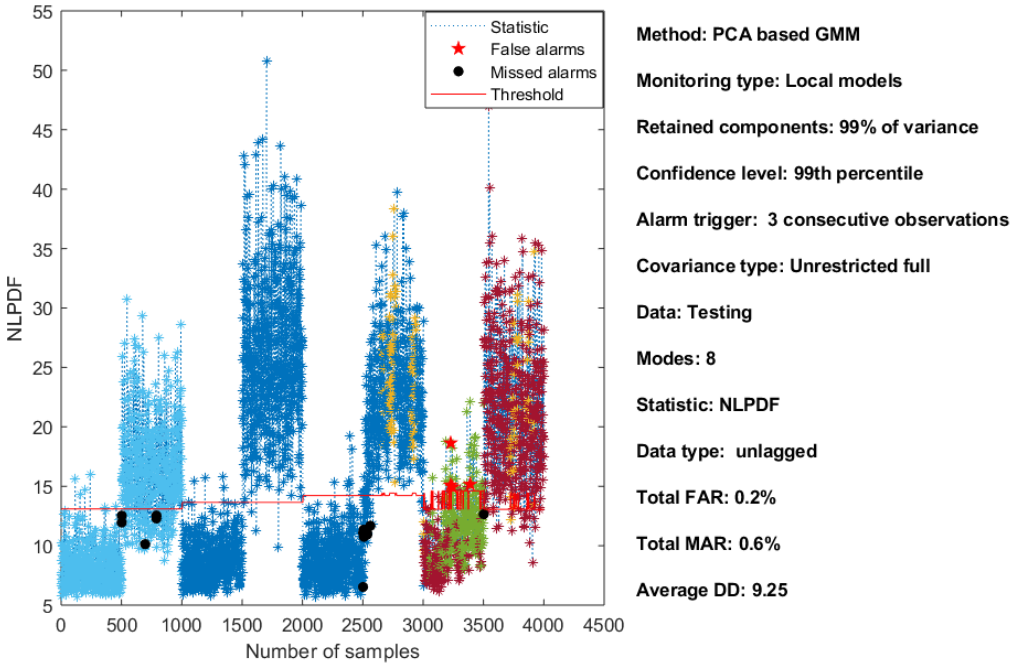


Figure G.2: NLPDF statistics for Fault C07 of GMM.

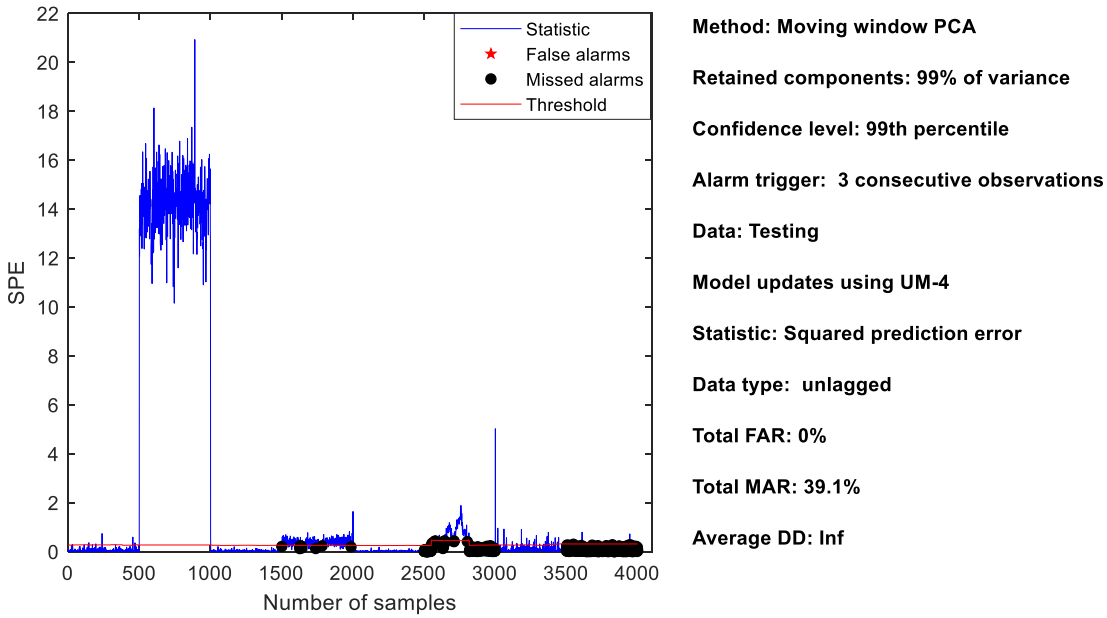


Figure G.3: SPE statistic for Fault C07.

G.2 Model development for GMM

The selected covariance types and number of clusters are assessed for the various model selection criteria by considering unimodal and multimodal process data.

G.2.1.1 TE process

The training data with 500 observations and a unimodal process. The maximum number of clusters to be considered in this case is six (according to Equation **3-3**). Figure G.4 provides the PCA score plots for the data under consideration which shows a unimodal process.

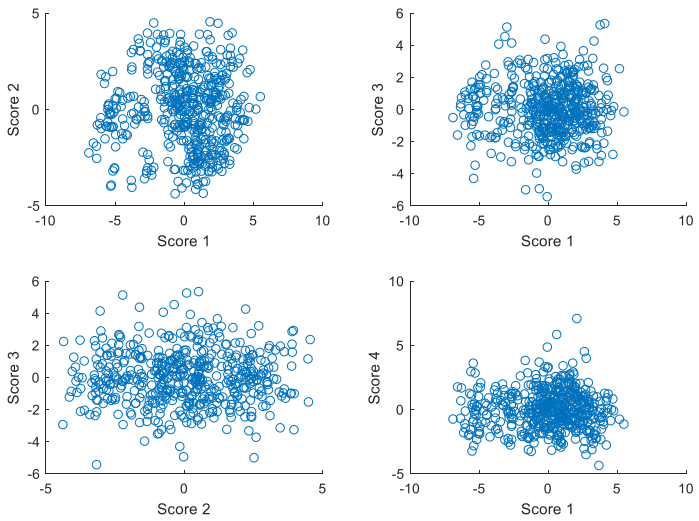


Figure G.4: Score plots for TE training data.

The results for the free run (shown in Section 5.3.2.1) show that the use of PCA scores enables a rather parsimonious diagonal covariance to be fitted to the data. Also, the two instances in which the AIC fitted the diagonal-unshared covariance are the instances in which the selected number of clusters was two (i.e. instance 3 and 23).

Since the diagonal-shared covariance was selected by the PCA-scores in most instances, the effect forcing only full covariance type on the PCA scores is analyzed next. Same is done for the normalized and raw data but with forced diagonal covariance

Figure G.5 shows the number of selected clusters in each instance. While the BIC consistently select a single cluster for the PCA scores, the AIC averaged four clusters. The results for the raw and normaized data almost always picked the highest possible cluster number in each instance. This proves the inability of the diagonal matrix to achieve good performance for the raw and normalized data. The cluster number selected increases with an increase in the maximum number of considered clusters which is six in this case.

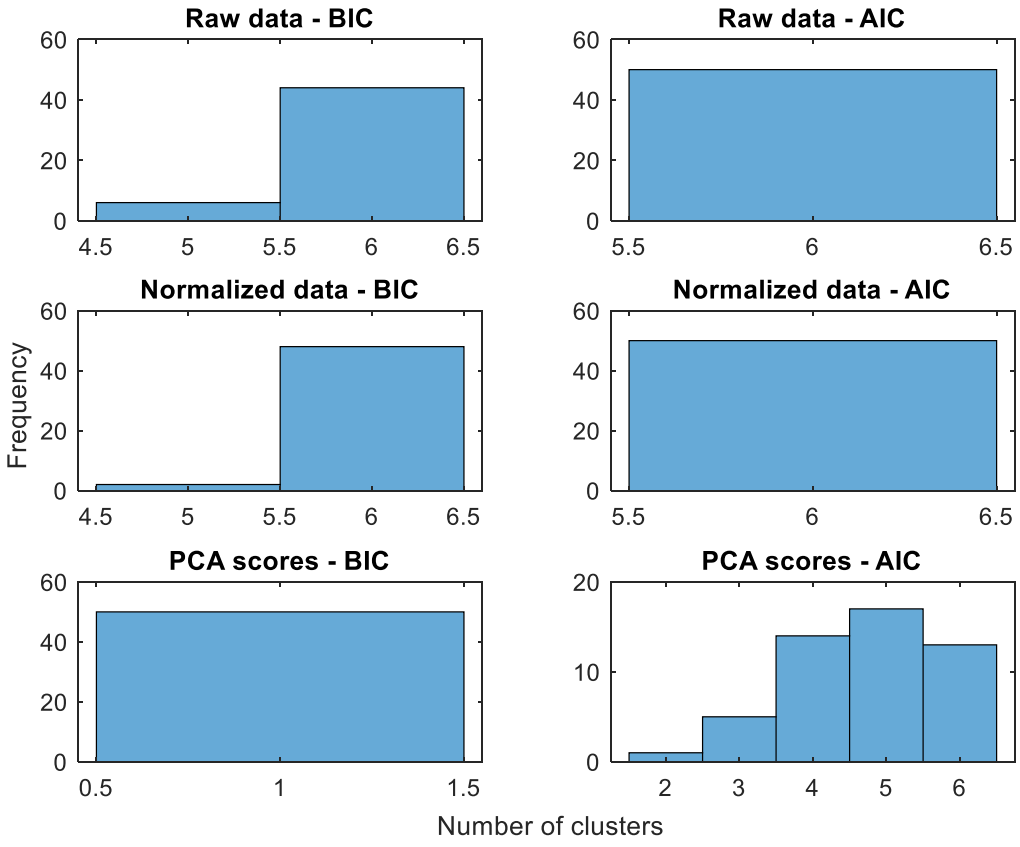


Figure G.5: Forced selected number of clusters for various data types using TE training data.

For the selected covariance shapes under the forced clustering (shown in Figure G.6), both the AIC and BIC picked the full-shared covariance in every instance for the PCA scores. The BIC

selected the shared covariance for all cases involving the normalized and raw data. The AIC, on the other hand, selects an almost equal amount of both shared and unshared for the normalized and raw data. Note that the only option available for the PCA scores is the full covariance and for the normalized and raw, the diagonal.

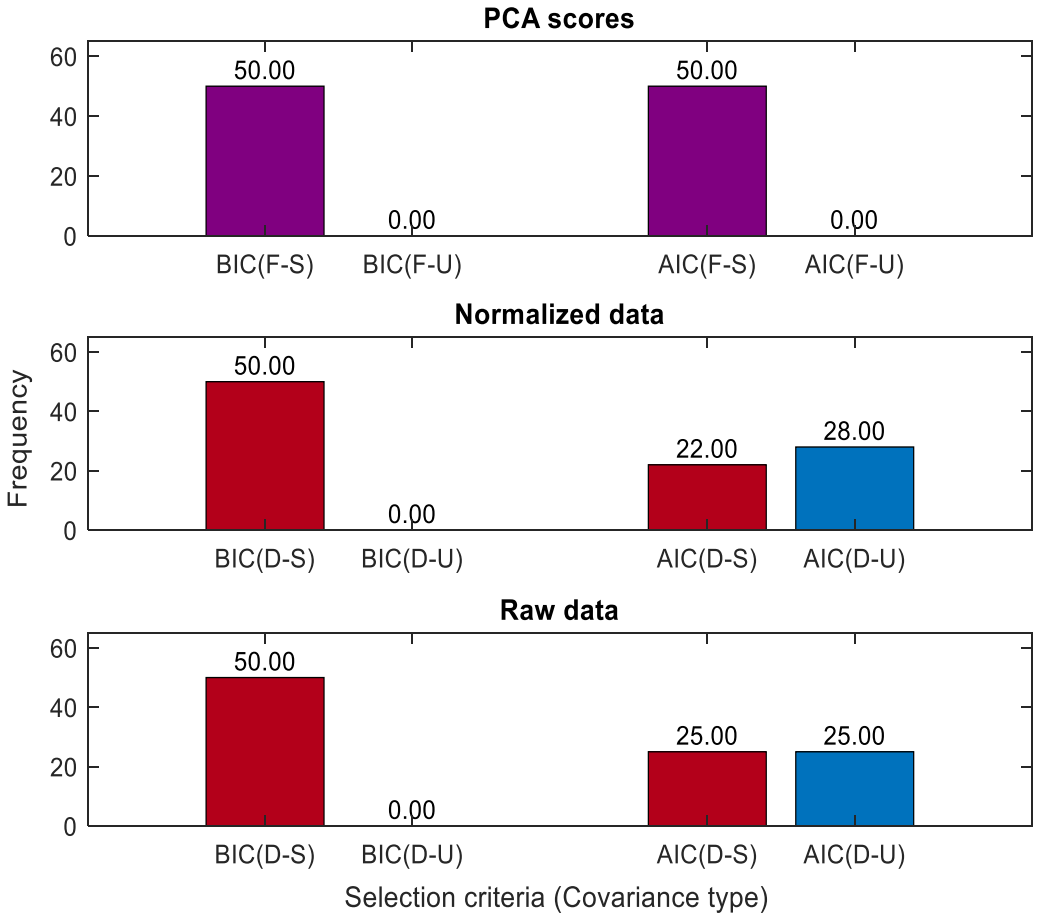


Figure G.6: Frequency of covariance types: diagonal-shared (D-S), diagonal-unshared (D-U), full-shared (F-S), full-unshared (F-U) for various data types using TE training data.

Finally, the effect of forcing an unshared diagonal covariance since the BIC consistently selects a shared full one for the scores (which is the best performing one). Figure G.7 shows the results for the number of selected clusters when a forced unshared diagonal covariance was used. The results for PCA scores remained unchanged for the BIC from the previous analysis. The AIC almost always selects the highest available option for the cluster numbers in all normalized and raw data cases. The BIC, however, averaged five number of clusters for the normalized data while the raw data alternates between four and five cluster numbers.

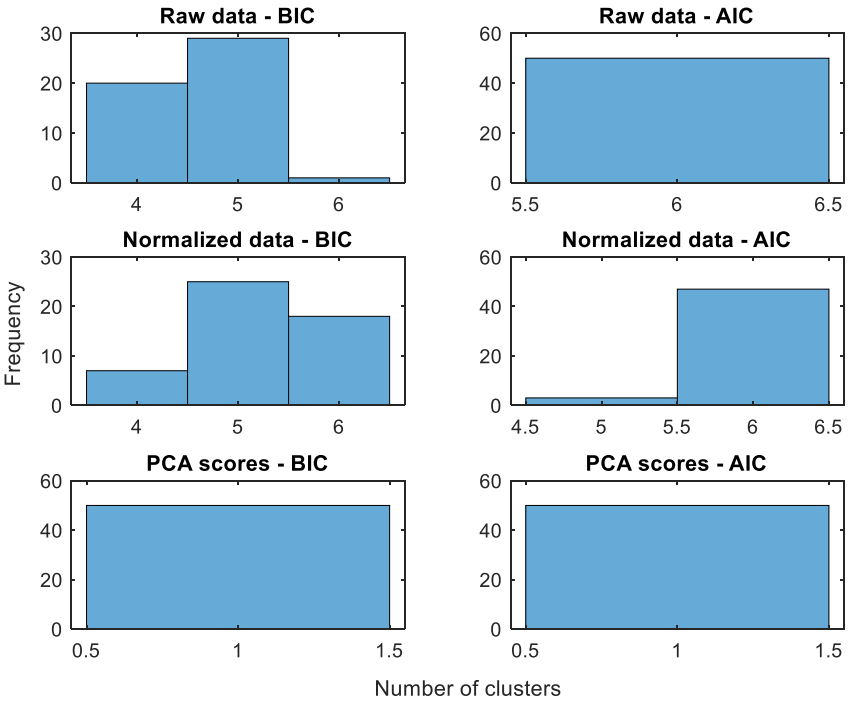


Figure G.7: Forced selected number of clusters for various data types for TE training data.

G.2.1.2 Unimodal CSTR process

The training data is made up of 1001 observations and a unimodal process. The maximum number of clusters to be considered in this case is 8 (according to Equation **3-3**). Figure G.8 provides the score plots for the data under consideration which shows a unimodal process.

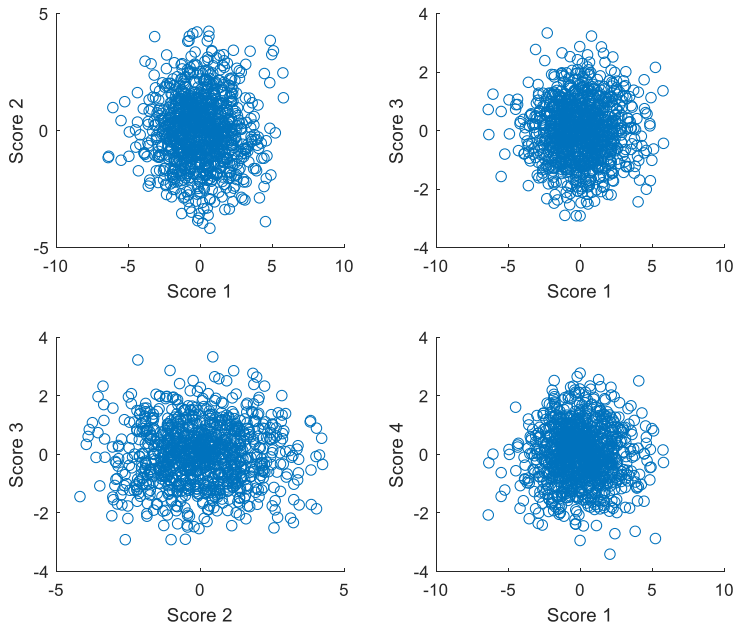


Figure G.8: Score plots for CSTR training data Ct00.

Figure G.9 shows the results for a selected number of clusters for the AIC and BIC at each instance for a free run.

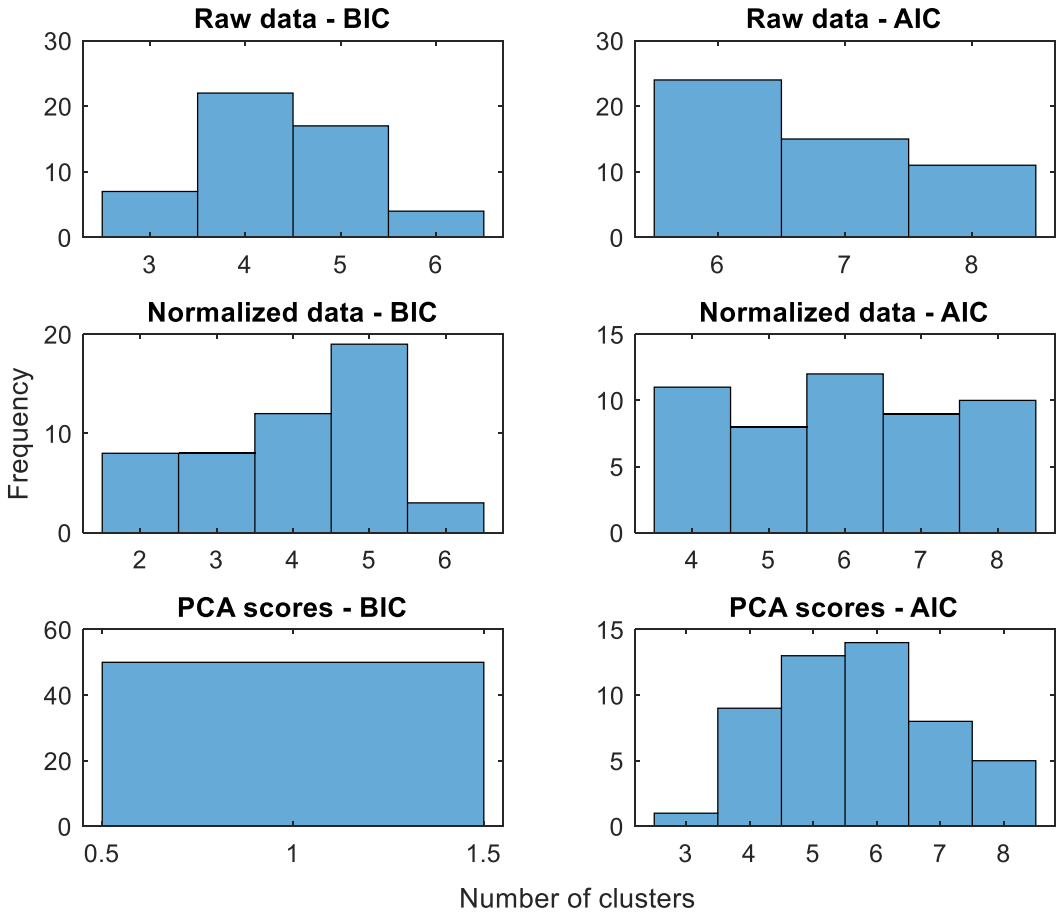


Figure G.9: Selected number of clusters for various data types for CSTR training data Ct00.

The results for the selected covariance shapes is presented in Figure G.10 for the unrestricted run.

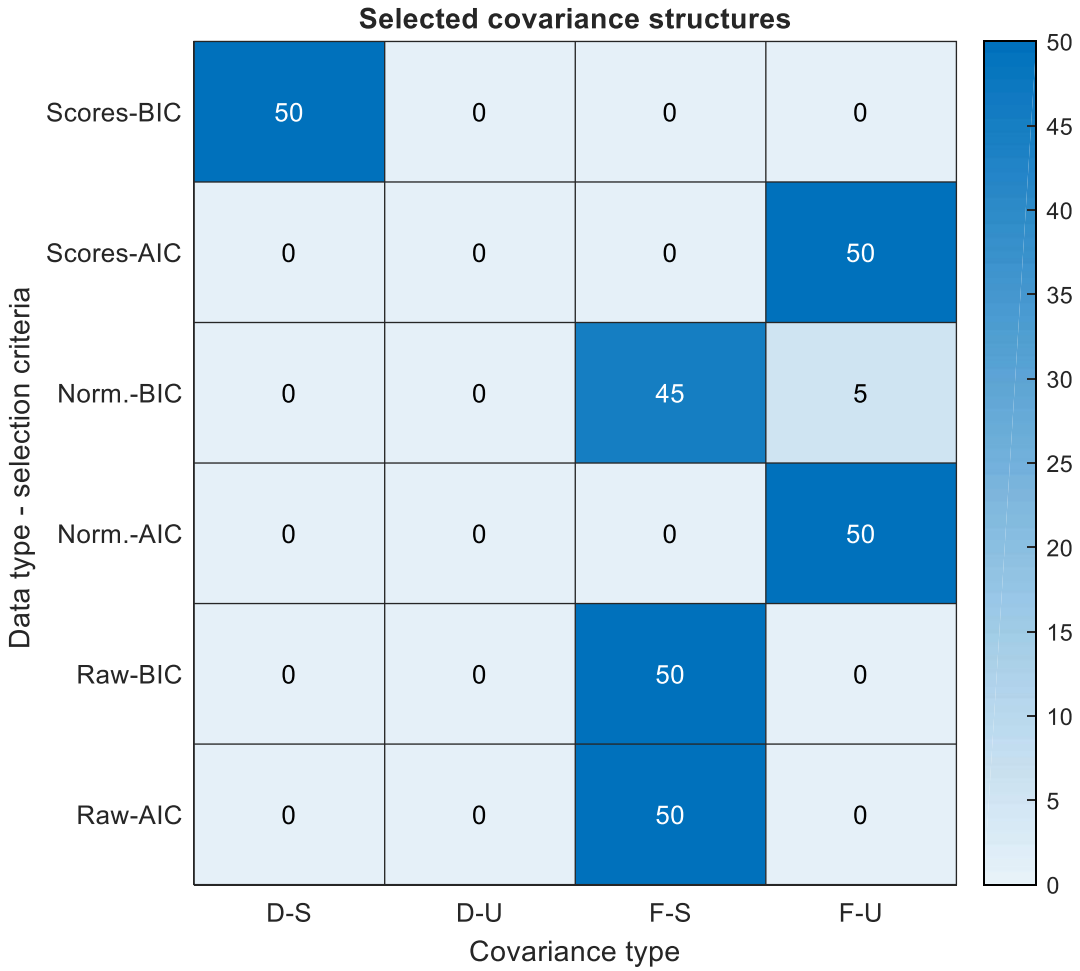


Figure G.10: Selected covariance types: diagonal-shared (D-S), diagonal-unshared (D-U), full-shared (F-S), full-unshared (F-U) for specified data types for CSTR training data Ct00.

The next result then considers forcing a full covariance matrix for the PCA scores to see the performance of the BIC since it is the best performing in that aspect. Figure G.11 shows the results for the covariance types.

Figure G.12 shows the results for the selected covariance shapes the selection of a shared covariance for the BIC in most instances for all data types. The AIC, on the other hand, opts for unshared covariance type.

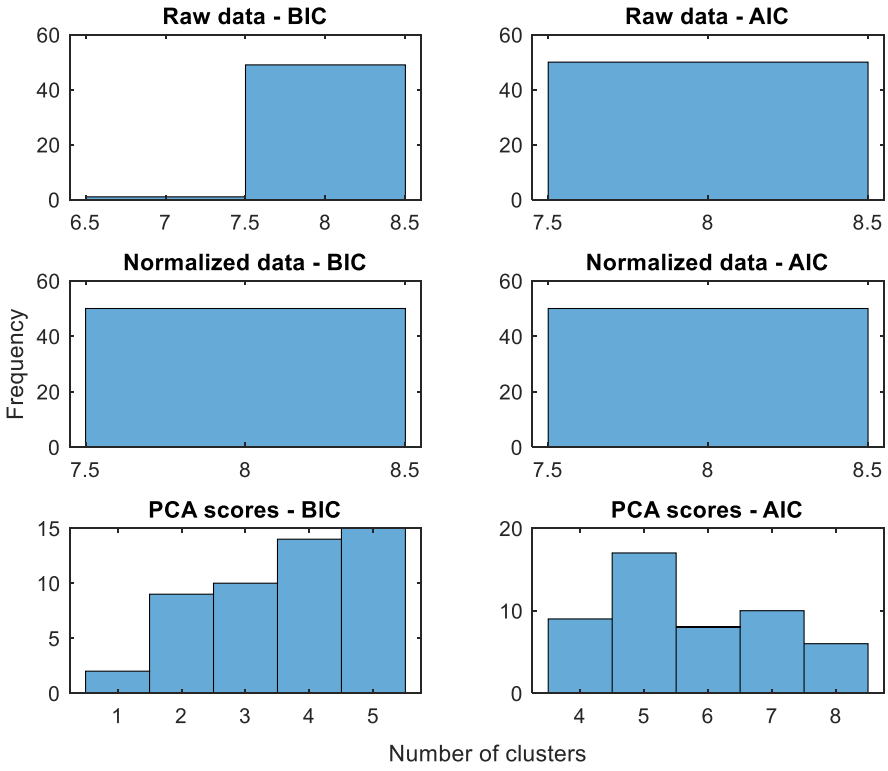


Figure G.11: Forced selected number of clusters for various data types for CSTR training data Ct00.

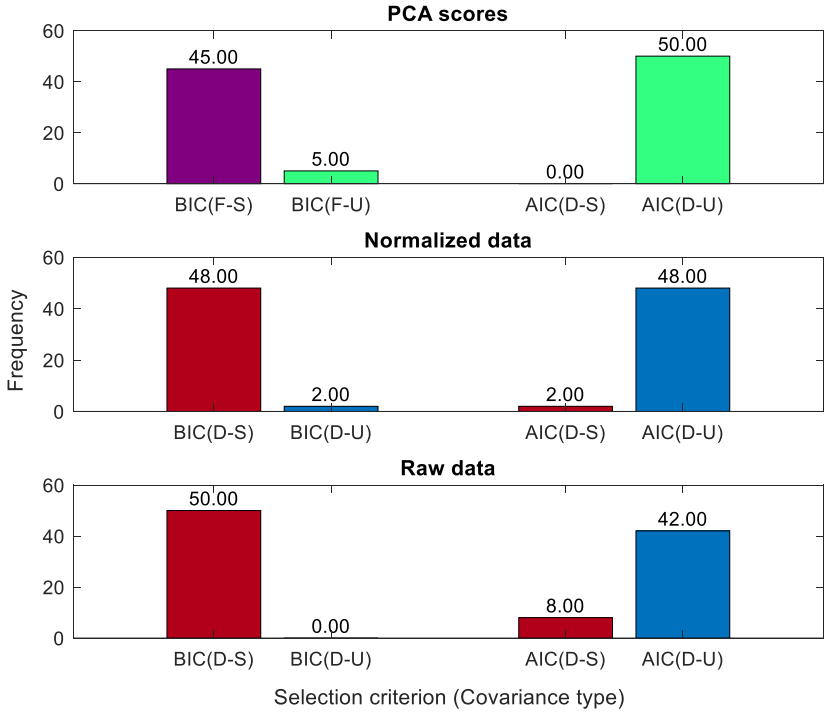


Figure G.12: Frequency of covariance types: diagonal-shared (D-S), diagonal-unshared (D-U), full-shared (F-S), full-unshared (F-U) for various data types using CSTR training data Ct00.

The BIC overall shows the best potential in estimating the cluster number as well as being consistent with the selection of shared covariance in most cases. This is expected of BIC as it penalizes more model complexity. The next section, therefore, considers a multimodal process.

G.2.2 Multimodal CSTR process

For the multimodal process, a 3-mode CSTR process data C02 is generated by setting the concentration set point to 150% and 200% of the original value to produce modes 2 and 3 respectively. A total of 1501 samples is generated and sampled at each minute interval. Mode changes occurred at times 502 and 1002. Figure G.13 shows the simulation results for C02.

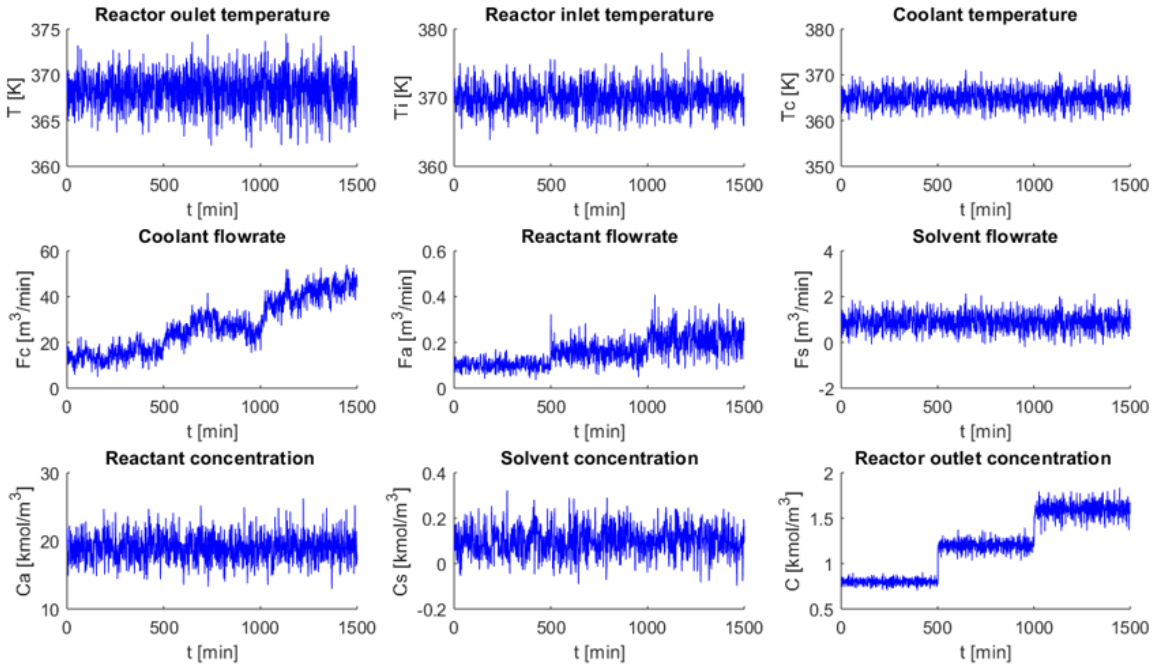


Figure G.13: Simulation results for the 3-mode process achieved by changing the set point of the concentration controller.

The maximum number of clusters to be considered is nine in this case. Figure G.14 provides the score plots for the data under consideration which shows a 3-mode process.

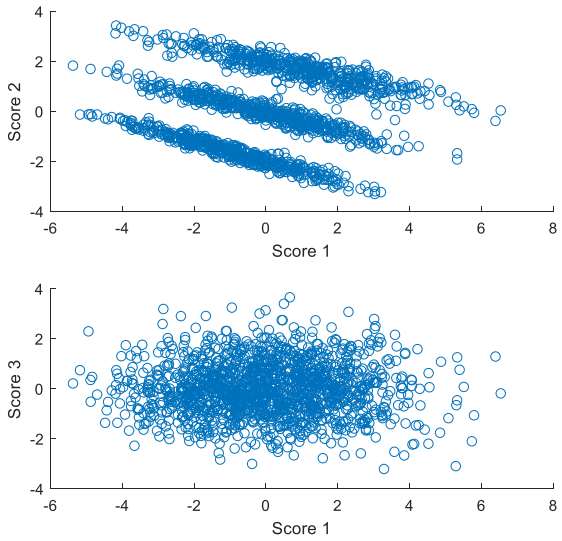


Figure G.14: Score plots for CSTR training data Ct01.

Forcing of a diagonal covariance type on all the data types results in the selection of the highest number of considered clusters in almost every case (a shown in Figure F.22). Also, Diagonal-unshared covariance is consistent with the AIC while the BIC selected few shared covariance types in a few instances.

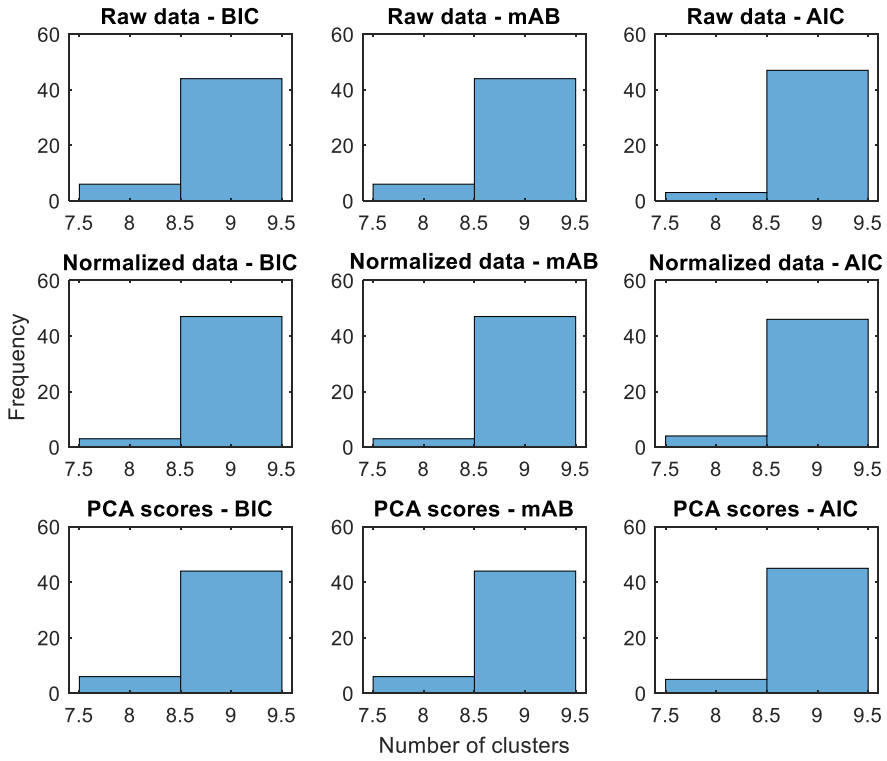


Figure G.15: Selected number of clusters using forced covariance type for various data types for CSTR training data Ct01.

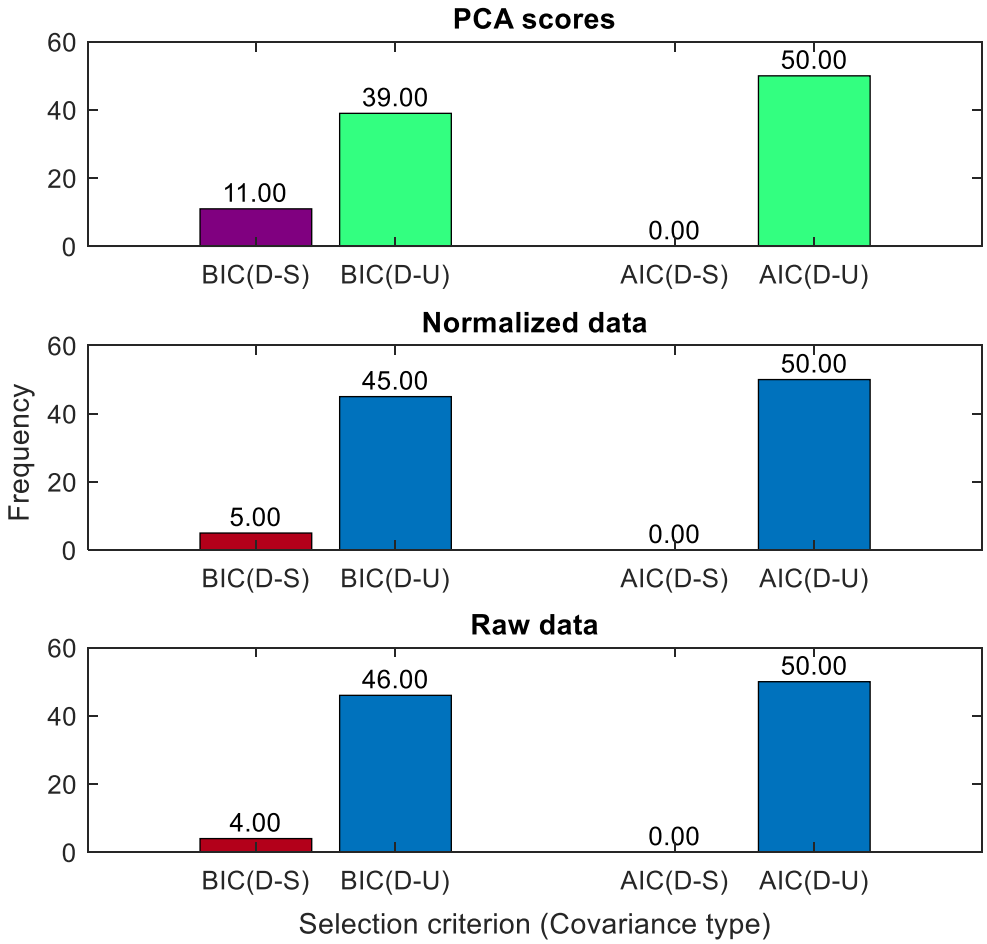


Figure G.16: Selected number of clusters for forced covariance type for various data types for CSTR training data Ct01.

G.3 Implementation of the developed GMM model

This aspect considers the impact of various hyperparameters in fault detection.

G.3.1 TE process

Two TE process faults considered are Fault T04 and Fault T05.

G.3.1.1 Fault detection for Fault T04

Figure G.17 and Figure G.18 respectively show sample monitoring results for normalized data and raw data based GMM. The two approaches achieve the same alarm rates at evaluated parameters. Results for the scores based approach (shown in Figure G.19**)** where 90% of retained PCs show inferior results to that obtained for the other approaches. But the results are similar to that obtained for conventional PCA and APCA. Just like the Fault T05 of the TE process, retaining all the PCs achieve the same performance for the normalized and raw data as shown in Figure G.20.

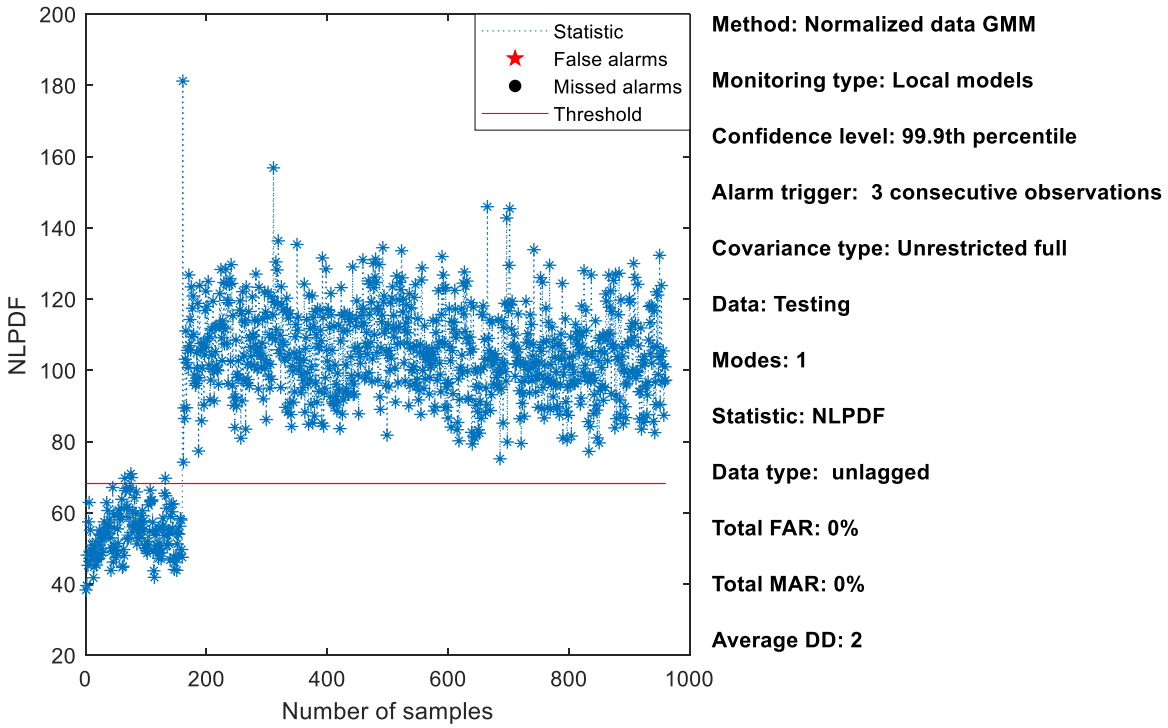


Figure G.17: NLPDF statistics for Fault T04 for normalized data based GMM.

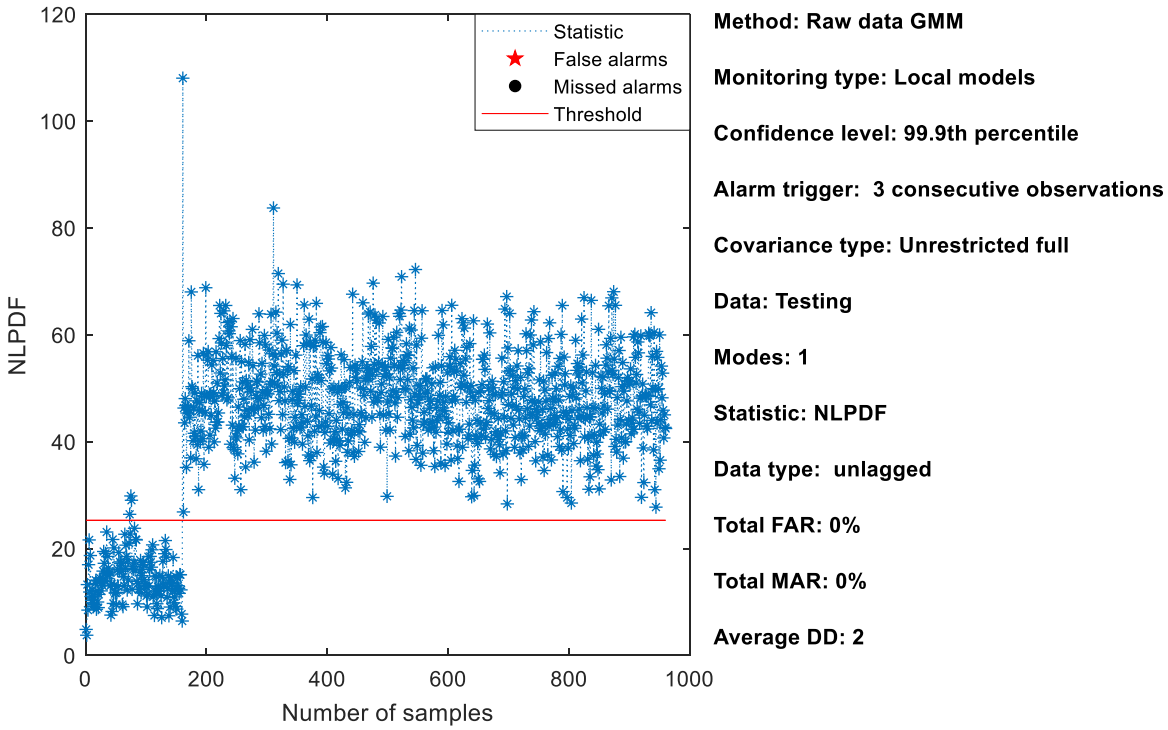


Figure G.18: NLPDF statistics for Fault T04 for raw data based GMM.

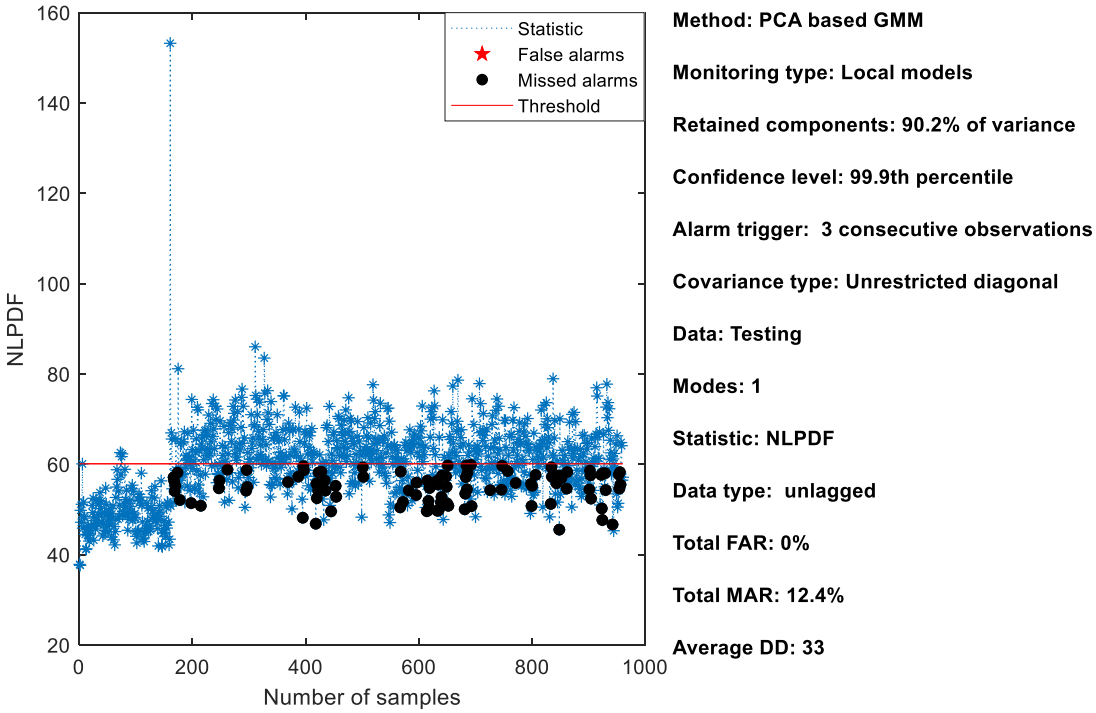


Figure G.19: NLPDF statistics for Fault T04 for PCA-based GMM for 90% of variance.

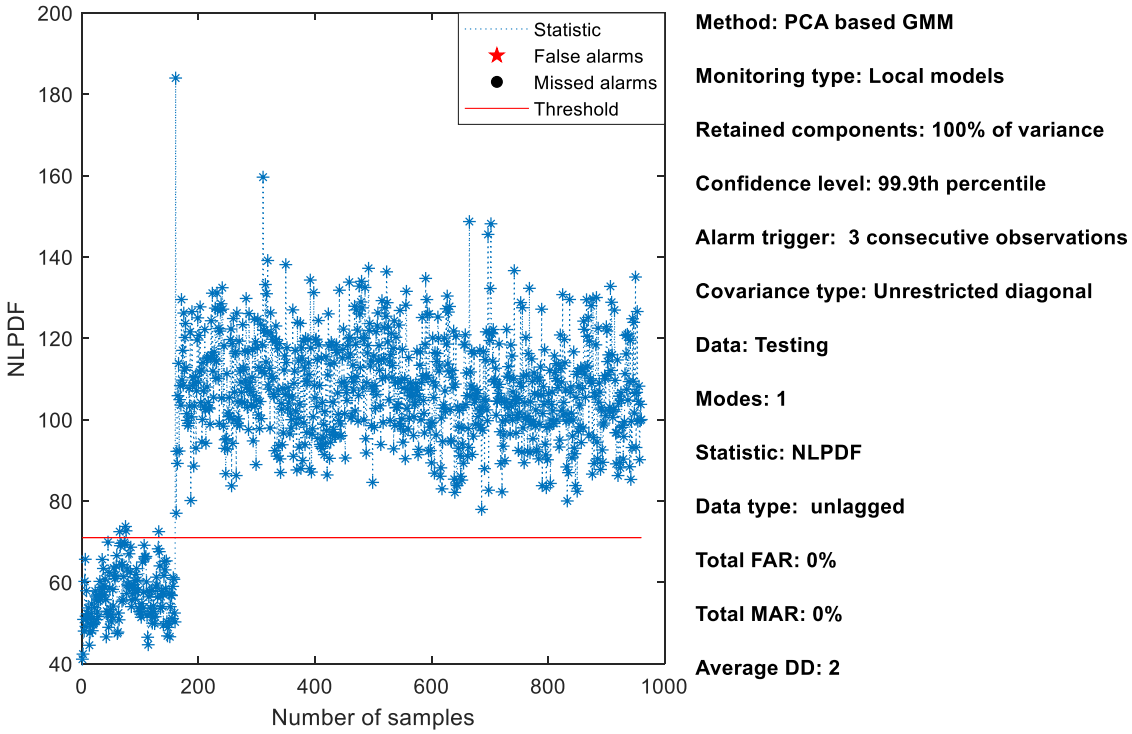


Figure G.20: NLPDF statistics for Fault T04 for PCA-based GMM for 100% retained variance.

Results for the MAR (in Figure G.21) and FAR (in Figure G.22) shows an overall best performance for the raw data based GMM with the diagonal covariance providing the best result. The full covariance types also provided the worst results for the normalized data.

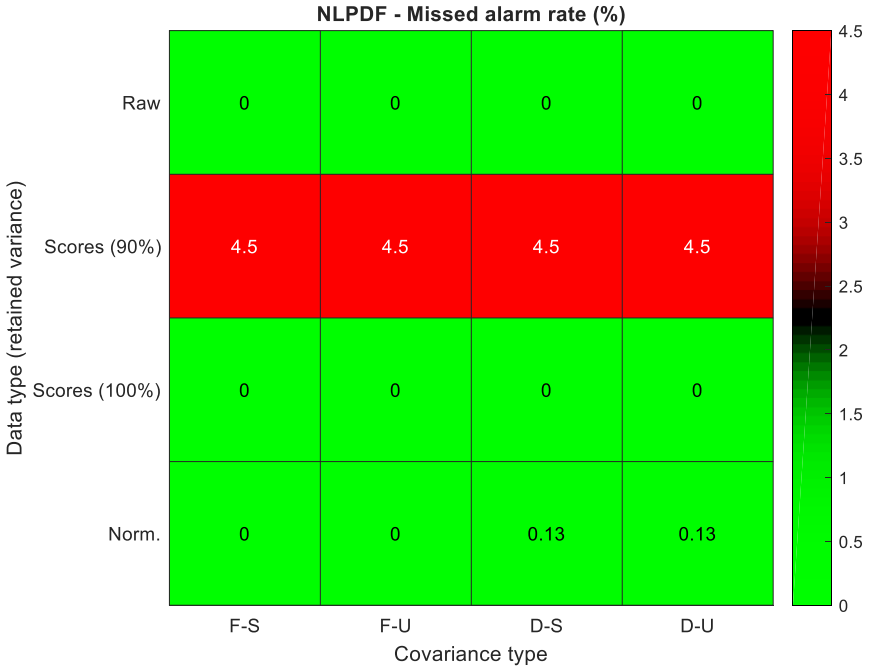


Figure G.21: NLPDF statistics MAR for Fault T04 evaluated at the 99th percentile and z = 3.

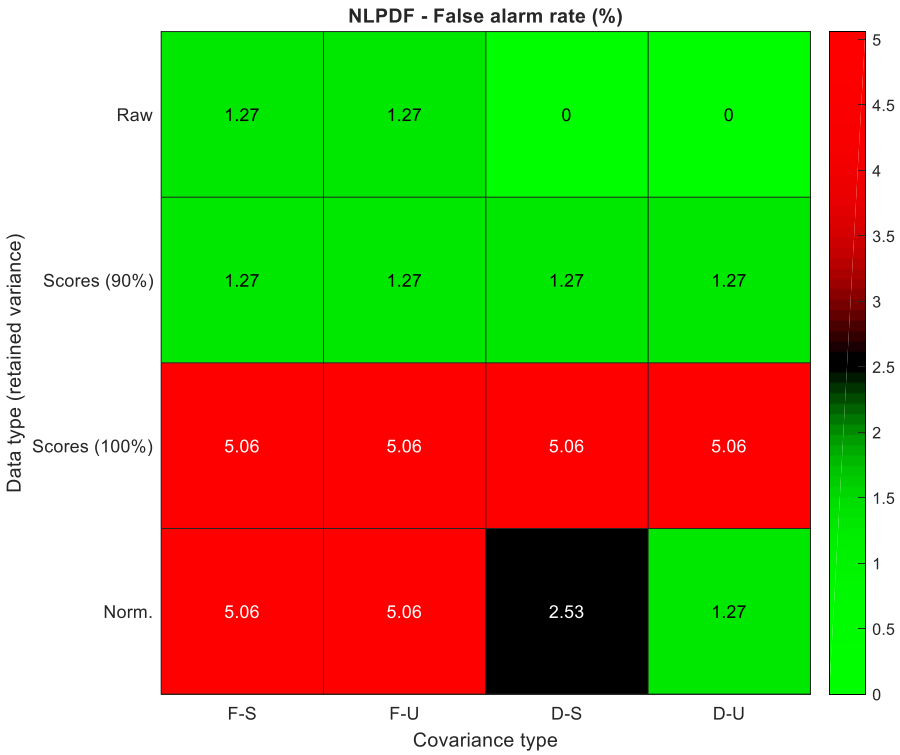


Figure G.22: NLPDF FAR statistic for Fault T04 evaluated at the 99th percentile and z = 3.

G.3.1.2 Fault detection for Fault T05

Sample results for the MAR (in Figure G.23) and FAR (in Figure G.24) show the best performance overall to be achieved by the PCA scores for all retained PCs. This was consistent overall covariance types. The normalized and raw data suffer most under the diagonal covariance types which were established as often unsuitable for such data types.

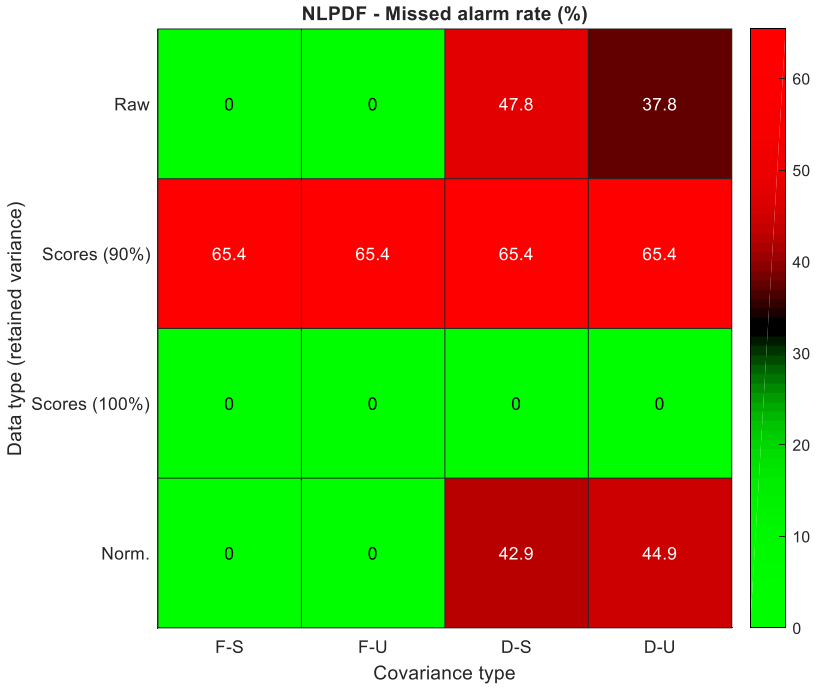


Figure G.23: NLPDF statistics MAR for Fault T05 evaluated at the 99.9th percentile and z = 3.

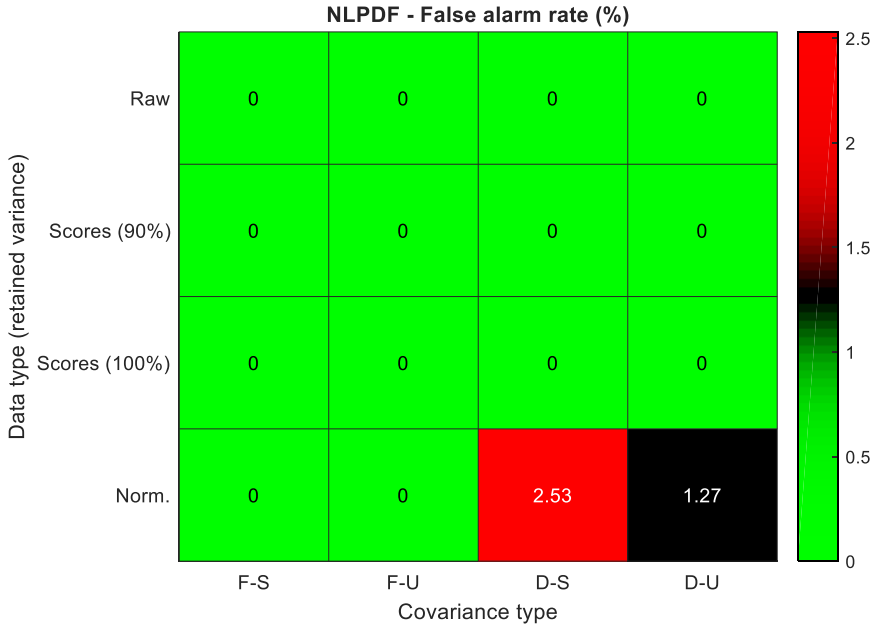


Figure G.24: NLPDF statistics FAR for Fault T05 evaluated at the 99.9th percentile and z = 3.

Overall results for the TE process show that the full covariance matrix type is suitable for all data types in most cases. The PCA scores performance over the covariance types is almost similar while the raw and normalized data suffer much when diagonal covariance which is unsuitable for them is used. Comparison of global and local statistic is not available for the TE process as a single mode is always fitted to the data. This makes the results for the global statistic same as that for the local.

The next aspect considers the CSTR fault data C07 which is a four-mode process as described earlier.

G.3.1.3 Fault detection for Fault C07

Figure G.25 shows sample results for scores based on GMM where 5 PCs were retained. The results show a value of less than 1% for alarm rates at the evaluated parameters for the implementation of local monitoring models. The results for 4 retained PCs for scores (Figure G.26) show a poorer performance than that for the higher number of retained PCs. The results for the 4 retained PCs show better performance when local monitoring model approach is used than for the global approach as shown in Figure G.27. Note that unshared covariance is the same as unrestricted covariance as shared is restricted.

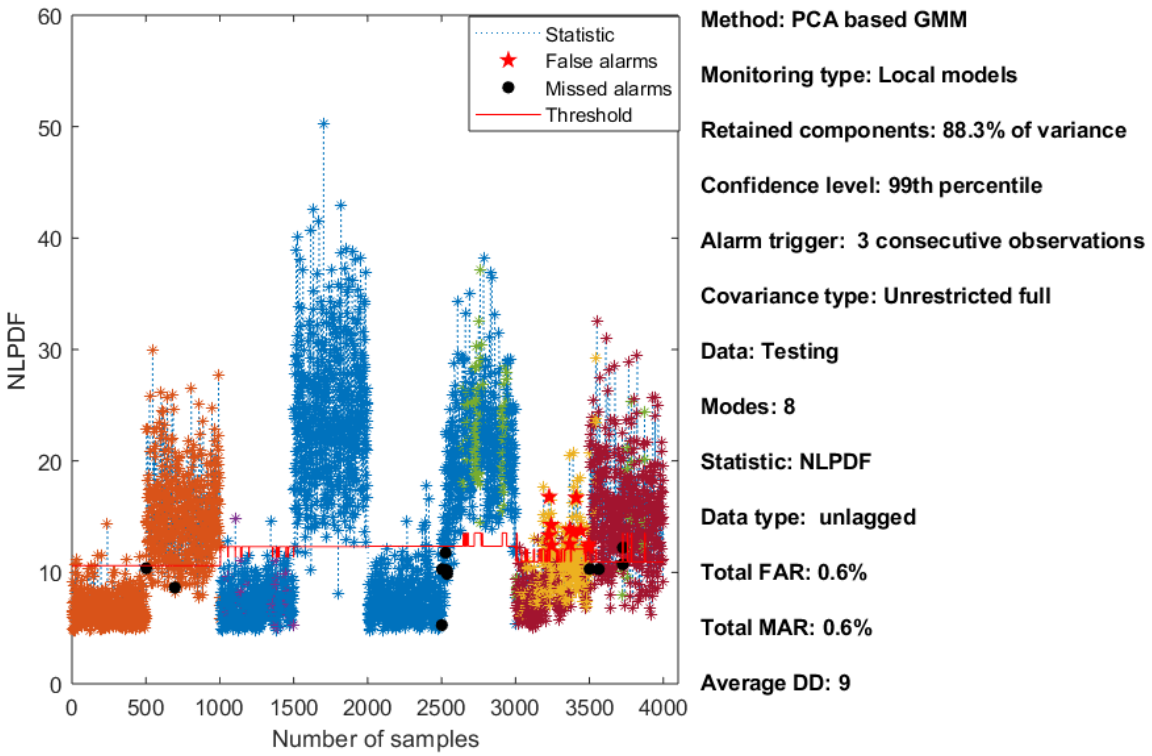


Figure G.25: NLPDF statistics for Fault C07 for PCA-based GMM for 88% of variance.

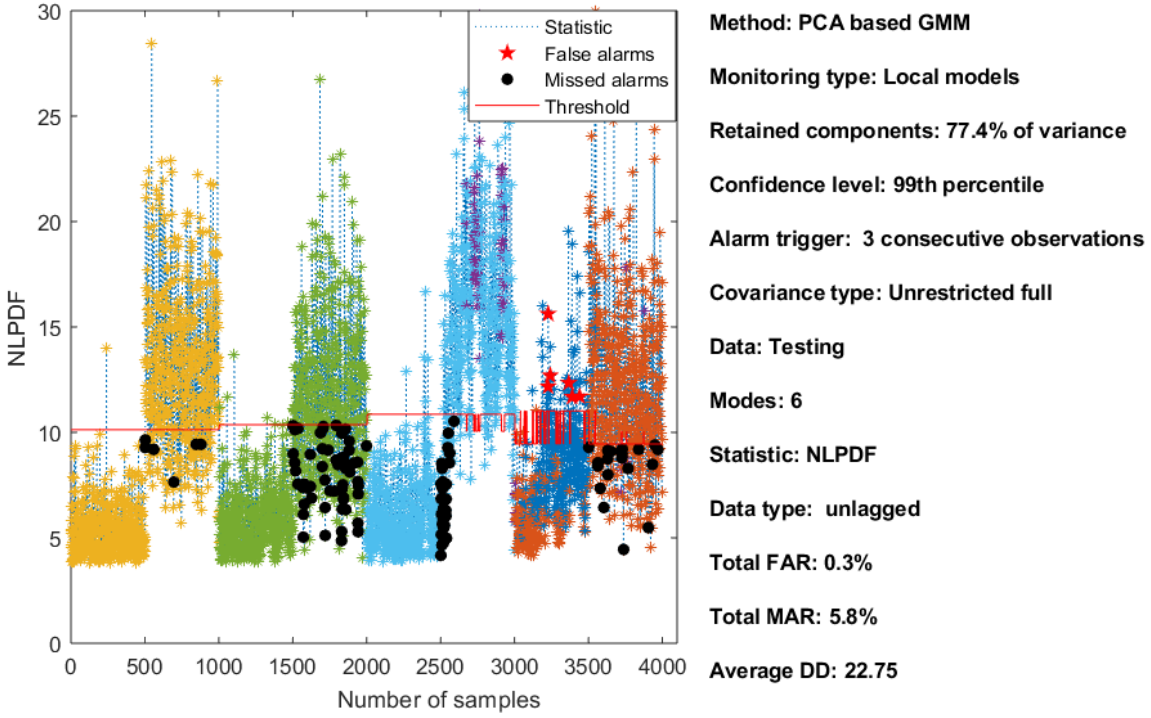


Figure G.26: NLPDF statistics for Fault C07 for PCA-based GMM for 77% of variance.

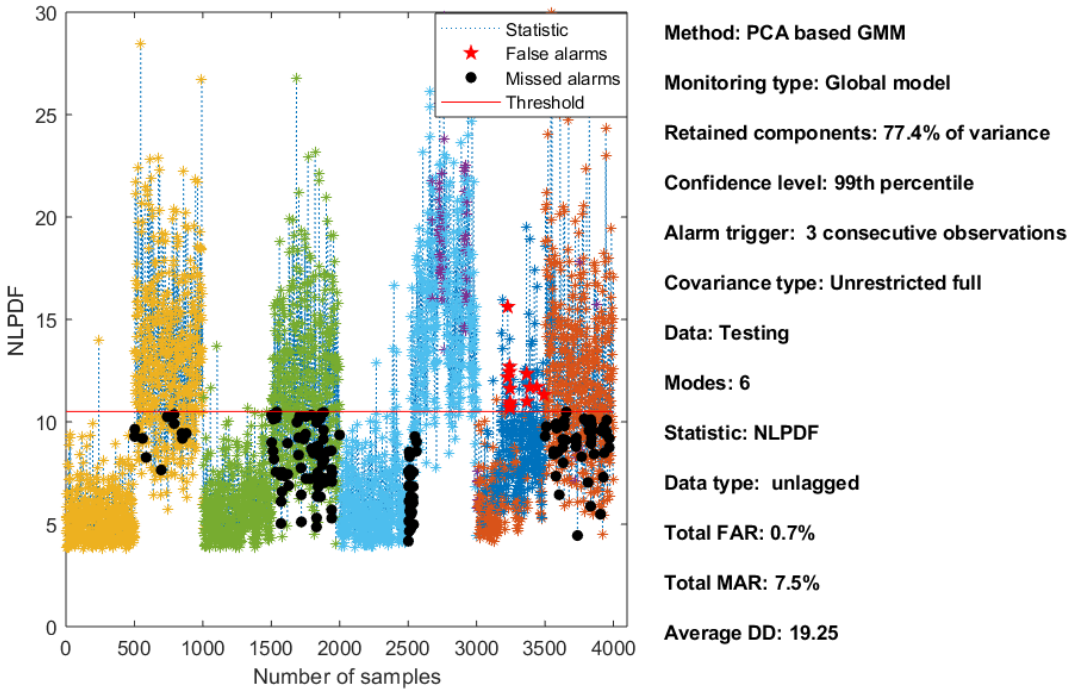


Figure G.27: NLPDF statistics for Fault C07 for PCA-based GMM for 77% of variance.

Sample results for normalized data based GMM (in Figure G.28) show the lowest FAR as compared to the other three scenarios but with a little higher MAR than that for scores with 5 retained PCs.

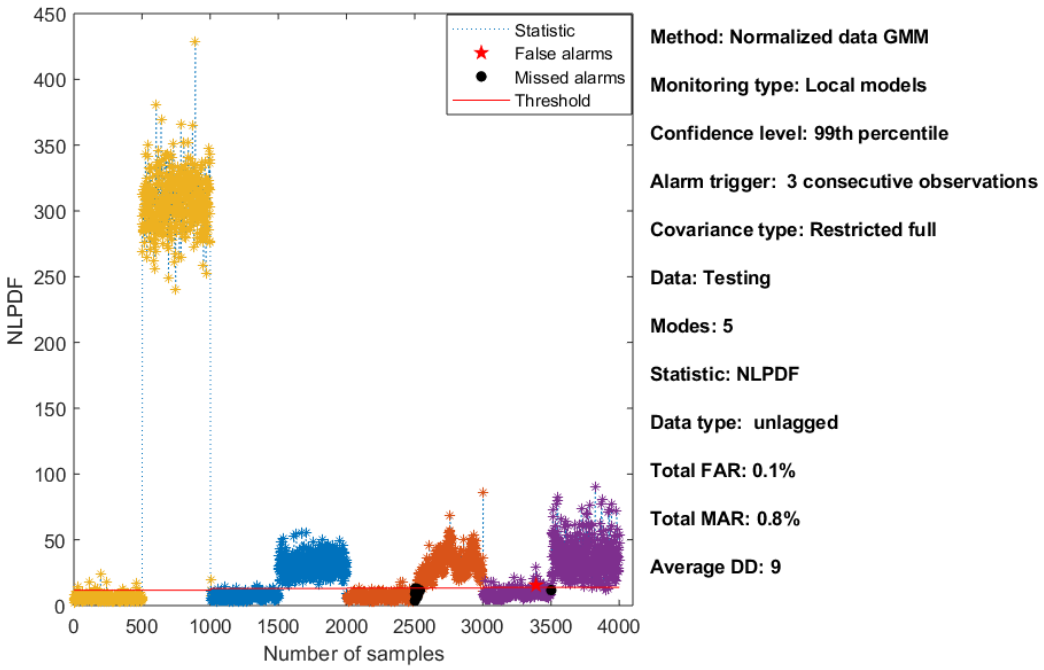


Figure G.28: NLPDF statistics for Fault C07 for normalized data based GMM.

A further analysis of the FAR for the various data types (in Figure G.29) shows the highest amount of FAR when a full-shared covariance is used for almost all the data types. Best performing covariance type is that for diagonal-shared almost always shows the lowest FAR over all data types. Apart from the results for the diagonal-unshared covariance, the results for the PCA scores with 9 retained PCs showed close results for the covariance types

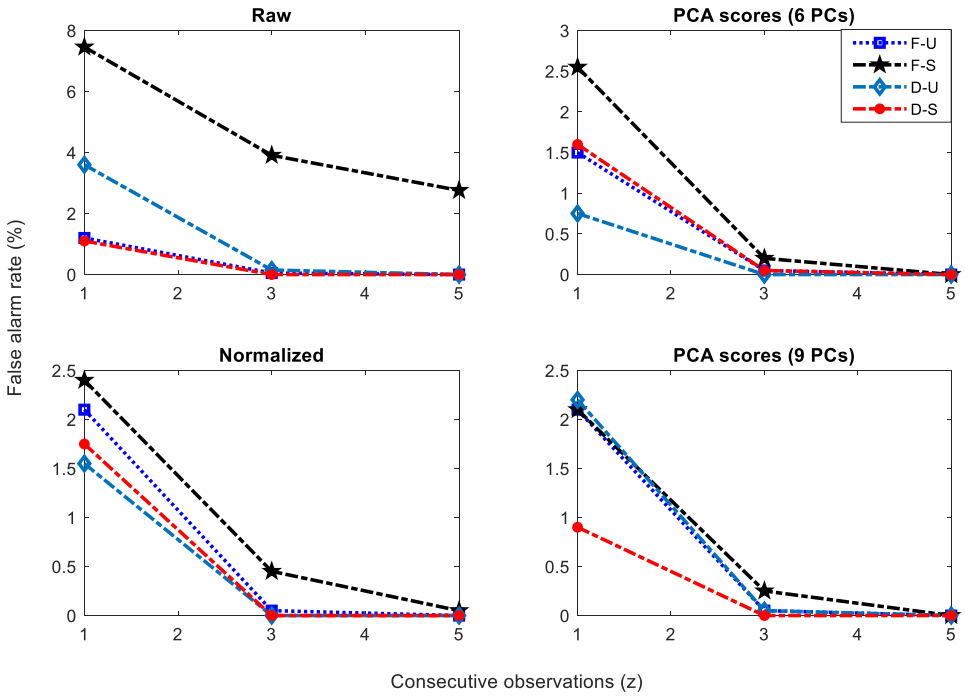


Figure G.29: FAR for various data types evaluated at 99.5th percentile and with local models approach.

Results for the MAR, on the other hand, shows high values for the diagonal covariance when used for the raw data, which is not unexpected. The unexpected poor performance is that for diagonal-shared covariance for the PCA scores with six retained PCs. The full-unshared covariance type has proved to provide the best results for all instances. This is also not unexpected as experienced from the cluster number selection section. Results for PCA scores with all retained PCs (i.e. nine) has shown to provide best results for all covariance types. This also not unexpected because the diagonal covariance proves suitable in score space.

The poor performance for the retained PCs of six could be attributed to information loss in the discarded components. This is synonymous with the cases experienced for the TE process faults four and five.

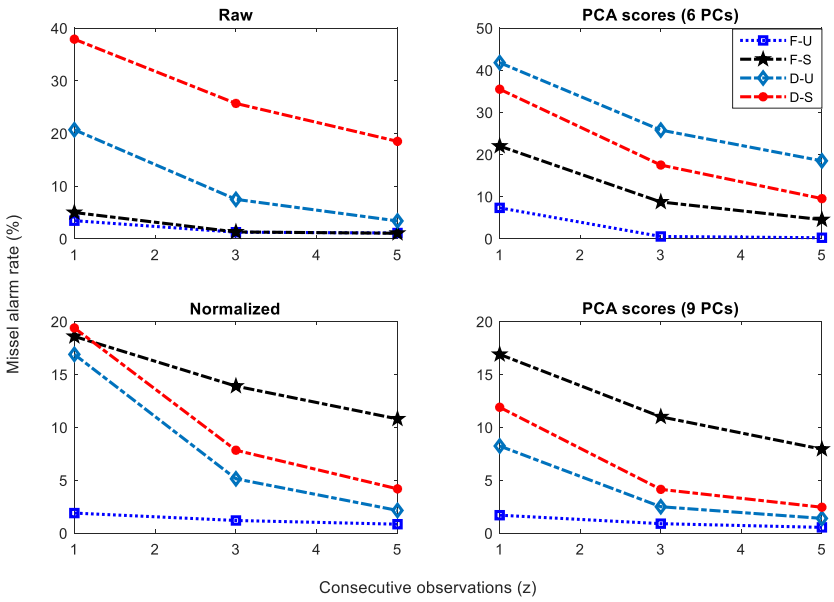


Figure G.30: MAR for various data types evaluated at 99.5th percentile and with local models approach.

Overall results for total alarm rates gives full-unshared covariance the edge over the other types, with PCA scores GMM with nine retained PCs providing the best results.

Comparison of the MAR of the global and local monitoring approaches (in Figure G.31) show close results for the cases with the full covariance matrix. Results for the diagonal matrix on the hand favours the local models in all cases apart from that for the scores with six retained PCs.

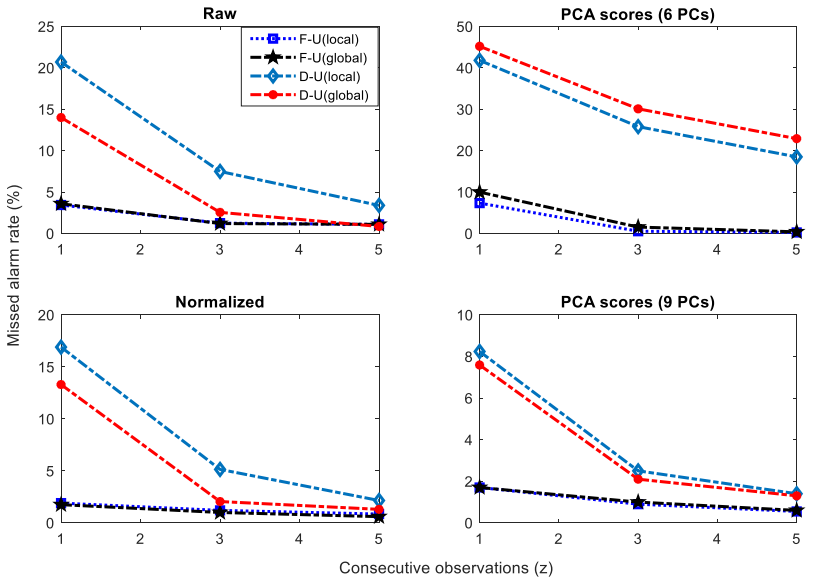


Figure G.31: MAR for various data types evaluated at 99.5th percentile and with local and global models approach.

Results for the FAR (in Figure G.32) on the other hand shows the normalized data having a slight upper edge over the PCA scores with 9 retained PCs. The local monitoring model once again provides better results than that for the global approach; apart from the PCA scores with 6 PCs which shows a slight deviation at an instance.

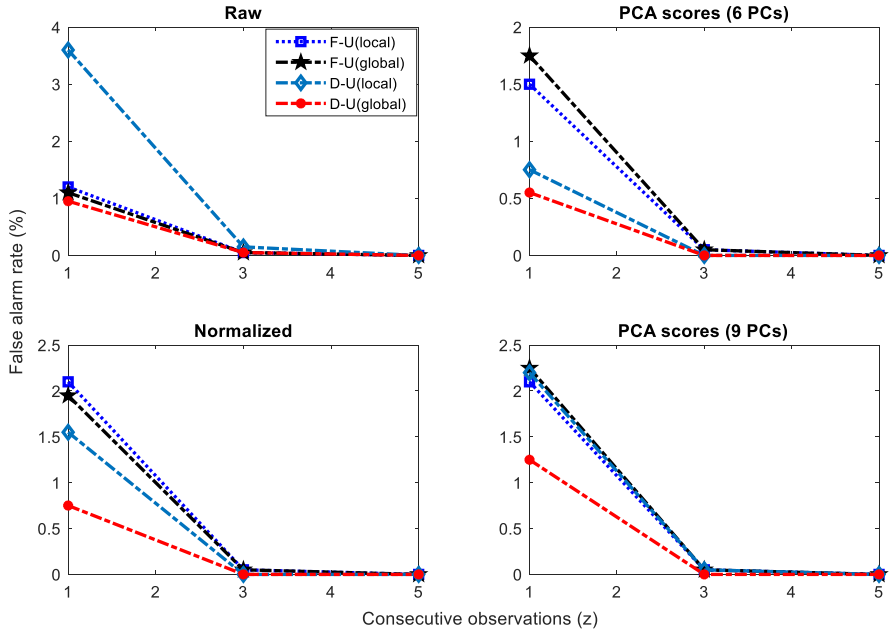


Figure G.32: FAR for various data types evaluated at 99.5th percentile and with local and global models approach.

Overall alarm rates put PCA scores with all retained PCs on top of normalized data, with the raw data also having a little edge over the scores with few PCs in the lower two performers.

Further analysis for the effect of retained PCs as shown in Figure G.33, supports the earlier analysis of information loss made for the 6 and 9 retained PCs, as the monitoring approach with 5 PCs performs worse than that for the 9 PCs as well as the 6 PCs.

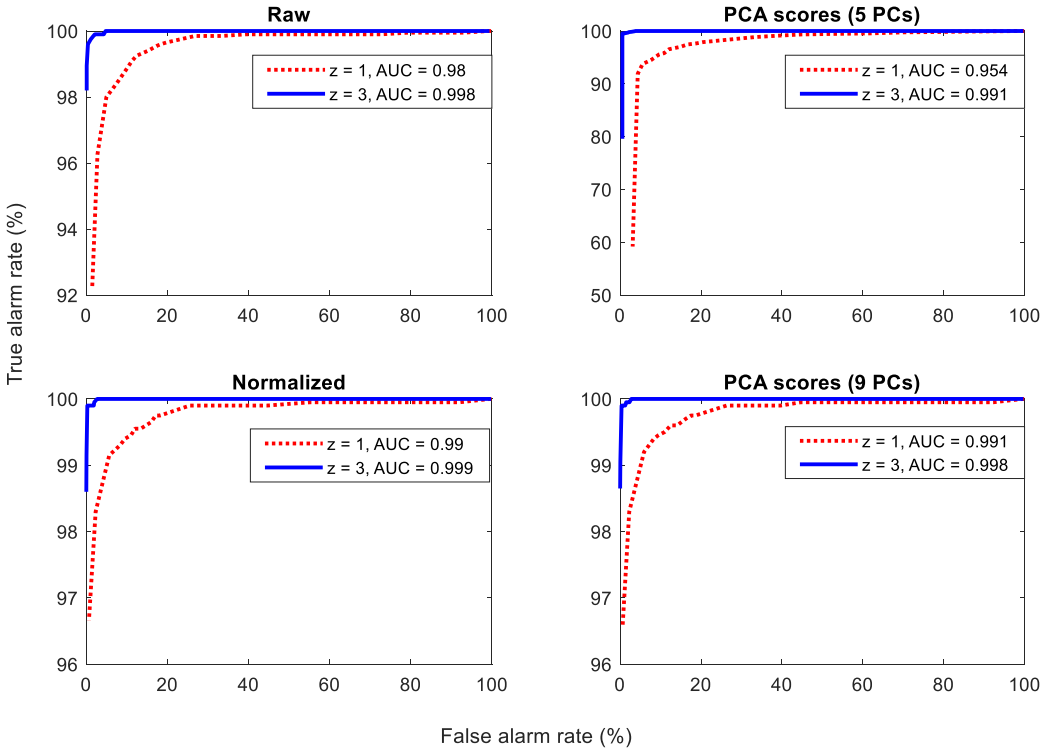


Figure G.33: AUCs for various data types evaluated at 99.5th percentile and with local models approach.

APPENDIX H: APCA-BASED GMM

Appendix H provides a comparison of results for the unimodal and multimodal approaches with a focus on APCA-based GMM.

H.1 Fault detection for T01

Figure H.1shows the results for APCA-based GMM. The fault is easy to detect for all other considered methods as established in the previous sections and confirmed by Figure H.2. The results show close performance across all methods employed.

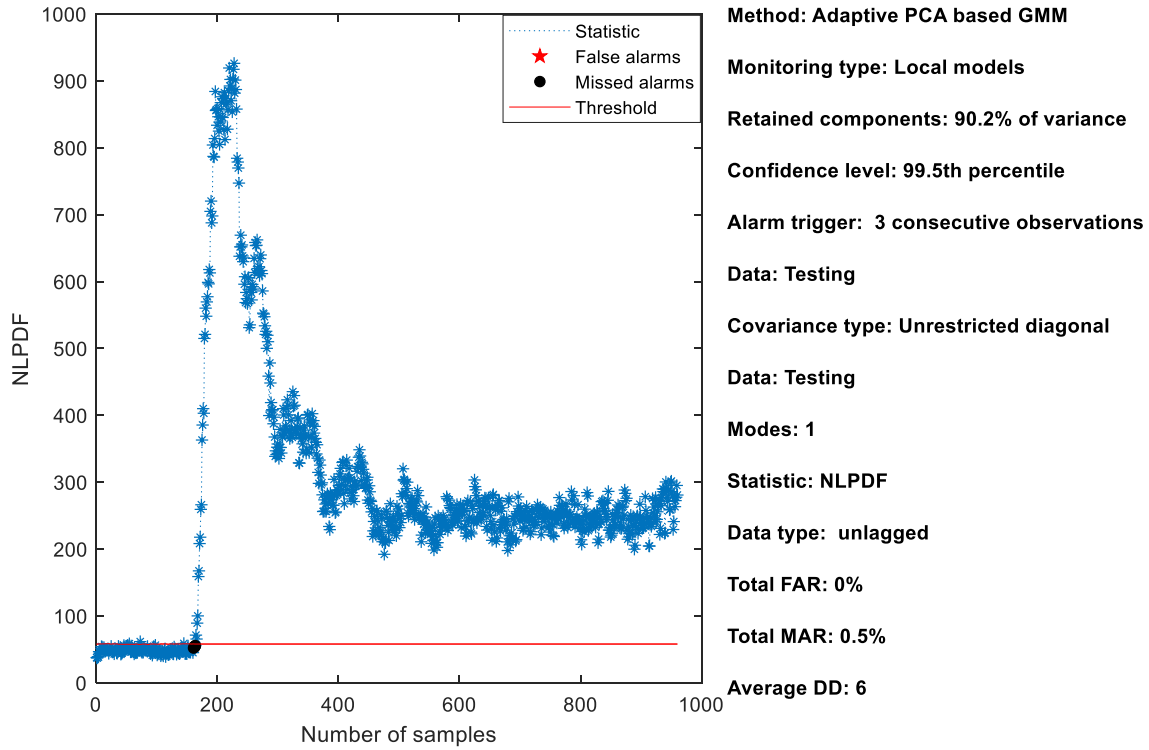


Figure H.1: NLPDF statistics for Fault T01 of APCA-based GMM.

	PCA		MWPCA		PCA-GMM	MWPCA-GMM
	SPE	T^2	SPE	T^2	NLPDF	NLPDF
Total FAR	0.6	0	0	0	0	0
Total MAR	0.1	0.5	0.3	0.5	0.5	0.5
Avg DD	3	8	4	6	6	6

Figure H.2: Monitoring performance for Fault T01 evaluated at 90% retained variance, 99th percentile threshold and z = 3.

H.2 Fault detection for C05

Figure H.3 and Figure H.4 respectively show sample results for APCA-based GMM and PCA-based GMM. The results clearly show a better performance by the former.

Results for the APCA-GMM show 4 modes fitted to the model, this is because the model is re-trained at intervals which is also unsupervised. Although the model may start with a single mode, that number does not remain constant.

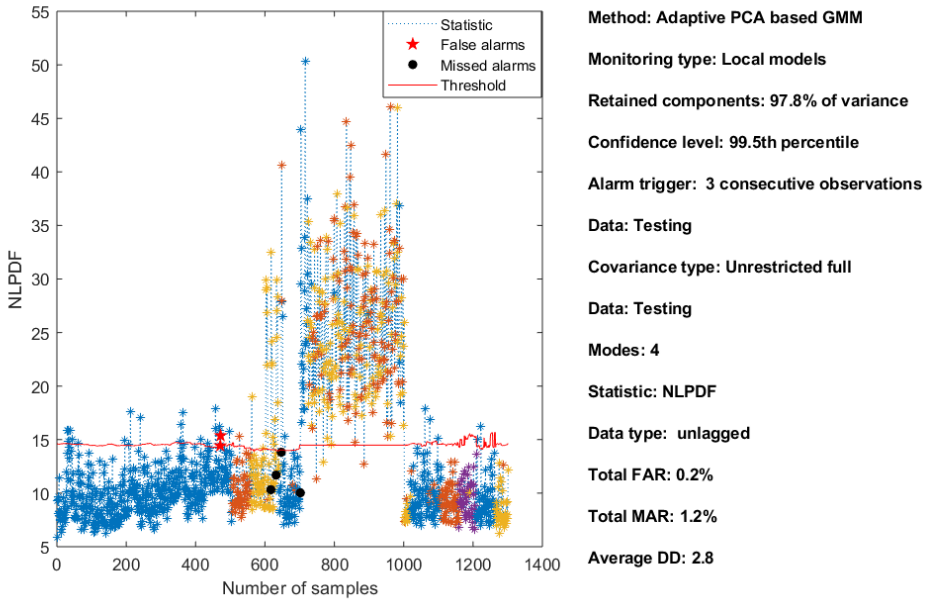


Figure H.3: NLPDF statistics for Fault C05 of APCA-based GMM.

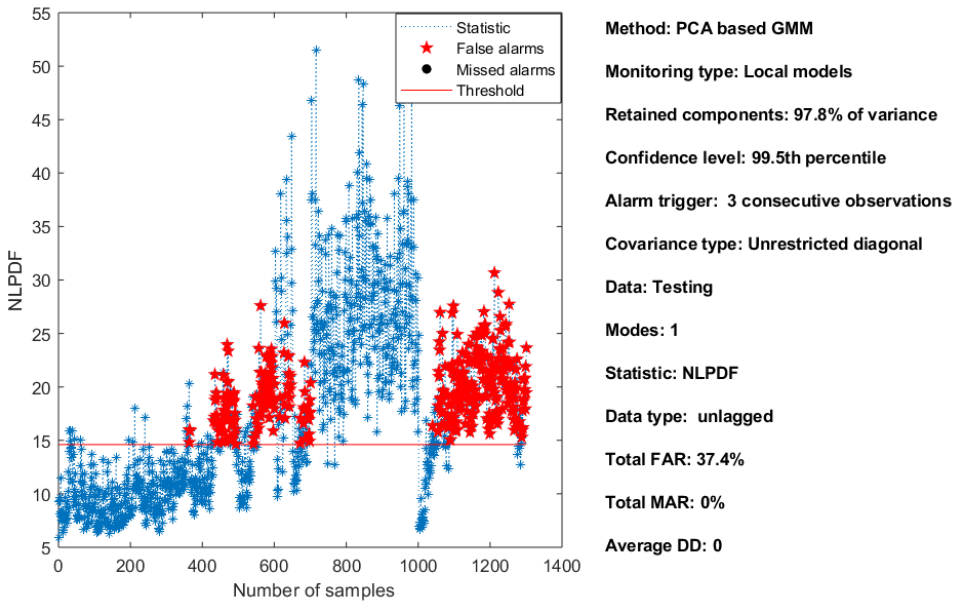


Figure H.4: NLPDF statistics for Fault C05 of PCA-based GMM.

H.3 Fault detection for C06

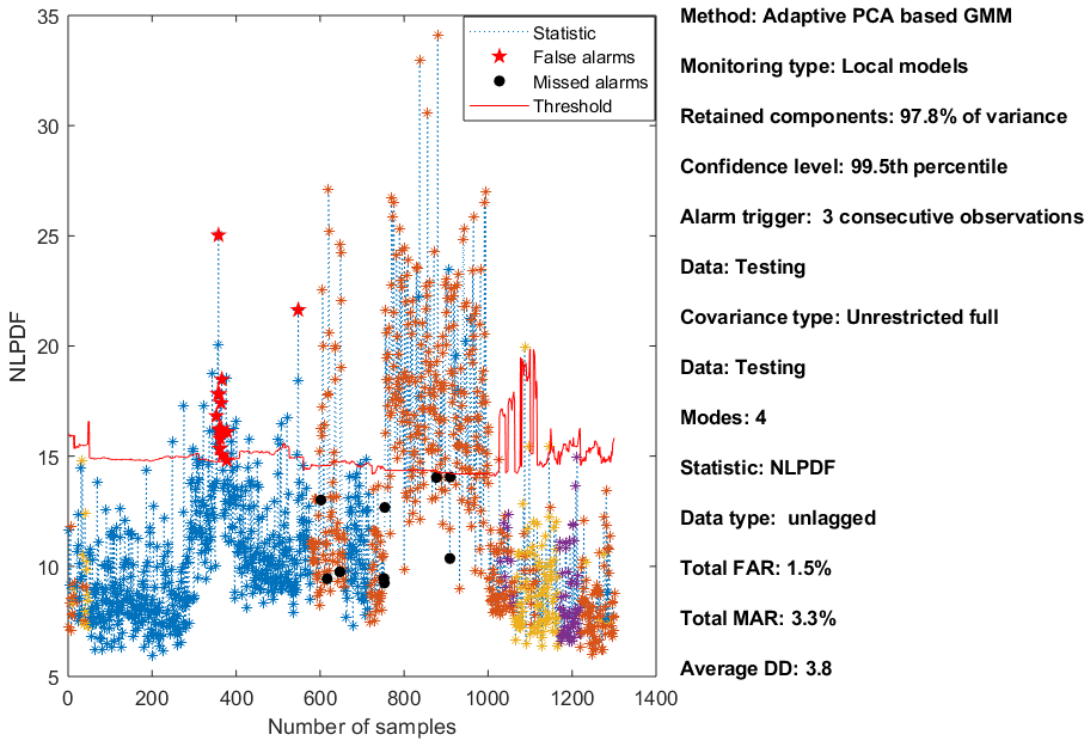


Figure H.5: NLPDF statistics for Fault C06 of APCA-based GMM.

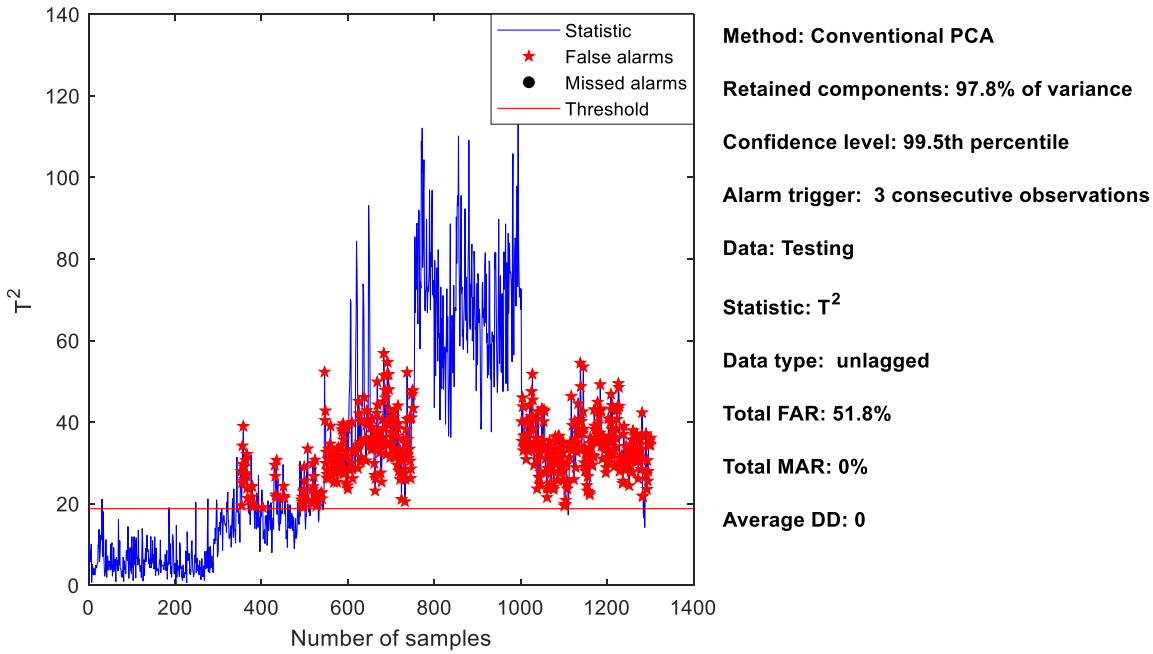


Figure H.6: T^2 statistic for Fault C06 of conventional PCA.

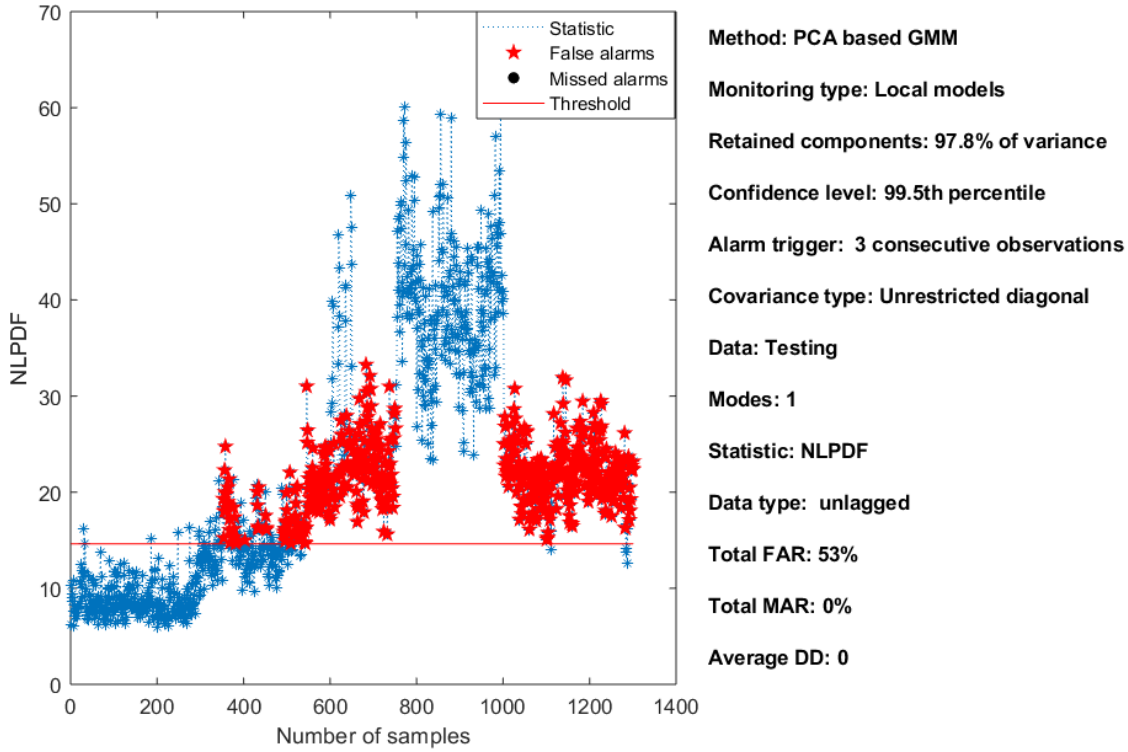


Figure H.7: NLPDF statistics for Fault C06 of PCA-based GMM.

H.4 Fault detection for C07

Figure H.8 and Figure H.9 respectively show the results of APCA-based GMM and PCA-based GMM. Both methods provide significantly similar results for the considered case. Threshold changes for the PCA-based GMM are that for changes in modes and not for adaptation (each mode is represented with distinct colour).

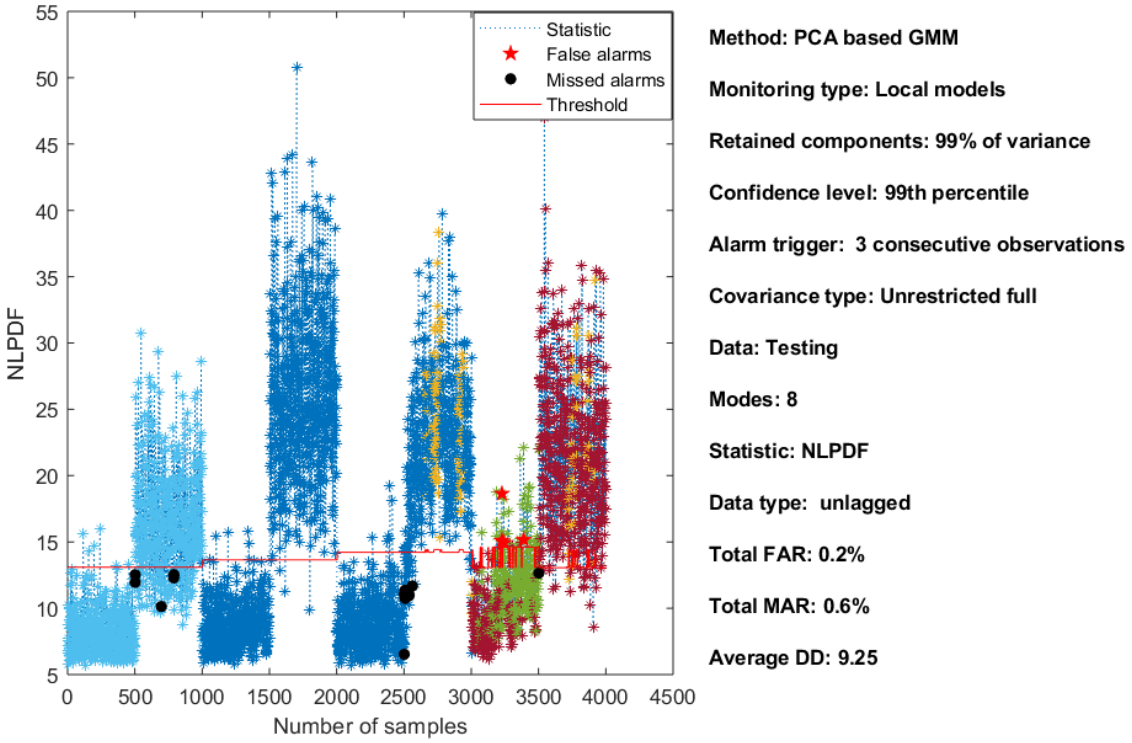


Figure H.8: NLPDF statistics for Fault C07 of PCA-based GMM.

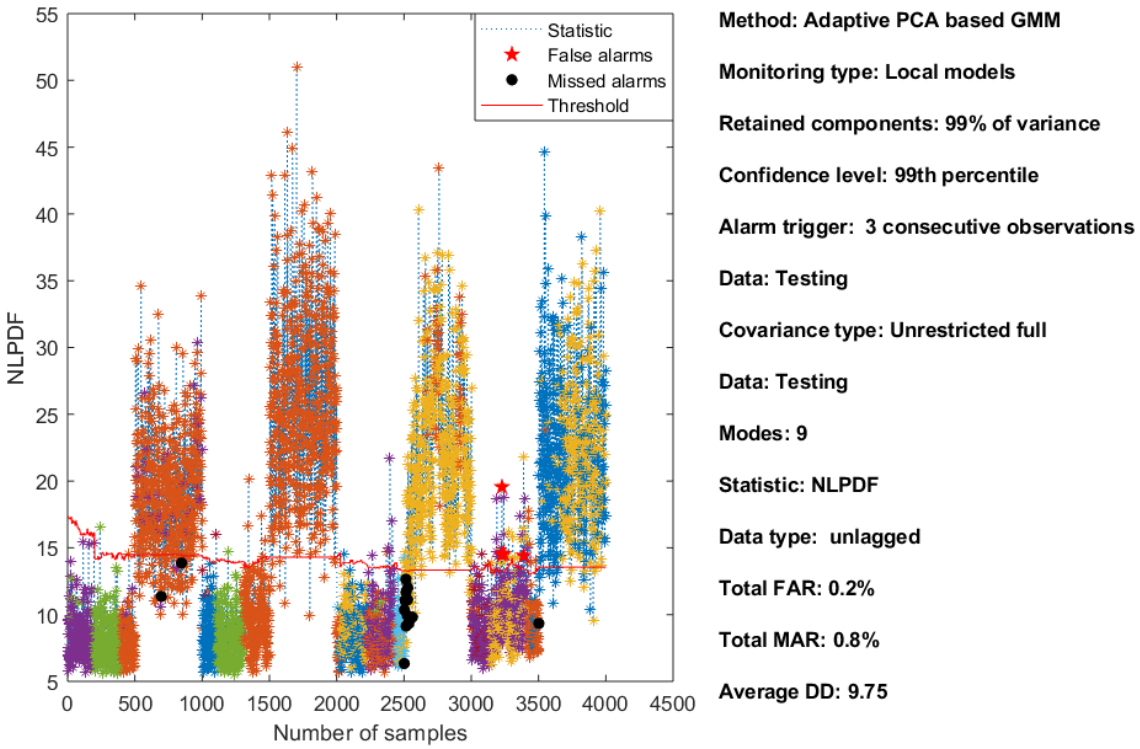


Figure H.9: NLPDF statistics for Fault C07 of APCA-based GMM.

H.5 CSTR process Fault C08

Test data C08 is generated using the same faults and modes as that specified for C07. A further extension is the introduction of process drift from time 202 to 802. The modes changes for modes 2, 3 and 4, however, occurred at times 1002, 1502 and 2202 respectively. The Tc and Ti sensor bias faults occurred at times 802 to 1002 and 1302 to 1502 respectively. The step faults in Ti and Ca respectively occurred from times 1802 to 2202 and 2702 to 3001. A total number of samples is 3001. Figure H.10 shows the simulation results for C08.

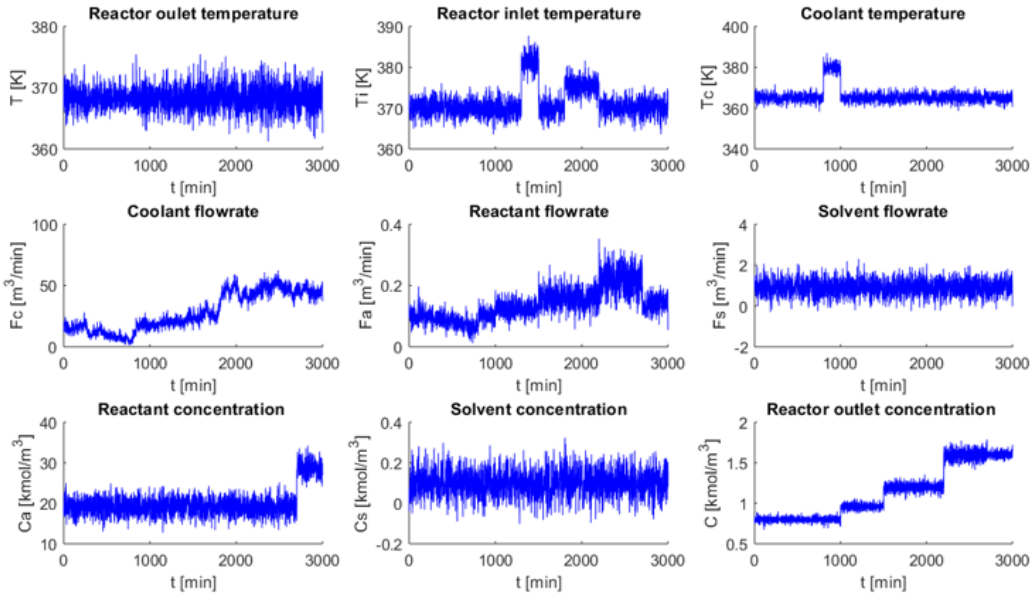


Figure H.10: Simulation results for process drift and process faults with the response from manipulated variables Fc and Fa.

H.6 Fault detection for C08

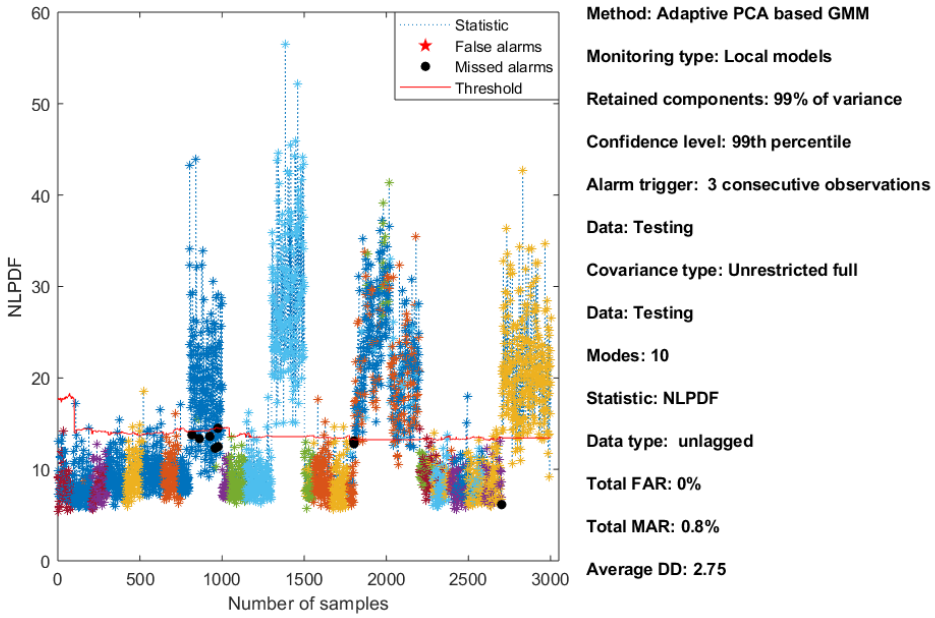


Figure H.11: NLPDF statistics for Fault C08 of APCA-based GMM.

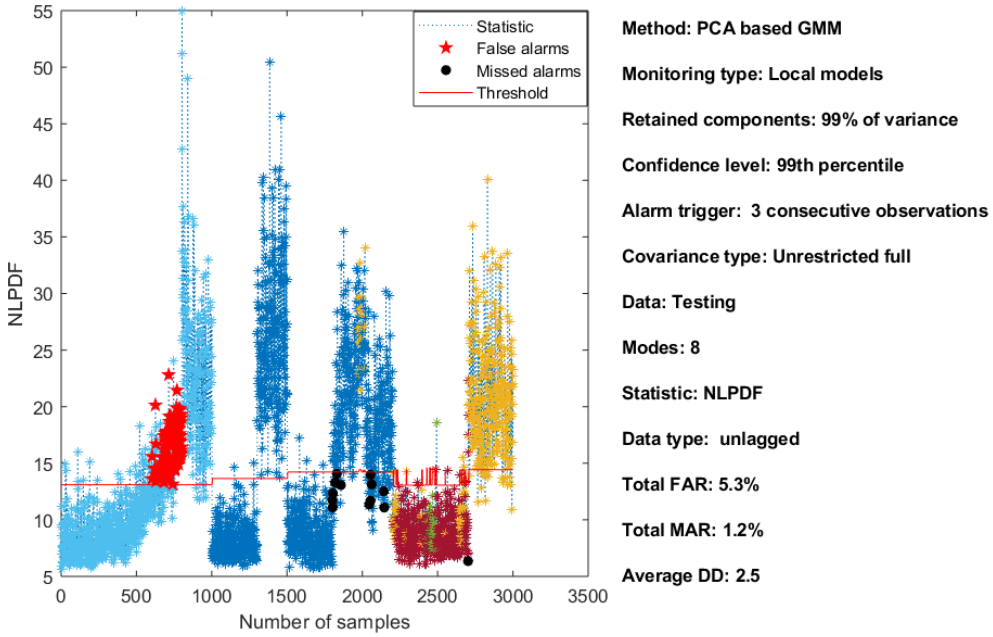


Figure H.12: NLPDF statistics for Fault C08 of PCA-based GMM.

Overall results for APCA-based GMM shows an added advantage to the multimodal approach. The APCA-based GMM is comparable to the APCA method as well in its adaptive ability. Also, fitting a large number of clusters to 'normal data' does not generally impact the monitoring performance negatively, i.e. fitting more modes is better than fewer modes.