INTERVAL PRINCIPAL COMPONENT ANALYSIS AND ITS APPLICATION TO FAULT DETECTION AND DATA CLASSIFICATION

A Thesis

by

NOUR MANSOUR ABDELHAFEZ MOHAMED BASHA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2018

Major Subject: Chemical Enigneering

ABSTRACT

Principal Component Analysis (PCA) is a linear data analysis tool that aims to reduce the dimensionality of a dataset, while retaining most of the variation found in it. It transforms the variables of a dataset into a new set of variables, called principal components, using linear combinations of the original variables. PCA is a powerful statistical technique used in research for fault detection, classification and feature extraction. Interval Principal Component Analysis (IPCA) is an extension to PCA designed to apply PCA to large datasets using interval data generated from single-valued samples. In this thesis, three IPCA methods are introduced: Centers IPCA (CIPCA), Midpoint-Radii IPCA (MRIPCA), and Symbolic Covariance IPCA (SCIPCA). In addition, the methods and parameters used for fault detection and classification applications are described for classical and interval data.

The performance of the methods used for interval generation in IPCA are analyzed under different conditions. Moreover, three synthetic datasets were used to test the fault detection performances of all methods, and three real datasets were used to test their classification performances. The results show that IPCA methods have a higher detection rate than classical PCA on average, for the same false alarm rate. Moreover, unlike PCA, IPCA methods are capable of accurately differentiating the type of fault. Interval centers were capable of detecting changes in mean, while interval radii were capable of detecting changes in variance. On the other hand, for data classification, the results show that MRIPCA has the highest precision on average than other methods.

ACKNOWLEDGMENTS

I am deeply grateful to my advisors, Dr. Mohamed Nounou and Dr. Hazem Nounou, for all of their feedback and guidance throughout this process. I would also like to acknowledge my committee member, Dr. Ahmed Abdel-Wahab. Thanks to all the members and staff of Texas A&M University at Qatar.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Dr. Mohamed Nounou and Dr. Ahmed Abdel-Wahab of the Chemical Engineering Department, and Dr. Hazem Nounou of the Electrical and Computer Engineering Department.

All work for this thesis was completed by the student, under the advisement of Dr. Mohamed Nounou of the Chemical Engineering Department, and Dr. Hazem Nounou of the Electrical and Computer Engineering Department.

Funding Sources

Graduate study was supported by the Itochu Professorship of Dr. Hazem Nounou.

NOMENCLATURE

CIPCA	Centers Interval Principal Component Analysis
DR	Detection Rate
EDF	Empirical Distribution Function
FAR	False Alarm Rate
FDU	Fault Detection Unit
IPCA	Interval Principal Component Analysis
MRIPCA	Midpoint-Radii Interval Principal Component Analysis
m	The Number of Samples Aggregated per Interval
PCA	Principal Component Analysis
SCIPCA	Symbolic Covariance Interval Principal Component Analysis
SNR	Signal-to-Noise Ratio
SPC	Statistical Process Control
β	Interval Width
ϵ	Noise Matrix
σ	Standard Deviation
μ	Mean

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Statistical Process Control

Statistical process control, or process monitoring, is needed for a consistent high level of quality control and safety. In a process with multiple inter-correlated variables, it is vital to be able to detect any abnormalities, or deviations from optimal operating conditions, in order to avoid damaging the equipment (for example, due to runaway reactions), or to reduce costs by minimizing wasted resources. The idea is to make process adjustments so as to avoid these deviations from occurring again. Process monitoring is carried out using control charts, which detect and display any variability in the system [1].

In its most basic form, a control chart is a time-based plot of the current state of the system and a limit line of the optimal or tolerable operating conditions. Process monitoring is developed in two phases. First, finding the limit of a fault-free operation, thereby defining the threshold value beyond which the system is considered out-of-control. Second, finding the state of an unknown operation in real-time, and comparing it to the threshold value for the detection of any sub-optimal conditions. Examples of systems that may use real-time process monitoring are waste water treatment plants, distillation columns in oil and gas applications or a food-processing unit [2]. There are several methods available to monitor the state of the system and build control charts. One of the most powerful linear methods is the Principal Component Analysis.

1.2 Principal Component Analysis

Principal Component Analysis (PCA) is a powerful linear data analysis tool used to reduce the dimensionality (i.e. number of variables) of a dataset of cross-correlated variables, while simultaneously retaining most of their variability. It was first introduced by Pearson [3], and later developed for statistical monitoring by Hotelling [4]. PCA linearly combines the original variables into a new set of variables, called principal components (PCs). PCs have orthogonal loadings, and are thus decorrelated, for real data [5], which is crucial for fault detection and classification applications. Moreover, in order to compare variables with different magnitudes and keep the data dimensionless, variables are normalized to zero mean and unit variance before applying PCA. The original variability in the dataset is redistributed so as to be located in the first few PCs. The ones retained can be referred to as core PCs, while the remaining are residual PCs. The latter are assumed to hold the variability in the dataset caused by noise, and removing them is akin to filtering out noise from the dataset.

1.2.1 Algorithm

In process monitoring, PCA requires two datasets: training and testing. Training datasets, labeled with X, define the system under normal operating conditions (i.e. without any faults) and are used to create the PCA model. Testing datasets, labeled with S, define the system under faulty or sub-optimal operating conditions, where the objective of PCA is to detect this deviation using its created model. Given an $n \times p$ classical training dataset X defined as,

$$X = \begin{array}{c} \\ \text{Sample } 1 \\ \text{Sample } 2 \\ \vdots \\ \text{Sample } n \end{array} \begin{array}{c} \begin{array}{cccc} \text{Var. } 1 & \text{Var. } 2 & \cdots & \text{Var. } p \end{array} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix} \end{array}, \tag{1.1}$$

where n is the number of samples and p is the number of variables, the PCA model can be constructed as follows:

1. Find the correlation matrix R of X. This is equivalent to the covariance matrix of the normalized X.

2. Find the eigenvectors matrix P and the diagonal eigenvalues matrix Λ of R. The eigenvectors define the linear combination coefficients of the PCs, and the eigenvalues represent the amount of variance that each PC covers in the dataset.

   Reorder the eigenvectors in decreasing order of their respective eigenvalues' magnitudes. The eigenvalues for each variable are found as follows:

$$\det\left(R - \lambda_i I\right) = 0 \quad \text{for } 1 \leq i \leq p, \tag{1.2}$$

   where det() is the determinant function. For each eigenvalue, its eigenvector v_i is calculated as follows:

$$\left(R - \lambda_i I\right) v_i = 0. \tag{1.3}$$

3. Retain the eigenvectors corresponding to the largest l eigenvalues, where $l \leq p$. Define the matrix $\hat{P}$, containing only the retained eigenvectors.

   There are different ways to decide how many principal components to retain in PCA:

   3.1. Using the Cumulative Percent Variance (CPV) criterion, find the l core principal components such that

$$l = \arg\min_{l} \left| \frac{\sum_{i=1}^{l} \lambda_i}{\sum_{j=1}^{p} \lambda_j} - \rho \right|, \tag{1.4}$$

   where ρ is the desired CPV, which is a percentage value defining the minimum amount of variance to be retained in the dataset.

   3.2. Using the scree plot, which plots the eigenvalues versus their respective indices, as seen in Figure 1.1. Since the eigenvalues are reordered in decreasing order, the plot will typically be a decreasing elbow curve. A horizontal line is added to the bottom of the curve, such that it intersects with the scree plot at a point beyond which its slope is approximately zero. The number of core principal components is the index value of that intersection.

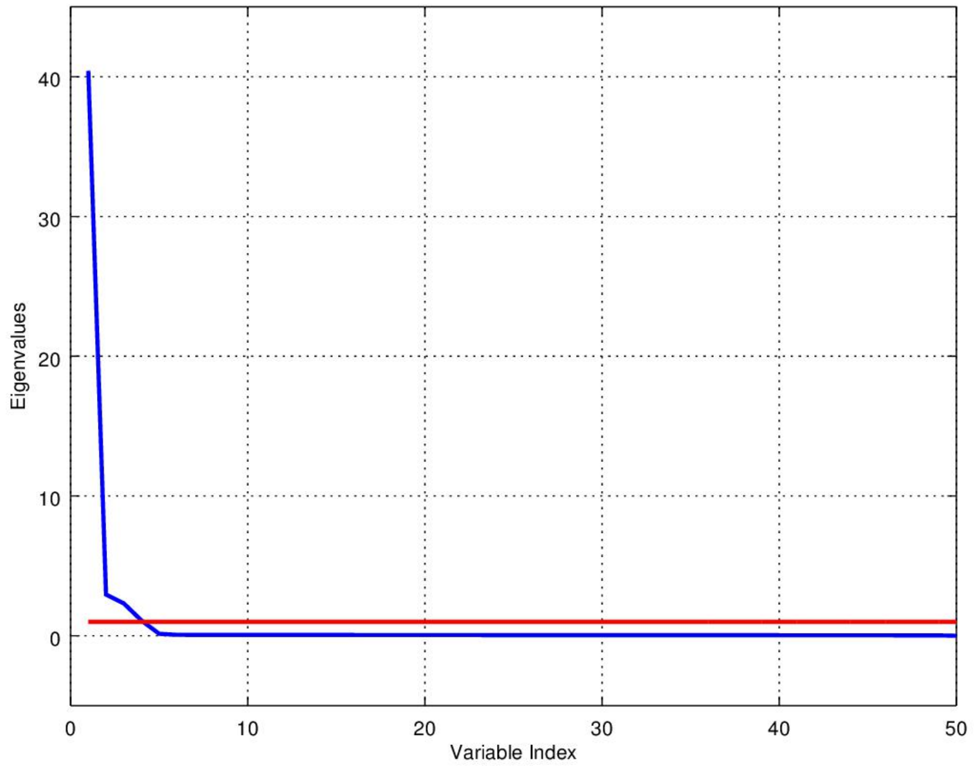


Figure 1.1: Example of a scree plot. The blue line shows how the eigenvalues change for a system of 50 inter-correlated variables. The red line intersects with the blue line at the point after which all eigenvalues remain relatively constant and negligible in value.

3.3. Using the Guttman-Kaiser criterion, where only the PCs corresponding to eigenvalues greater than or equal to 1 are retained. However, this method seems to retain more principal components on average [6].

3.4. Using the Mean Squared Error (MSE), find the number of eigenvectors that minimizes its value. The MSE is the distance between the training dataset denoised by PCA ($\hat{X}$) and the noise-free training dataset (X^0). The noise-free dataset is typically only available in synthetic models, where the mathematical model is known.

     In the case where the noise-free dataset is not available, this method can still be applied using the noisy dataset (X). The optimal number of PCs to retain is found using the same method applied for the scree plot. We find the point at which the slope of the MSE curve approaches zero.

4. Find the predictive transformation matrix $\hat{C} = \hat{P}\hat{P}^\mathsf{T}$, and the predicted dataset $\hat{X} = X\hat{C}$. The predicted dataset is equivalent to X denoised by the PCA model.

5. Find the residual transformation matrix $\tilde{C} = I - \hat{C}$, and the matrix of residuals $\tilde{X} = X\tilde{C}$. The matrix of residuals is equivalent to the filtered out noise from X.

The transformation matrices $\hat{C}$ and $\tilde{C}$ define the PCA model. Given a testing dataset S, the matrix of residuals $\tilde{S} = S\tilde{C}$ defines the degree of deviation of S from the PCA model for each sample, and, if large enough, a fault can be declared. These residuals are used in fault detection and classification, as will be seen later. For more comprehensive details of PCA, please refer to [5, 7].

1.2.2 Applications

PCA has been explored in literature for many engineering applications, where it was important to extract underlying dominant features without a priori knowledge of the data. For example, PCA has been applied to Multi-Sensor Data Fusion in [8], where measurements of a machine's perceived smell and taste of input food samples were gathered from multiple sensors in parallel, used to quantify the quality of the food samples.

Another example is in the study of the trajectories of a system's constituent atoms as defined by Molecular Dynamics (MD) [9], where generated data matrices are typically high-dimensional and rich in samples since it commonly operates in minuscule time-steps, sometimes in the order of 1 femtosecond. The authors concluded that the PCA's ability to extract the "essential subspace" is useful in understanding the dynamics of the system. However, there were reservations regarding the accuracy and robustness of PCA, where sampling issues may arise for massive datasets, and stated that it must be clearly investigated in order to avoid any misleading results.

Other engineering applications include gas chromatographic analysis, video and image processing, climatic variability and automatic target recognition [6]. PCA has also been applied in other multidisciplinary applications, such as biology, ecology, health and finance [10]. For example, Belasco et al. [11] used PCA in order to explore the relationship between different socioeconomic variables and the rate of cancer mortality and delayed detection.

1.3 Interval Data

The use of interval data is motivated by applications involving large datasets, in order to reduce them to a more manageable size [12]. In addition, it is motivated by applications involving datasets with missing values, which, for example, may be due to varying sampling frequencies for different variables in the dataset or malfunctioning sensors, which may result in corrupted/lost data samples.

Interval samples cannot be naturally measured but are artificially generated using classical samples. An interval is defined using a lower and upper bound: $[a, b]$, where $a \leq b$ and $\{a, b\} \in \mathbb{R}$. Basic interval arithmetic is defined as follows: given two elementary intervals $[a,b]$ and $[c,d]$,

$$\begin{aligned}
[a,b]+[c,d] &= [a+c, b+d]\,, \\
[a,b]-[c,d] &= [a-d, b-c]\,, \\
[a,b]*[c,d] &= \left[\min\left(ac,ad,bc,bd\right), \max\left(ac,ad,bc,bd\right)\right], \\
\frac{[a,b]}{[c,d]} &= \left[\min\left(\frac{a}{c},\frac{a}{d},\frac{b}{c},\frac{b}{d}\right), \max\left(\frac{a}{c},\frac{a}{d},\frac{b}{c},\frac{b}{d}\right)\right] \\
&\iff \text{sgn}\left(c\right)=\text{sgn}\left(d\right)\neq 0,
\end{aligned} \tag{1.5}$$

where sgn$(\cdot)$ is the signum function, whose output is necessarily 1, -1 or 0.

Interval datasets, identified using square brackets ($[\cdot]$), are defined as

$$[X] = \begin{array}{c} \\ \text{Sample } 1 \\ \text{Sample } 2 \\ \vdots \\ \text{Sample } n \end{array} \begin{array}{c} \begin{array}{cccc} \text{Var. } 1 & \text{Var. } 2 & \cdots & \text{Var. } p \end{array} \\ \begin{bmatrix} \left[\underline{x}_{1,1},\overline{x}_{1,1}\right] & \left[\underline{x}_{1,2},\overline{x}_{1,2}\right] & \cdots & \left[\underline{x}_{1,p},\overline{x}_{1,p}\right] \\ \left[\underline{x}_{2,1},\overline{x}_{2,1}\right] & \left[\underline{x}_{2,2},\overline{x}_{2,2}\right] & \cdots & \left[\underline{x}_{2,p},\overline{x}_{2,p}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \left[\underline{x}_{n,1},\overline{x}_{n,1}\right] & \left[\underline{x}_{n,2},\overline{x}_{n,2}\right] & \cdots & \left[\underline{x}_{n,p},\overline{x}_{n,p}\right] \end{bmatrix} \end{array}. \tag{1.6}$$

where the lower and upper bound matrices are respectively defined as,

$$X^l = \begin{bmatrix} \underline{x}_{1,1} & \underline{x}_{1,2} & \cdots & \underline{x}_{1,p} \\ \underline{x}_{2,1} & \underline{x}_{2,2} & \cdots & \underline{x}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{x}_{n,1} & \underline{x}_{n,2} & \cdots & \underline{x}_{n,p} \end{bmatrix}, \quad X^u = \begin{bmatrix} \overline{x}_{1,1} & \overline{x}_{1,2} & \cdots & \overline{x}_{1,p} \\ \overline{x}_{2,1} & \overline{x}_{2,2} & \cdots & \overline{x}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x}_{n,1} & \overline{x}_{n,2} & \cdots & \overline{x}_{n,p} \end{bmatrix}. \tag{1.7}$$

Moreover, the center and radii matrices are respectively defined as,

$$
\begin{aligned}
X^c &= \frac{1}{2}\begin{bmatrix} \underline{x}_{1,1}+\overline{x}_{1,1} & \underline{x}_{1,2}+\overline{x}_{1,2} & \cdots & \underline{x}_{1,p}+\overline{x}_{1,p} \\ \underline{x}_{2,1}+\overline{x}_{2,1} & \underline{x}_{2,2}+\overline{x}_{2,2} & \cdots & \underline{x}_{2,p}+\overline{x}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{x}_{n,1}+\overline{x}_{n,1} & \underline{x}_{n,2}+\overline{x}_{n,2} & \cdots & \underline{x}_{n,p}+\overline{x}_{n,p} \end{bmatrix}, \\
X^r &= \frac{1}{2}\begin{bmatrix} \overline{x}_{1,1}-\underline{x}_{1,1} & \overline{x}_{1,2}-\underline{x}_{1,2} & \cdots & \overline{x}_{1,p}-\underline{x}_{1,p} \\ \overline{x}_{2,1}-\underline{x}_{2,1} & \overline{x}_{2,2}-\underline{x}_{2,2} & \cdots & \overline{x}_{2,p}-\underline{x}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x}_{n,1}-\underline{x}_{n,1} & \overline{x}_{n,2}-\underline{x}_{n,2} & \cdots & \overline{x}_{n,p}-\underline{x}_{n,p} \end{bmatrix}.
\end{aligned}
\tag{1.8}
$$

Unlike the lower and upper bound matrices, the centers and radii matrices are of particular importance in this thesis because they can be used to represent different unique properties of the datasets when generated via aggregation. Given a classical dataset X, we generate its respective interval dataset $[X]$ as follows. First, we split the samples of X into blocks of m classical samples each, such that the j-th variable's column is defined as

$$
X^{\text{agg}}_{(j)} = \begin{bmatrix} \{x_{1,j}\ ,\ \ldots\ ,\ x_{m,j}\} \\ \{x_{(m+1),j}\ ,\ \ldots\ ,\ x_{2m,j}\} \\ \vdots \\ \{x_{(n-m+1),j}\ ,\ \ldots\ ,\ x_{n,j}\} \end{bmatrix}, \tag{1.9}
$$

for $1 \le j \le p$.

Second, the samples in each block are separately aggregated into their own distinct intervals. Typically, aggregation is done by simply selecting the minimum and maximum values in each block as the lower and upper bound of each interval respectively. However, in this paper, aggregation is done such that the mean of the samples is taken as the interval center c, while the standard deviation of the samples is taken as its radius r, such that each interval is defined as $[c - r, c + r]$. Similarly, interval datasets are defined using their classical centers and radii matrices, such that $[X] = \{X^c, X^r\}$.

Aggregation of classical samples is expected to be more useful in applications with high sampling frequencies, or multiple independent parallel data inputs, due to its requirement to first gather a certain numbers of classical samples before processing. Intuitively, aggregation is not practical in applications where the classification of every classical sample is necessary, due to the unpredictability of that system's output. However, it is useful for applications where the states of the system are sustainable for at least as long as required to gather the needed classical samples.

1.4 Interval Principal Component Analysis

Interval Principal Component Analysis (IPCA) has been explored in literature for fault detection and isolation applications [13,14]. However, there are common oversights when evaluating the IPCA model's performance. First, the analysis is applied to a single iteration of a synthetic linear model. No Monte Carlo simulations are carried out to evaluate the effect that random noise has on the modeling or monitoring performance. Second, interval datasets are generated by setting the noise-free matrix as the centers, and the noise matrix as the radii for a limited number of samples. Since it is unusual for applications to have the noise-free matrix available, this method of interval generation is not easily transferable to real data. Third, the limited number of samples makes little use of the inherent benefit of interval data, which is the compilation of massive amounts of classical samples into a more manageable quantity for faster and more accurate computations.

In this paper, three IPCA methods are taken into consideration: the Centers IPCA (CIPCA), Midpoint-Radii IPCA (MRIPCA), and the Symbolic Covariance IPCA (SCIPCA).

Algorithms for three IPCA methods used are briefly described in the following sections, and they are extensions of the algorithm shown in Section 1.2.1.

1.4.1 Centers IPCA

The Centers IPCA (CIPCA) was introduced by Cazes et al. [15], where the idea is to focus on the variation between the intervals of a dataset, rather than the variations within them [12, 16]. Given an interval training dataset $\{X^c, X^r\}$, PCA is first applied to X^c, taken as the classical training dataset. The resulting predictive and residual transformation matrices, $\hat{C}^c$ and $\tilde{C}^c$ respectively, are then used to model the matrix of radii, X^r. Then, given an interval testing dataset $\{S^c, S^r\}$, the interval matrix of residuals is defined as $\left\{\tilde{S}^c, \tilde{S}^r\right\} = \left\{S^c\tilde{C}^c, S^r\tilde{C}^c\right\}$.

1.4.2 Midpoint-Radii IPCA

Midpoint-Radii IPCA (MRIPCA) was developed by Lauro et al. [17–20], where PCA is applied separately to the centers and radii matrices of an interval dataset. This method is best suited for modeling and monitoring because it takes into consideration the variations between and within the intervals of a dataset [14]. Le-Rademacher [16] discussed the use of a rotation matrix in order to allow MRIPCA to cover the variability caused by the inter-connection between the centers and radii. However, based on preliminary results, this step was not beneficial for the purposes of modeling and/or monitoring and was not implemented.

Given an interval training dataset $\{X^c, X^r\}$, PCA is applied in parallel to X^c and X^r, and thus four transformation matrices: $\hat{C}^c$, $\tilde{C}^c$, $\hat{C}^r$, and $\tilde{C}^r$ are estimated. Then, given an interval testing dataset $\{S^c, S^r\}$, the interval matrix of residuals is then defined as $\left\{\tilde{S}^c, \tilde{S}^r\right\} = \left\{S^c\tilde{C}^c, S^r\tilde{C}^r\right\}$.

1.4.3 Symbolic Covariance IPCA

Le-Rademacher et al. [16, 21] introduced the symbolic covariance matrix as a way to better represent the range and variability found in interval data. The authors also expanded their work to include histograms, where each interval is represented as a group of sub-intervals, each with its probability calculated from the data's distribution. Theoretically, this could prove beneficial in applications involving narrow or skewed distributions of data. However, it is not practically utilized due to the minimal improvement to fault detection performance it provides at the expense of a massive increase in computational time, especially for high-dimensional large matrices.

Given a training dataset $\{X^c, X^r\}$, the symbolic covariance matrix C_{sym} is calculated as follows

$$C_{sym}(i,j) = \frac{1}{3n}\sum_{k=1}^{n} G\left(k,i\right) G\left(k,j\right) \sqrt{Q(k,i)Q(k,j)}, \tag{1.10}$$

where,

$$\begin{aligned}
W(k,i) =& \frac{1}{2}(X^l(k,i) + X^u(k,i)) \\
\mu(i) =& \frac{1}{n}\sum_{k=1}^{n} W(k,i) \\
G(k,i) =& \begin{cases} -1, & \text{if } W(k,i) \le \mu(i) \\ 1, & \text{if } W(k,i) > \mu(i) \end{cases} \\
Q(k,i) =& (X^l(k,i) - \mu(i))^2 + (X^l(k,i) - \mu(i))(X^u(k,i) - \mu(i)) + \\
& (X^u(k,i) - \mu(i))^2,
\end{aligned} \tag{1.11}$$

for $1 \le i, j \le p$. The symbolic correlation matrix R_{sym} is then defined as

$$R_{sym}(i,j) = \frac{C_{sym}(i,j)}{\sqrt{C_{sym}(i,i)C_{sym}(j,j)}}. \tag{1.12}$$

Knowing R_{sym}, the PCA algorithm can then be applied starting at Step 2 in Section 1.2.1. The resulting predictive and residual transformation matrices, $\hat{C}^{sym}$ and $\tilde{C}^{sym}$ respectively, are then used to model both matrices of centers and radii. Then, given an interval testing dataset $\{S^c, S^r\}$, the interval matrix of residuals can be computed as $\left\{\tilde{S}^c, \tilde{S}^r\right\} = \left\{S^c\tilde{C}^{sym}, S^r\tilde{C}^{sym}\right\}$.

2. OBJECTIVES AND METHODS

This section is dedicated to describing the methods used to generate control charts. and the parameters used to interpret the operating conditions of a given process. First, the methods of generation of classical training and testing datasets are defined. Second, the parameters for modeling and fault detection are defined respectively, used to benchmark the monitoring performance of each method. Third, the method used to apply fault detection for the purposes of multi-class classification is defined. Finally, the parameters of aggregation, for the method of interval generation defined in Section 1.3 are defined.

In summary, the objective of this work is to:

1. Develop IPCA methods for process monitoring.
2. Develop a method for accurate multi-class classification using the same principles used for fault detection.
3. Compare the fault detection and data classification performances of IPCA and classical PCA methods for different types of models, including synthetic and real data.

2.1 Data Generation

2.1.1 Training Data

The training data matrix is used to calibrate the PCA model. It is generated either by a mathematical model or taken directly from a process. Training data represents the behavior of the process under normal, or fault-free, operating conditions. If training data is generated using a mathematical model, then there exists two types of usable datasets: the noise-free (X^0) and the noisy (X). The former represents the ideal behavior of the process without any noise. The latter represents the process under more realistic normal operating conditions, and it is generated by adding Gaussian noise to X^0.

The noise matrix is found by first defining the Signal-to-Noise Ratio (SNR), which is the ratio of the standard deviations of the noise-free variables in X^0 to the noise added. The noise matrix is defined as ϵ, and the standard deviations of the variables in ϵ are defined as

$$\sigma_i^{\epsilon} = \frac{\sigma_i^{X^0}}{\sqrt{\text{SNR}}}, \tag{2.1}$$

for $1 \leq i \leq p$, where $\sigma_i^{X^0}$ is the standard deviation of noise-free variable i, and p is the total number of variables in the dataset. The noise added to each variable i is then defined as,

$$\epsilon_i \sim \mathcal{N}(0, \sigma_i^{\epsilon}). \tag{2.2}$$

where $\mathcal{N}(a, b)$ defines a normal (or Gaussian) distribution of mean a and standard deviation b. Finally, the noisy training data matrix is defined as $X = X^0 + \epsilon$.

2.1.2 Testing Data

The testing data matrix is used to benchmark the PCA's monitoring performance, defined as S. It is generated initially in the same way as the training dataset, but a fault of known magnitude is then added to a specific region in the dataset, known as the fault region. There are two types of faults simulated in this paper: change in mean and change in variance. A change in mean is simulated by adding a constant value to the samples in the fault region. A change in variance is simulated by adding Gaussian noise with a significantly lower SNR (e.g. SNR$= 1$) to the samples in the fault region. Gaussian noise has zero mean, so negligible changes are expected in the mean value of the samples.

For example, for a change in mean, given a 100×5 matrix S, if we want to simulate a fault on variable 1 for the sample fault region $[50, 51, \ldots, 75]$, then

$$S_{(1)}[50, 51, \ldots, 75] = S_{(1)}[50, 51, \ldots, 75] + \sigma_1^S, \tag{2.3}$$

where σ_1^S is the standard deviation of variable 1 in S before a fault is added. Typically, a fault of 1σ is considered to be small and can sometimes go undetected, while a fault of 3σ is considered to be severe, and must be detected by the fault detector. It is not practical to add a large mean shift to avoid biasing the results, so a change in mean fault of 1σ is used in the paper. In addition, the fault is simulated for a randomly selected variable at each iteration of the Monte Carlo simulation, and for a randomly selected, but fixed in size, fault region.

2.1.3 Interval Data

The method of interval generation used in this thesis is defined in Section 1.3, where blocks of classical samples are aggregated into single intervals each, such that the mean of each block of samples is taken as the centers and their standard deviation as the radii. The main idea behind this method of aggregation is to try and minimize the interference of noise in the monitoring process. The noise tested in this work is Gaussian and zero mean. As a result, when taking the mean of each sample block as its respective interval's center, the noise is removed. Intuitively, we can now assume that noise will have a larger influence on the radii rather than the centers, since the former takes the standard deviation of the sample blocks. This property allows for IPCA methods to differentiate the types of faults. Changes in mean will very likely be noticeable in the interval centers, while changes in the variance, caused by a higher magnitude of noise, will more likely to be noticeable in the interval radii.

Moreover, there are two more parameters to consider when evaluating the fault detection performances: the number of samples aggregated per interval m (i.e. the number of samples in each block), and the interval width β. In the following sections, the modeling and fault detection performances are explored for different values of m and β in order to optimize the method of interval generation.

2.2 Modeling and Fault Detection

Fault detection can be performed in two steps: modeling and evaluation of residuals. Modeling using PCA seeks to find the optimal model that maximizes the amount of noise being filtered, while minimizing the amount of valuable data being discarded. One way to do that would be by measuring the proximity of the predicted training dataset $\hat{X}$ to the noise-free training dataset X^0, defined as the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\frac{1}{p}\sum_{j=1}^{p}(\hat{X}_j[i] - X_j[i])^2), \tag{2.4}$$

where n and p are respectively the number of samples and variables in the dataset. MSE is calculated and plotted for different numbers of retained principal components. The optimal number of retained PCs is at the plot's global minimum. However, as mentioned in Section 1.2.1, if the noise-free training dataset is not available, the noisy training dataset X is used instead. In that case, the optimal number of PCs to retain is point beyond which the slope of the MSE plot is approximately zero, or horizontal.

After selecting the optimal number of PCs to retain and finding the matrix of residuals for a testing dataset S, the model residuals are then evaluated by two detection statistics: false alarm rate/probability (FAR) and detection rate/probability (DR). The false alarm rate is the average percentage of samples outside the fault region that were wrongfully declared as faults. The detection rate is the average percentage of samples inside the fault region that were rightfully declared as faults. It is desirable to maximize the detection rate, for the same false alarm rate, to achieve better fault detection. In this work, the Q statistic will be used to evaluate that.

The Q-statistics, also known as the Squared Prediction Error (SPE), for an $n \times p$ residual matrix $\tilde{X}$ is defined as,

$$\text{Q}[i] = \sum_{j=1}^{p} \left(\tilde{X}_j[i] \right)^2, \tag{2.5}$$

for $1 \leq i \leq n$. For the IPCA methods, the Q-statistics of the classical centers and radii datasets are calculated and evaluated separately.

The next step is to find the threshold value, beyond which a fault will be declared. First, we find an empirical distribution function (EDF) of the training data's Q-statistics. The EDF is an estimate of the cumulative distribution function for a set of discrete values. The threshold, defined as γ, is found using the desired false alarm rate α. The lower the value of α, the more confidence the PCA model will have in the training dataset, since it is desirable that smaller number of false alarms are declared. A higher value of α is usually chosen for training datasets with more noise, or for more conservative standards of operating conditions. For more detailed information, please refer to [22].

Afterwards, we find the Q-statistics of the testing data's residuals $\tilde{S}$ and compare them to γ to find the false alarm and detection rates. An example of a Q-statistics chart for testing data is shown in Figure 2.1. The left half of the samples in the testing data contains no faults, so detections (i.e. points at which the Q-statistics cross the threshold) are treated as false alarms. The right half of the samples in the chart have a fault simulated, called the "fault region", so the Q-statistics which fail to cross the threshold are considered to be missed detections. It is desirable to maximize the detection rate (i.e. minimize the number of missed detections) and minimize the false alarm rate for a better fault detector.

A simplified block diagram of a fault detection unit (FDU) using PCA and Q-statistics is shown in Figure 2.2.

For IPCA, two Q-charts for the centers and radii are separately generated, and the results for each are recorded. The results are typically displayed in the form of a ROC (Receiver Operating Characteristic) curve, which plots the detection rate versus the false

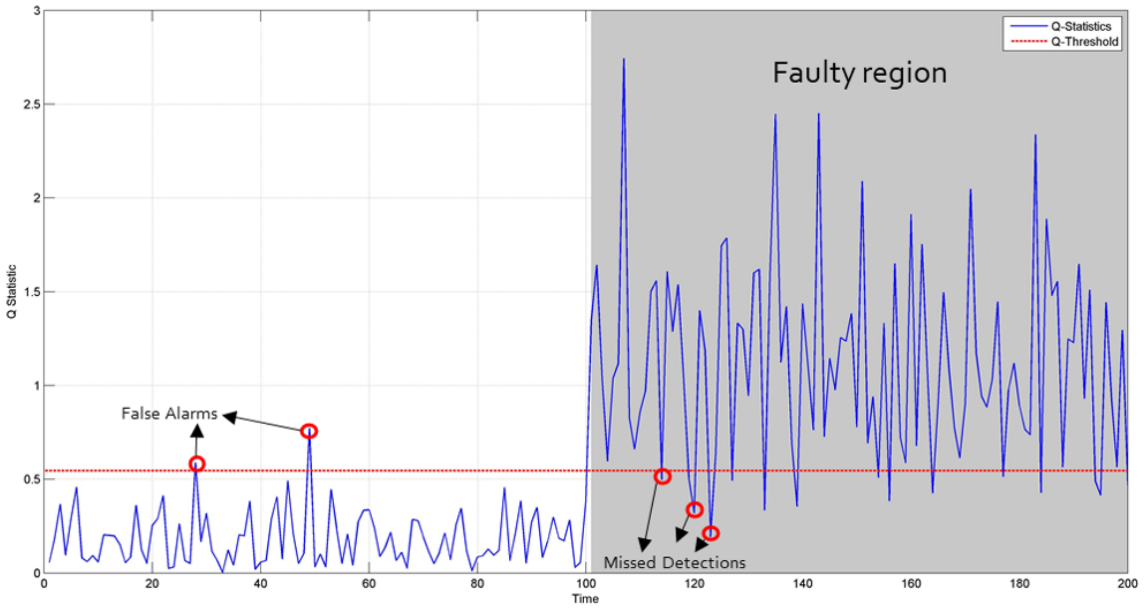


Figure 2.1: Example of a Q-chart for testing data. The red horizontal line is the Q-threshold, γ. The blue line is the Q-statistics of the testing data.

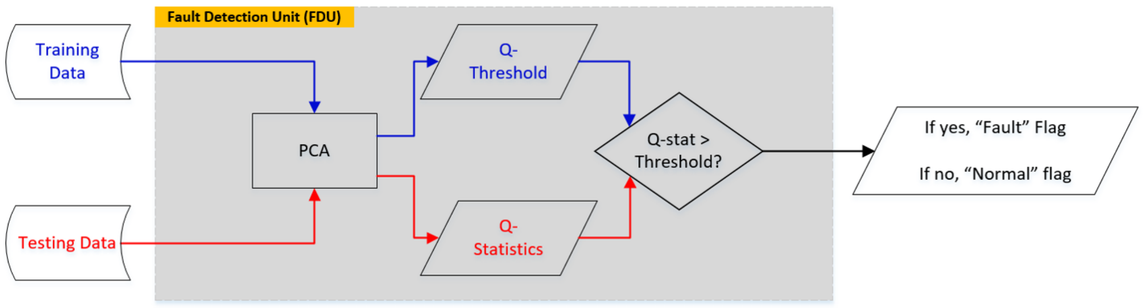


Figure 2.2: Block diagram of a fault detection unit (FDU) using PCA and Q-statistics. Training data is used to find the Q-threshold, which is then compared to the testing data's Q-statistics.

alarm rate. The larger the area under the ROC curve, the more robust the method is as a fault detector.

2.3 Data Classification

Data classification is widely used in data mining applications to tag data so that it can be found quickly and efficiently. In machine learning applications, high-dimensional datasets may also have a massive number of samples, which negatively impacts the execution time for classical PCA. Moreover, even though the fault detection performance of classical PCA typically improves as the number of samples increases, it has a limit beyond which more samples cause no change in its performance. An example of a dataset using binary classification (i.e. only two classes of data) is defined as follows,

$$X = \begin{array}{c} \\ \text{Sample } 1 \\ \text{Sample } 2 \\ \text{Sample } 3 \\ \text{Sample } 4 \\ \text{Sample } 5 \\ \text{Sample } 6 \\ \text{Sample } 7 \\ \text{Sample } 8 \\ \text{Sample } 9 \\ \vdots \\ \text{Sample } n \end{array} \begin{array}{c} \begin{array}{cccccc} \text{Var. } 1 & \text{Var. } 2 & \text{Var. } 3 & \text{Var. } 4 & \text{Var. } 5 & \text{Var. } 6 \end{array} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & x_{4,6} \\ x_{5,1} & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} \\ x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} \\ x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} \\ x_{8,1} & x_{8,2} & x_{8,3} & x_{8,4} & x_{8,5} & x_{8,6} \\ x_{9,1} & x_{9,2} & x_{9,3} & x_{9,4} & x_{9,5} & x_{9,6} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n,1} & x_{n,2} & x_{n,3} & x_{n,4} & x_{n,5} & x_{n,6} \end{bmatrix} \end{array} \begin{array}{c} \\ \text{Class A} \\ \text{Class A} \\ \text{Class B} \\ \text{Class B} \\ \text{Class A} \\ \text{Class A} \\ \text{Class B} \\ \text{Class B} \\ \text{Class A} \\ \vdots \\ \text{Class B} \end{array} . \tag{2.6}$$

Given the FDU shown in Figure 2.2, a block diagram of the application of fault detection for the purposes of binary data classification is shown in Figure 2.3. Two FDU's are generated for each class using their respective training datasets. It is desired to find the class of the samples in the testing dataset. If a "normal" flag is raised by the FDU, the testing sample is assumed to belong to the FDU's respective class.

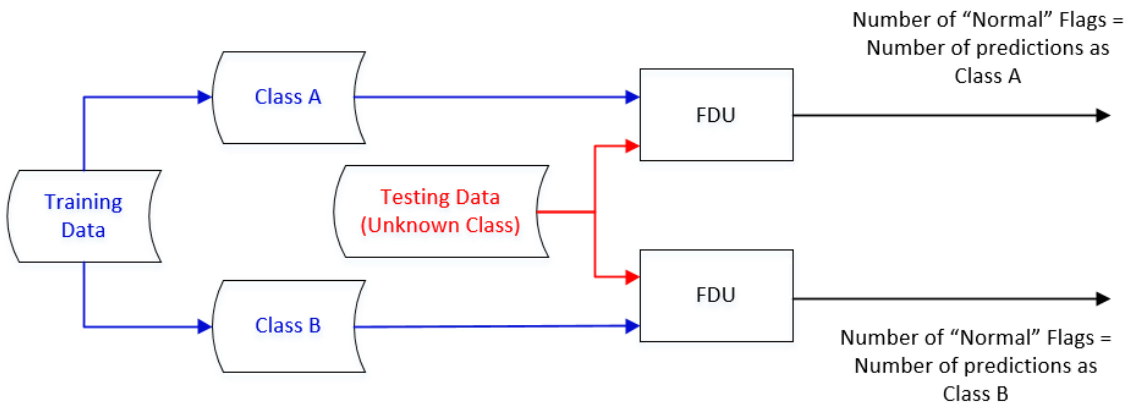

Number of "Normal" Flags = Number of predictions as Class A
Class A
FDU
Training Data
Testing Data (Unknown Class)
Class B
FDU
Number of "Normal" Flags = Number of predictions as Class B

Figure 2.3: Block diagram of binary classification using FDUs.

The number of normal flags on each FDU are then recorded in the square confusion matrix, which details the performance of the classifier used and is in the order of the number of classes in the dataset. In this simple example of binary classification, the confusion matrix has an order of two. As expected, it is desirable to maximize the number of predictions that match the actual class of the testing data (i.e. the diagonal entries), and minimize the predictions that differ from the actual class of the testing dataset (i.e. the off-diagonal entries). The average values of each entry in the confusion matrix can be calculated from the fault detection metrics, FAR and DR, as seen in Table 2.1.

For a small FAR, the higher the DR, the smaller the off-diagonal entries. This means that the fault detector is better at classifying the testing samples. The parameter used to benchmark the classifiers' performances is precision, which is the ratio of true positive predictions to the total number of predictions [23]. In other words, precision is the ratio of the number of times a class was correctly predicted to the total number of predictions made for said class. It is desirable for precision to be closer to 1 for all classes for a more robust classifier.

		Predicted (i.e. PCA Model Output)	
		Class A	Class B
Actual (i.e. Testing Data Class)	Class A	$(1 - FAR_A) \cdot n_A$	$(1 - DR_B) \cdot n_A$
	Class B	$(1 - DR_A) \cdot n_B$	$(1 - FAR_B) \cdot n_B$
	Precision:	$\left(1 + \frac{(1-DR_A) \cdot n_B}{(1-FAR_A) \cdot n_A}\right)^{-1}$	$\left(1 + \frac{(1-DR_B) \cdot n_A}{(1-FAR_B) \cdot n_B}\right)^{-1}$

Table 2.1: Example of a confusion matrix in binary classification. n_A and n_B are the number of samples in the testing dataset belonging to classes A and B respectively.

Note that it is necessary for n_A and n_B to be comparable in magnitude in order for the precision statistic not to be misleading. For example, if 95% of the testing samples are taken from class A, then even a precision of 90% is not good, even if it seems otherwise. That is because if you set a dummy classifier to always predict the samples as belonging to class A, then the precision will be 95%. The dummy classifier is obviously not usable in practice, yet it has a higher precision than the non-dummy one. Therefore, if the samples are split evenly, then the precision can be a useful and accurate depiction of the classifiers' performances.

3. RESULTS

In this section, the results for six case studies are presented. The first three case studies will evaluate the fault detection performance of all methods. The first case study is a synthetic linear example where the generated interval centers have a limited range of values due to the variables' Gaussian distribution, while the second case study is a synthetic linear example where the interval centers have a wider range of values due to the square or sinusoidal variables. The third case study is a synthetic non-linear example. Despite there being many data analysis tools available for non-linear models, like kernel PCA, it is still useful to evaluate the robustness of linear methods since they are significantly less expensive computationally, especially as the number of samples or the dimensionality increases.

Monte Carlo simulations of 500 iterations carried out for the first three case studies, and the average results are plotted for all methods. This is crucial in order to properly stress-test a model under different conditions and to ensure that the results are not misleading. The results for IPCA methods are split into two parts: centers and radii. CIPCA is represented in red, MRIPCA is represented in green, and SCIPCA is represented in black. Classical PCA is represented as blue, and it is taken as the control method for both parts of the graphs. The modeling and fault detection performances are evaluated at different numbers of samples aggregated per interval (m), and at different interval width coefficients (β) in order to find the optimal value for each. The interval width is defined as $\beta\sigma$, where σ is the standard deviation of the classical samples being aggregated for the interval. Moreover, the optimal number of retained principal components is selected using the MSE plot. The ROC curves for all aforementioned methods is shown for each case study, in order to observe the best fault detector for different types of faults.

Afterwards, the last three case studies are based on real data examples, and are used to evaluate the data classification performance of all methods. The fourth case study discusses the problem of sensor drift, and its effect on the performance of classifiers used in the recognition of different gases in real-time operations [24, 25]. Sensor drift is the gradual and unpredictable change in a sensor's response to the same chemical conditions over time, as a result of prolonged chemical interactions with the sensor, or due to changes in external operating conditions, such as temperature and humidity, or due to measurement noise caused by condensation in the tubes.

The fifth case study discusses the lack of a standard dataset for physical activity monitoring [26, 27]. The authors used three inertial measurement units (IMUs) and a heart rate monitor to collect data on different physical activities from nine subjects. The IMUs consist of a 3D accelerometer to measure acceleration, a 3D gyroscope to measure angular velocity, and a 3D magnetometer to measure earth's magnetism at a sampling rate of 100 Hz each. The IMUs are positioned on the wrist, chest and the dominant side's ankle. Moreover, a heart rate monitor is used, which has a sampling rate of $\approx$9 Hz.

Finally, the sixth case study compared properties of the brain's electrical activity in different regions and at different states, including seizure activity, using an EEG [28, 29]. It is crucial to be able to differentiate between the different states accurately and quickly in order to be allow for a swift medical response in the case of a seizure. For all three real examples, a precision curve is shown for both the centers and radii, with classical PCA taken as control. The curve shows the classification precision of each method for each category, where it is desirable to maximize them for a better classifier.

3.1 Case Study 1: Linear Synthetic Example

In the first case study, data is generated using the following linear model:

$$\begin{aligned}
X^0_{(1)} &\sim \mathcal{N}(5, 1) \\
X^0_{(2)} &\sim \mathcal{N}(-2, 0.2) \\
X^0_{(3)} &\sim \mathcal{N}(120, 2) \\
X^0_{(4)} &\sim \mathcal{N}(0, 1) \\
X^0_{(5)} &= \frac{1}{2} \cdot (X^0_{(1)} + X^0_{(2)}) \\
X^0_{(6)} &= \frac{1}{2} \cdot (X^0_{(1)} - X^0_{(2)}) \\
X^0_{(7)} &= \frac{1}{4} \cdot X^0_{(1)} + \frac{1}{10} \cdot X^0_{(2)} + \frac{1}{3} \cdot X^0_{(4)}
\end{aligned} \tag{3.1}$$

Variables $\{1, 2, 3, 4\}$ are independent, and variables $\{5, 6, 7\}$ are dependent. The modeling performances, represented using the MSE for the noise-free training dataset (X^0), are plotted for different values of m and β in Figure 3.1 and Figure 3.2 respectively. Furthermore, the fault detection performances, represented using the detection probability for a fixed false alarm probability of 5%, are plotted for different values of m and β for a change in mean fault in Figure 3.3 and Figure 3.5, and for a change in variance fault in Figure 3.4 and Figure 3.6 respectively.

Figures 3.1 and 3.2 show that m and β have a negligible effect on the MSE of all IPCA methods, unlike the fault detection results. For a change in mean fault, Figure 3.3 shows that the detection probability of the interval centers slightly improves for higher values of m, while the plots for the interval radii remain near horizontal. Conversely, for a change in variance fault, Figure 3.4 shows that the detection probability of the interval radii significantly improves for higher values of m, while the plots for the interval centers remain relatively unchanged. For both cases, the plots plateau at $m = 100$, with minimal change for higher values of m. On the other hand, the detection probability for both types of faults did not change for different values of β, as seen in Figures 3.5 and 3.6.

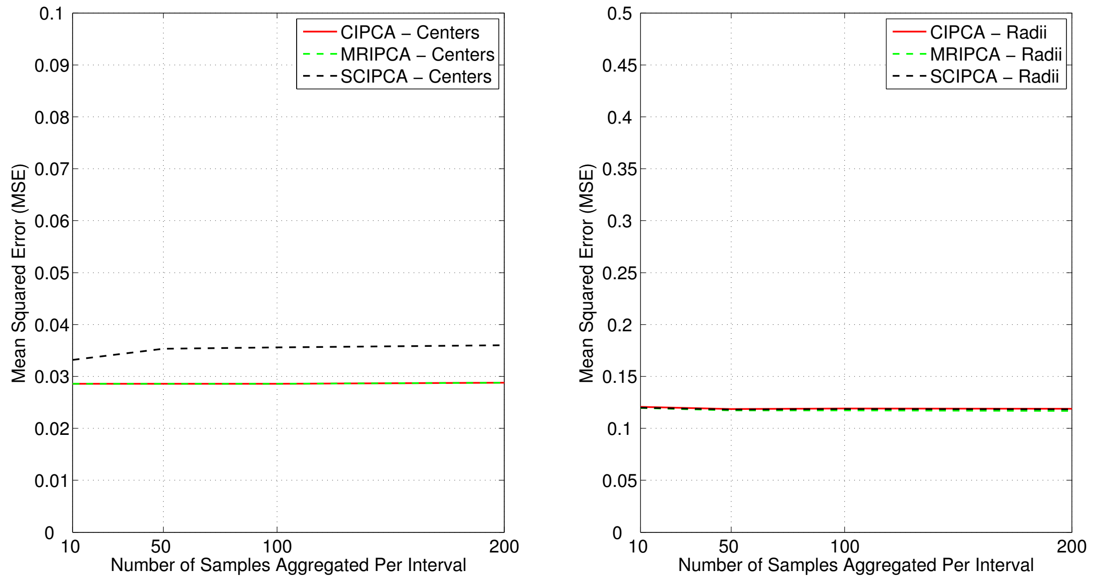


Figure 3.1: MSE for X^0 vs m for the interval centers (left) and radii (right) - case study 1.

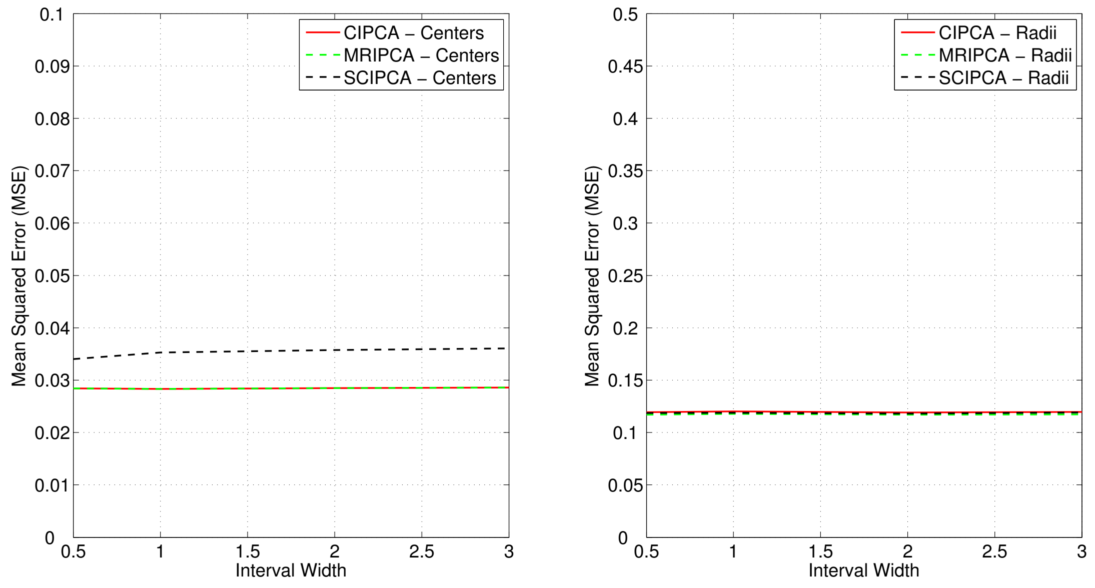


Figure 3.2: MSE for X^0 vs β for the interval centers (left) and radii (right) - case study 1.

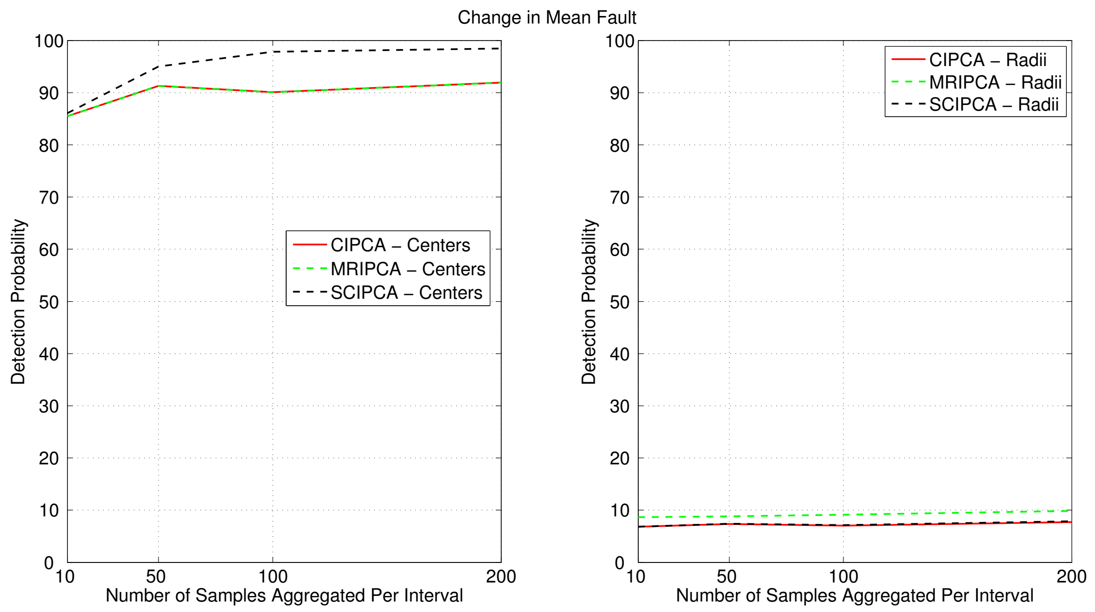


Figure 3.3: Detection probability vs m for 5% false alarm probability - change in mean fault - case study 1.

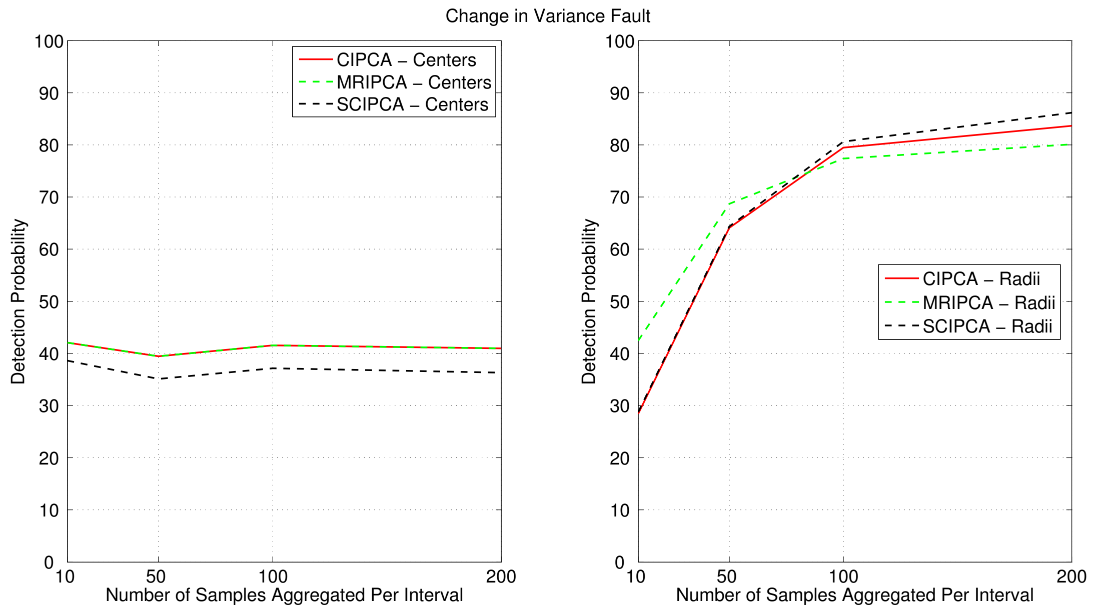


Figure 3.4: Detection probability vs m for 5% false alarm probability - change in variance fault - case study 1.

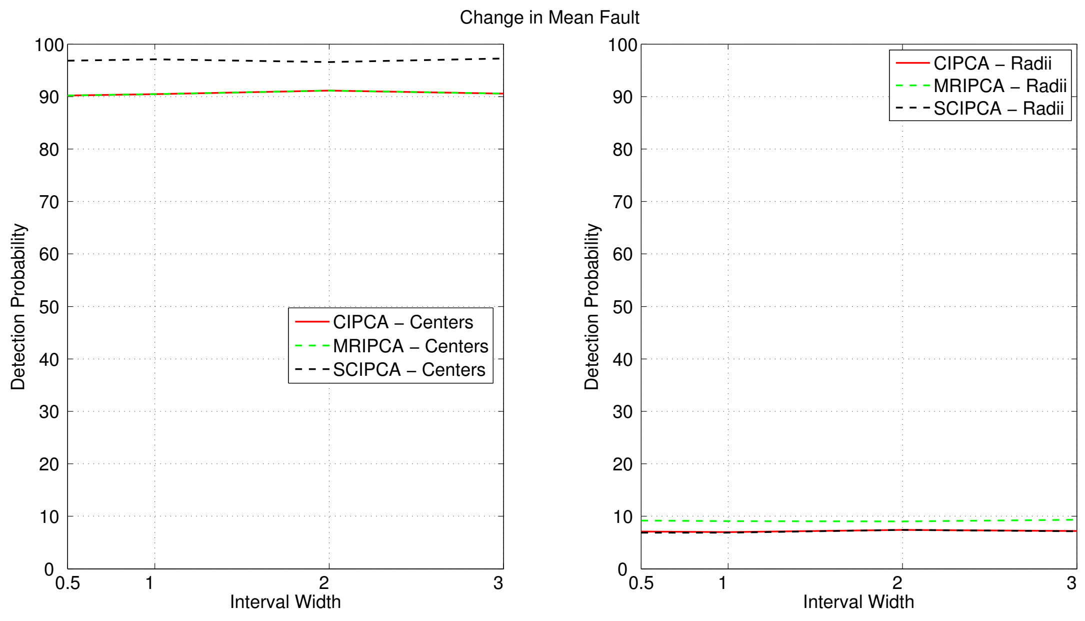


Figure 3.5: Detection probability vs β for 5% false alarm probability - change in mean fault - case study 1.

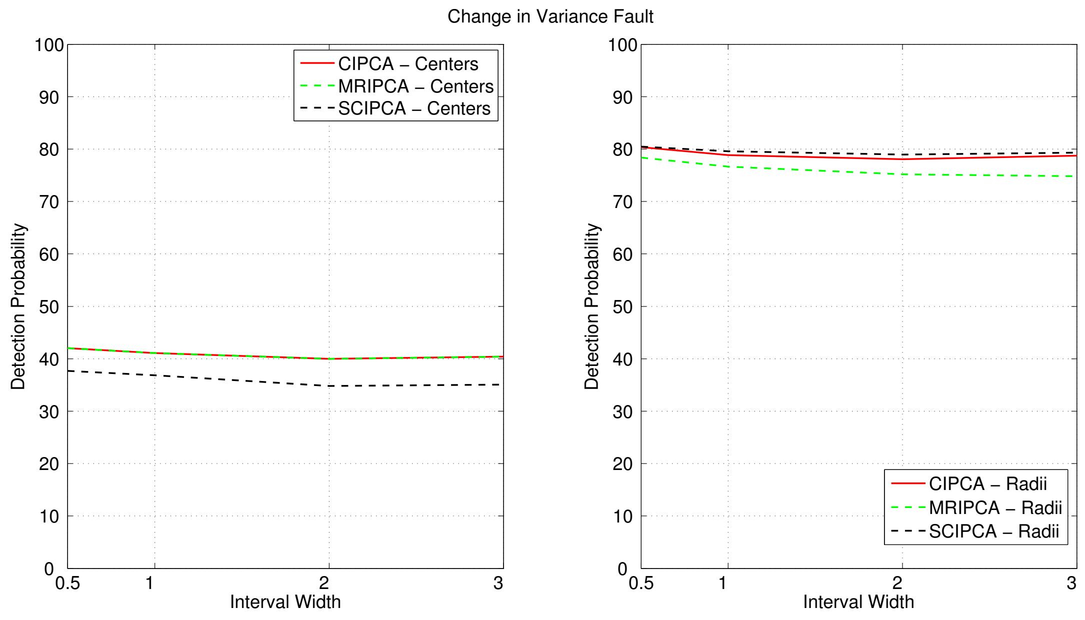


Figure 3.6: Detection probability vs β for 5% false alarm probability - change in variance fault - case study 1.

As a result, m is set to 100, while β is set to 1 for all IPCA methods. The data modeling results are shown in Figure 3.7. The optimal number of principal components to retain is 4 for all methods, since it is the point at which the curves begin to plateau.

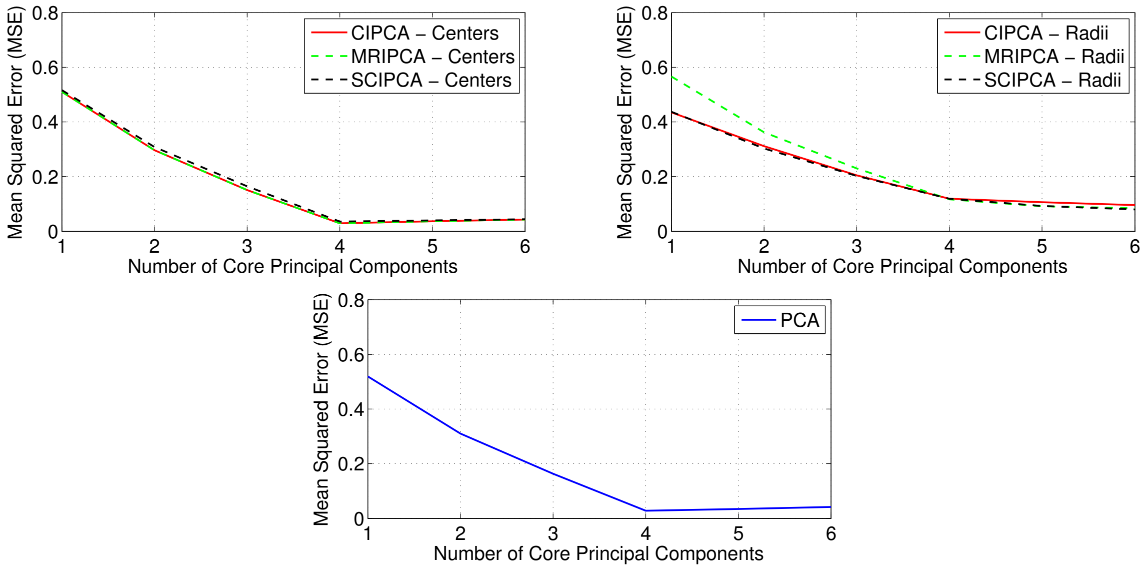


Figure 3.7: MSE for IPCA methods versus the number of retained principal components for the interval centers (top left) and interval radii (top right), and for classical PCA (bottom) - case study 1.

Knowing the optimal number of PCs to retain, the fault detection performances of all IPCA methods are evaluated, in comparison to that of classical PCA. Figure 3.8 shows the results for a change in mean fault, and Figure 3.9 shows the results for a change in variance fault.

In both cases, all IPCA methods were better fault detectors than classical PCA, with SCIPCA having the best performance. In addition, the results show that the interval centers were more capable than the interval radii of detecting changes in mean, where a linear plot for the interval radii indicates almost no detection. Conversely, the interval radii were more capable of detecting changes in variance, where the interval centers of all IPCA methods had a nearly identical performance to that of classical PCA. Examples of single iterations of the Monte Carlo simulation, for a change in mean and a change in variance, are shown in Figure 3.10 and Figure 3.11 respectively.

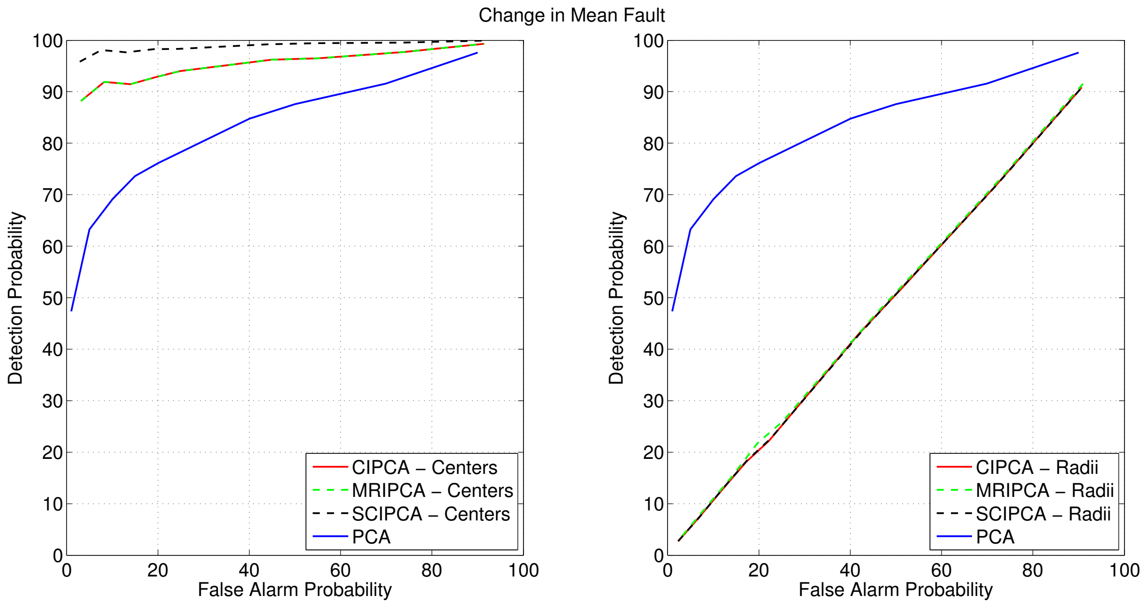


Figure 3.8: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in mean fault - case study 1.

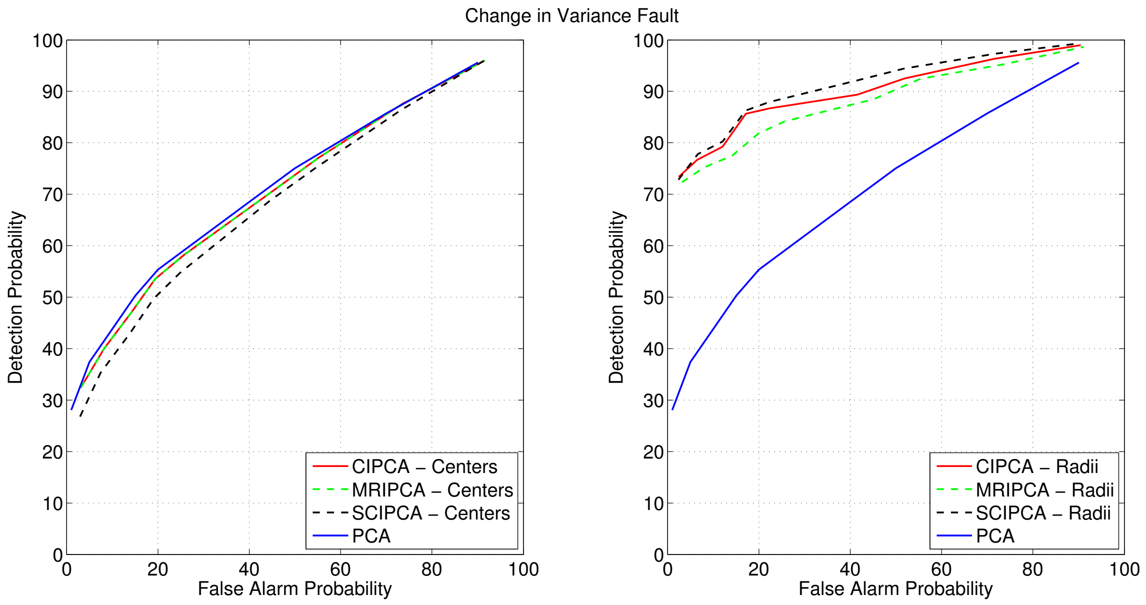


Figure 3.9: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in variance fault - case study 1.

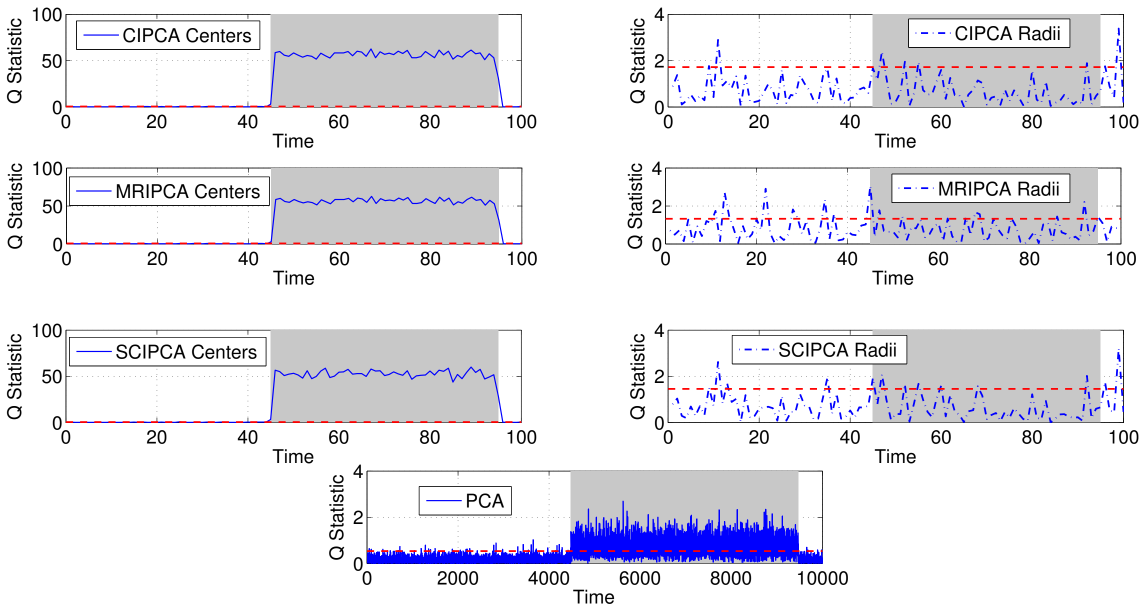


Figure 3.10: Q-Statistics for the interval centers (left) and radii (right) of all IPCA methods and PCA (bottom) - change in mean fault - case study 1.

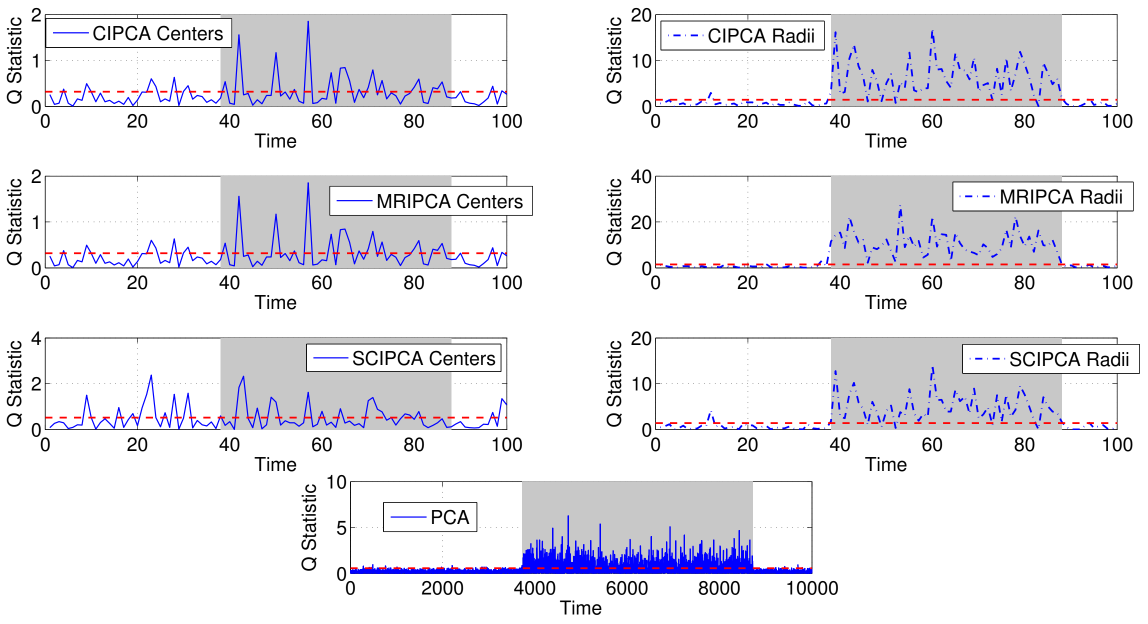


Figure 3.11: Q-Statistics for the interval centers (left) and radii (right) of all IPCA methods and PCA (bottom) - change in variance fault - case study 1.

3.2 Case Study 2: Linear Synthetic Example

In the second case study, data is generated using the following linear model:

$$\begin{aligned}
X^0_{(1)} &= 5 + A_1 \sin(t/T_1), \quad \text{where } A_1 \sim \mathcal{U}(0.8, 1.2) \\
X^0_{(2)} &= -2 + A_2 \cos(t/T_2), \quad \text{where } A_2 \sim \mathcal{U}(0.16, 0.24) \\
X^0_{(3)} &= 120 + A_3 \text{SQUARE}(t, D), \\
X^0_{(4)} &= \frac{1}{2} \cdot (X^0_{(1)} + X^0_{(2)}) \\
X^0_{(5)} &= \frac{1}{2} \cdot (X^0_{(1)} - X^0_{(2)}) \\
X^0_{(6)} &= \frac{1}{4} \cdot X^0_{(1)} + \frac{1}{10} \cdot X^0_{(2)} + \frac{1}{3} \cdot X^0_{(3)}
\end{aligned} \tag{3.2}$$

where T_1 and T_2 are the time period of the first and second variables respectively, and D is the duty cycle of the square waveform, and are randomly generated using the uniform distributions $\mathcal{U}(0.6, 0.9)$, $\mathcal{U}(0.3, 0.5)$, and $\mathcal{U}(10, 80)$ respectively. A_1, A_2 and A_3 define the amplitudes of the first three variables respectively.

Variables $\{1, 2, 3\}$ are independent, and variables $\{4, 5, 6\}$ are dependent. The MSE for X^0 is plotted for different values of m and β in Figure 3.12 and Figure 3.13 respectively, and the detection probability, for a fixed false alarm probability of 5%, is plotted for different values of m and β for a change in mean fault in Figure 3.14 and Figure 3.16, and for a change in variance fault in Figure 3.15 and Figure 3.17.

Figure 3.12 shows that the MSE slightly decreases for larger values of m, plateauing at $m = 50$ for both the centers and radii, while Figure 3.13 shows that β has a negligible effect on the MSE of all IPCA methods. On the other hand, the fault detection results in Figure 3.14 show that, for a change in mean fault, there was a minimal change in the detection probability for both the centers and radii. However, for a change in variance fault, Figure 3.15 shows that the detection probability of the interval radii improves for higher values of m, plateauing at $m = 100$, while the plots for the interval centers remain relatively unchanged for CIPCA and MRIPCA, and decrease for SCIPCA.

The detection probability for a change in mean did not change for different values of β, as seen in Figures 3.16. On the other hand, for a change in variance fault, the detection probability decreases at higher values of $beta$ for the interval centers of SCIPCA only, while remaining fixed in other cases.

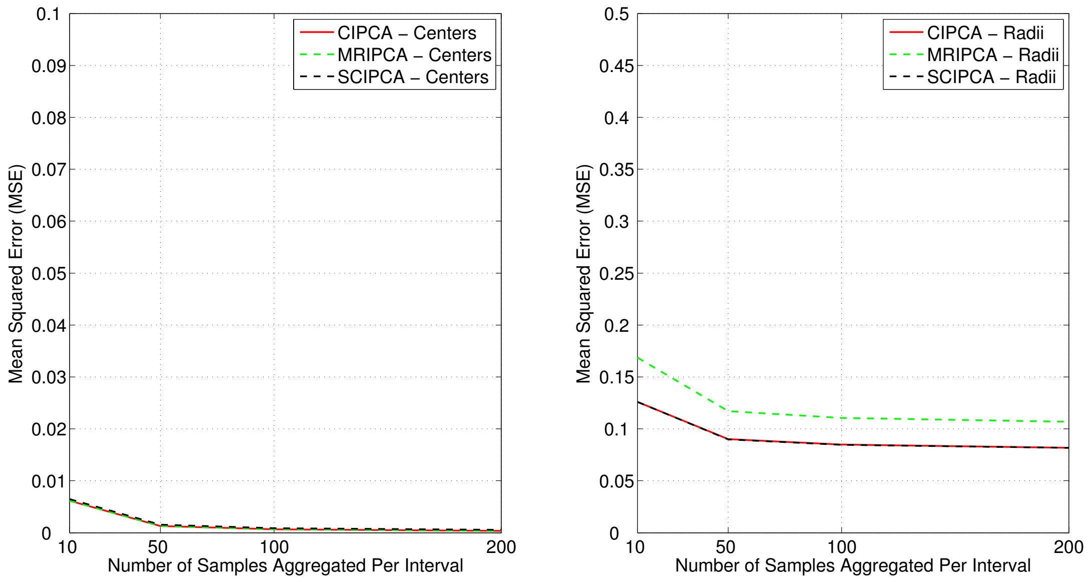


Figure 3.12: MSE for X^0 vs m for the interval centers (left) and radii (right) - case study 2.

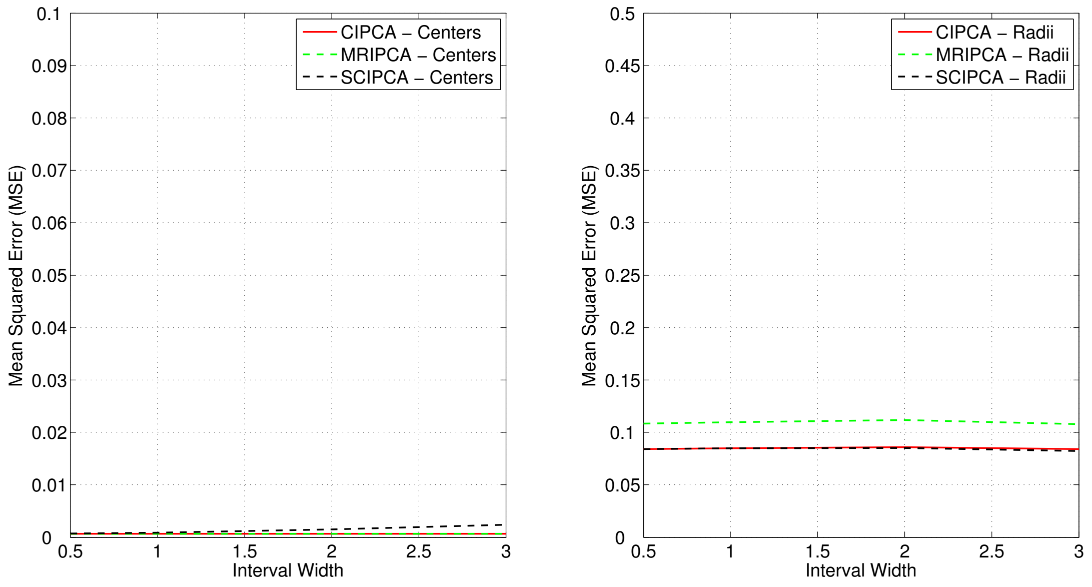


Figure 3.13: MSE for X^0 vs β for the interval centers (left) and radii (right) - case study 2.

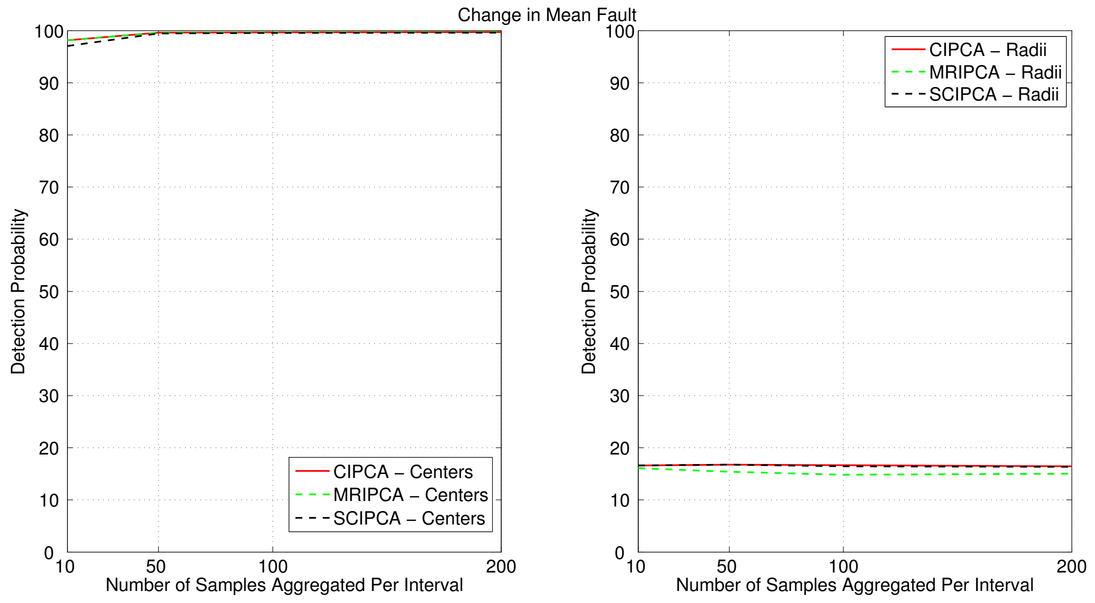


Figure 3.14: Detection probability vs m for 5% false alarm probability - change in mean fault - case study 2.

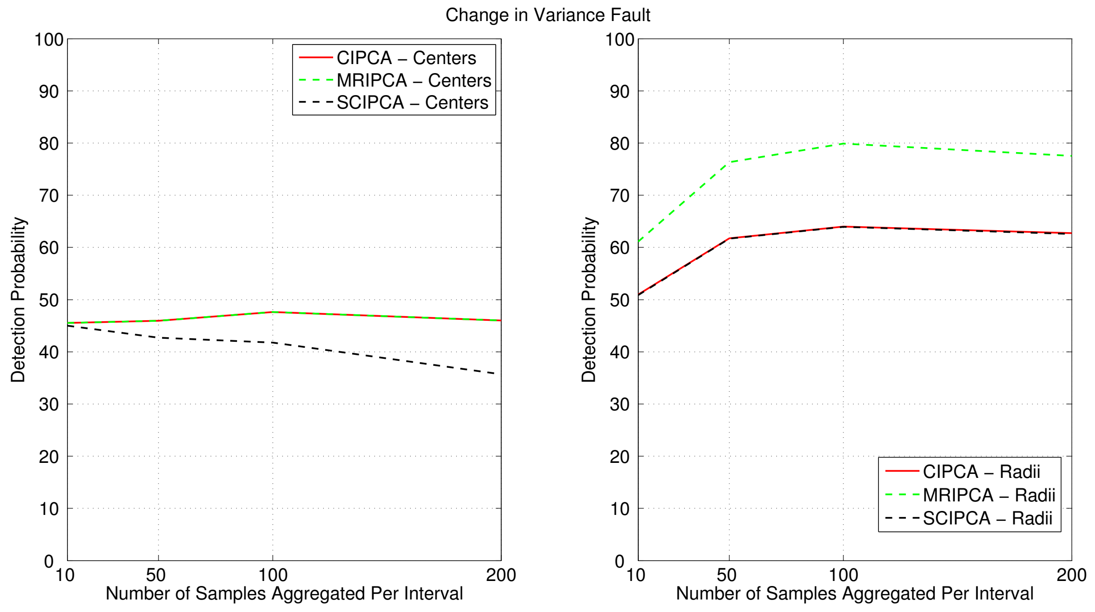


Figure 3.15: Detection probability vs m for 5% false alarm probability - change in variance fault - case study 2.

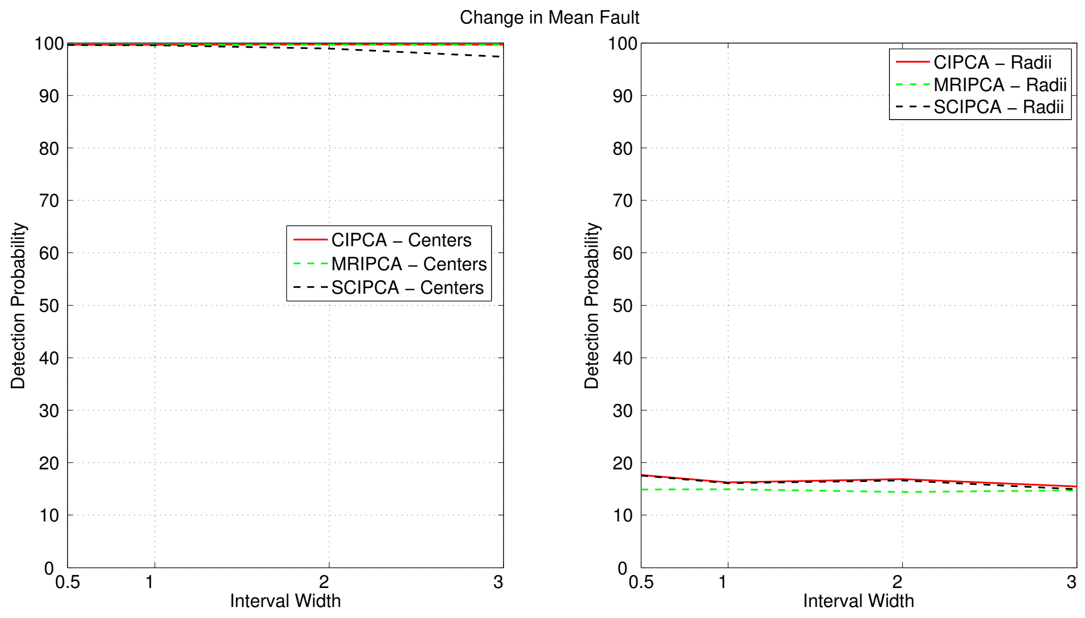


Figure 3.16: Detection probability vs β for 5% false alarm probability - change in mean fault - case study 2.

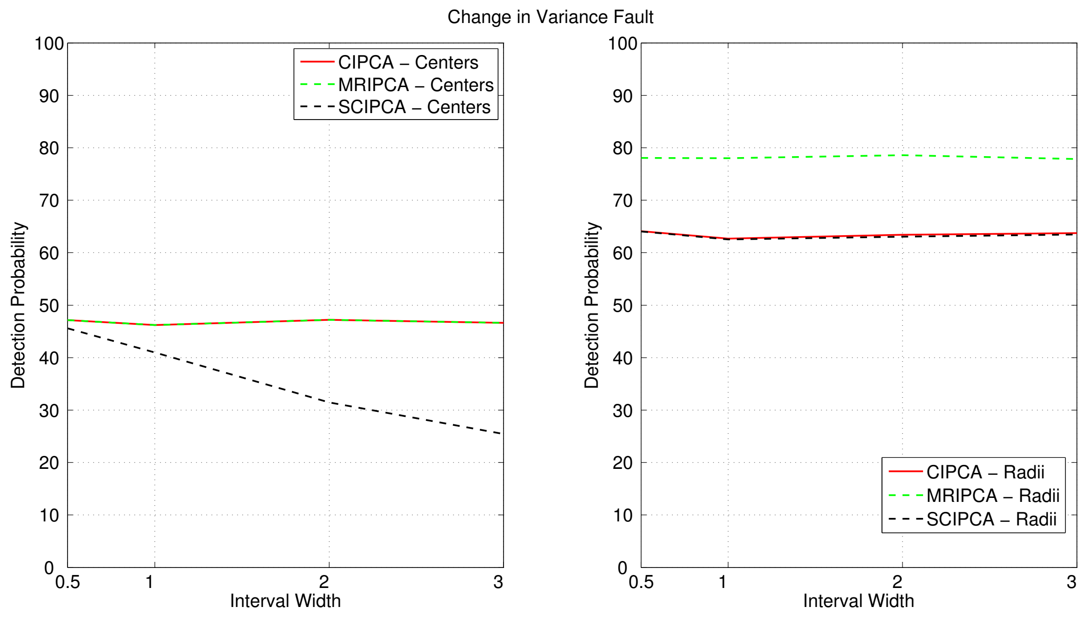


Figure 3.17: Detection probability vs β for 5% false alarm probability - change in variance fault - case study 2.

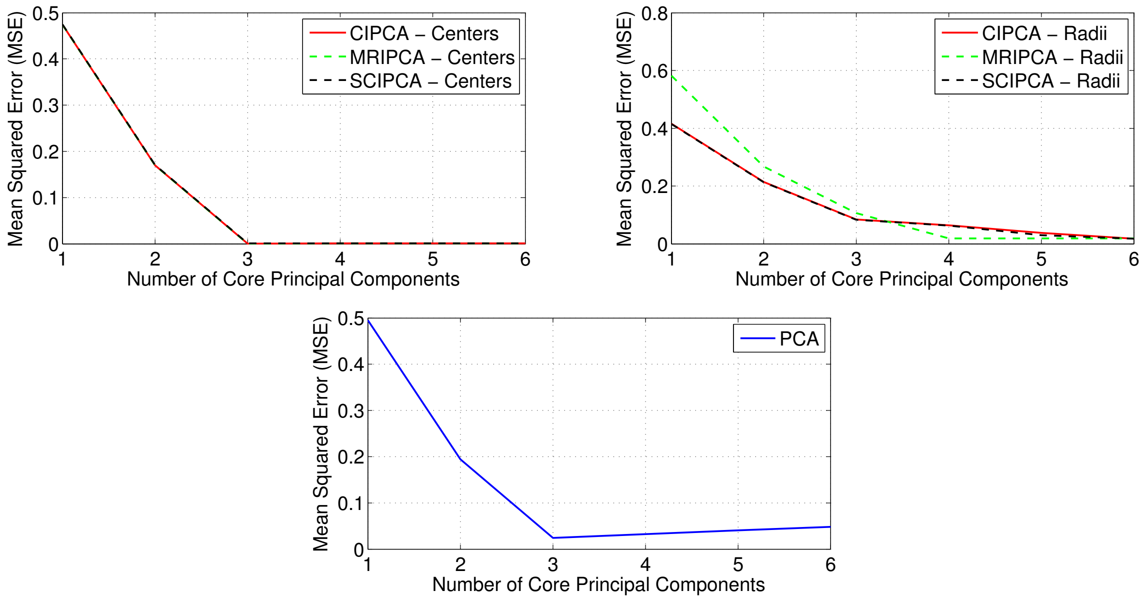


Figure 3.18: MSE for IPCA methods versus the number of retained principal components for the interval centers (top left) and interval radii (top right), and for classical PCA (bottom) - case study 2.

m is set to 100, while β is set to 1 for all IPCA methods. The data modeling results are shown in Figure 3.18. The optimal number of retained principal components is 3. Figure 3.19 shows the results for a change in mean fault, and Figure 3.20 shows the results for a change in variance fault.

In both cases, all IPCA methods were better fault detectors than classical PCA. Both the MRIPCA and CIPCA had the best performance in the case of a change in mean fault, and MRIPCA had the best performance for a change in variance fault. In addition, similar to the first case study, the results show that the interval centers were able to detect changes in mean, and interval radii were able to detect changes in variance.

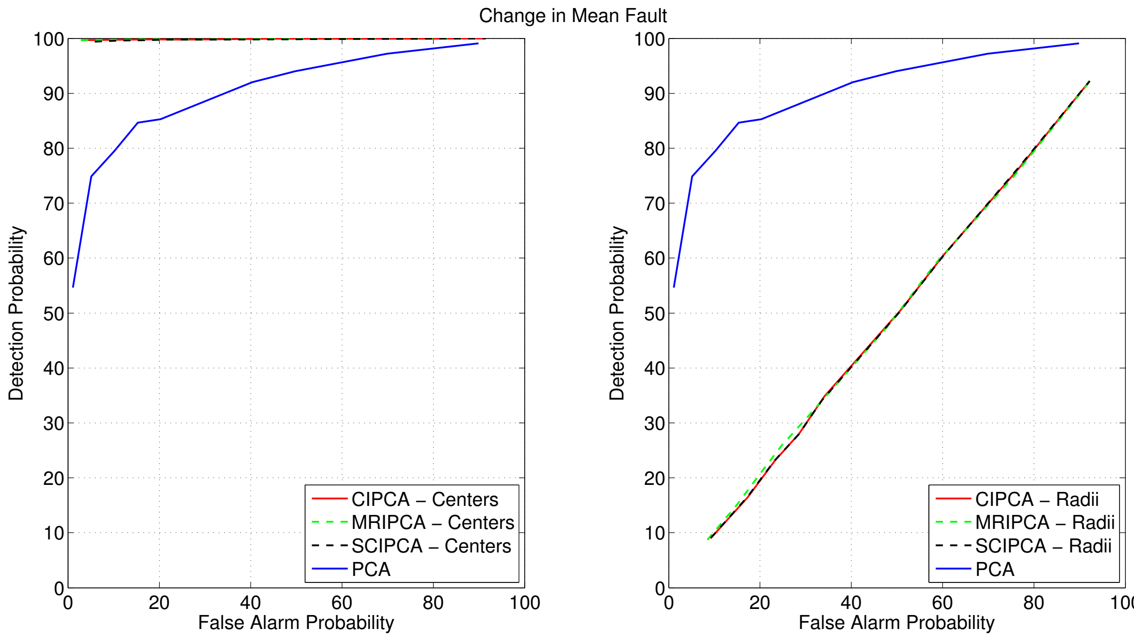


Figure 3.19: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in mean fault - case study 2.

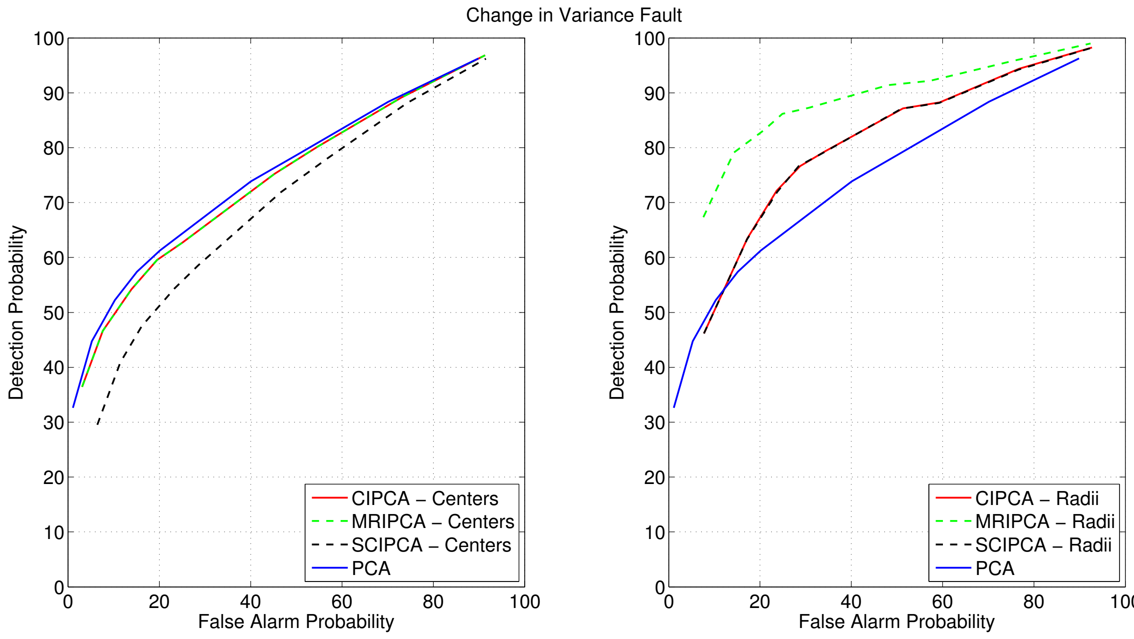


Figure 3.20: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in variance fault - case study 2.

3.3 Case Study 3: Non-Linear Synthetic Example

In the third case study, data is generated using the following non-linear model:

$$\begin{aligned}
X^0_{(1)} &\sim \chi^2(4) \\
X^0_{(2)} &\sim \mathcal{N}(0, 0.6) \\
X^0_{(3)} &= A_1 \sin(t/T_1) \\
X^0_{(4)} &= (X^0_{(2)})^2 + 5X^0_{(2)} \\
X^0_{(5)} &= 2\exp(-X^0_{(1)}) \\
X^0_{(6)} &= (-X^0_{(3)})^2 + \frac{(A_1 \cos(t/T_1))^2}{\sqrt{-X^0_{(1)}}}
\end{aligned} \tag{3.3}$$

where A_1 and T_1 are the amplitude and time period of the third variable, generated using the uniform distributions $\mathcal{U}(10, 11)$ and $\mathcal{U}(0.75, 1)$ respectively.

Variables $\{1, 2, 3\}$ are independent, and variables $\{4, 5, 6\}$ are dependent. The MSE for X^0 is plotted for different values of m and β in Figure 3.21 and Figure 3.22 respectively, and the detection probability, for a fixed false alarm probability of 5%, is plotted for different values of m and β for a change in mean fault in Figure 3.23 and Figure 3.25, and for a change in variance fault in Figure 3.24 and Figure 3.26. The results show that both m and β had a limited effect on the modeling and fault detection performance in all cases except for a change in variance. Figure 3.24 shows that the detection probability significantly improves for higher values of m.

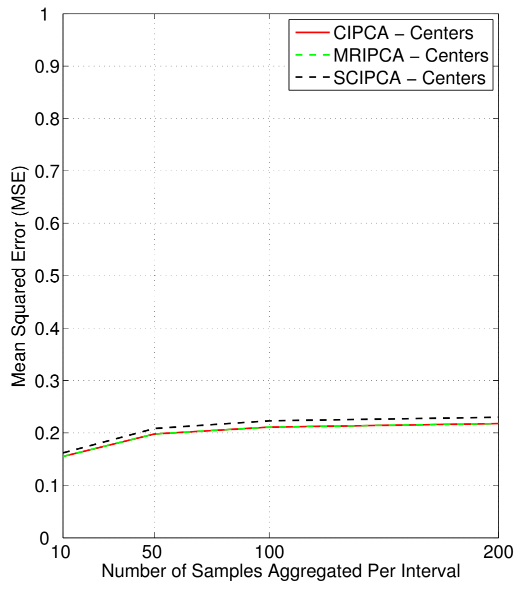


Figure 3.21: MSE for X^0 vs m for the interval centers (left) and radii (right) - case study 3.

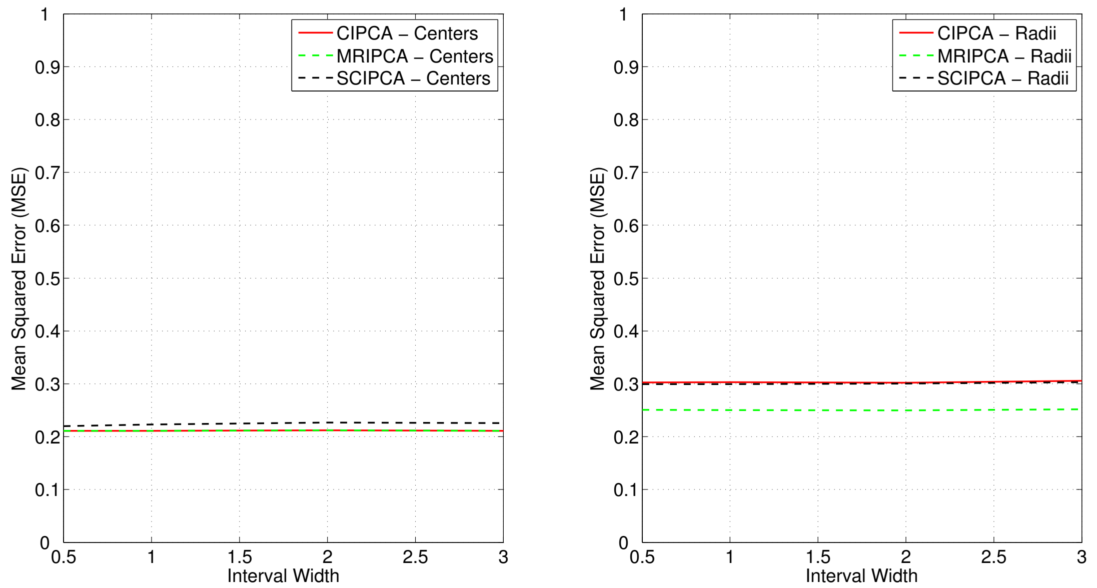


Figure 3.22: MSE for X^0 vs β for the interval centers (left) and radii (right) - case study 3.

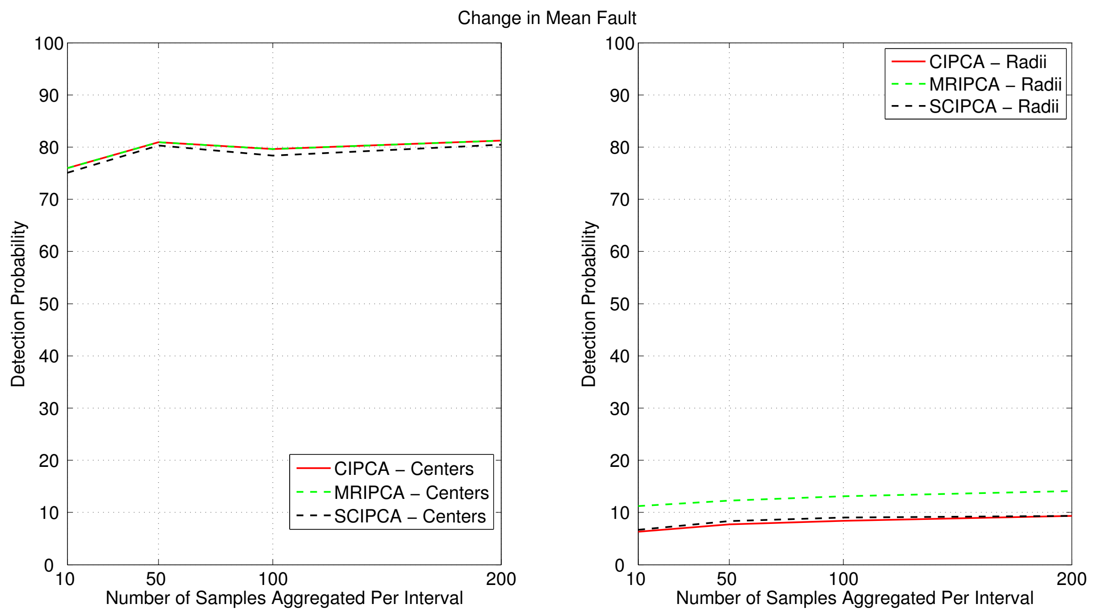


Figure 3.23: Detection probability vs m for 5% false alarm probability - change in mean fault - case study 3.

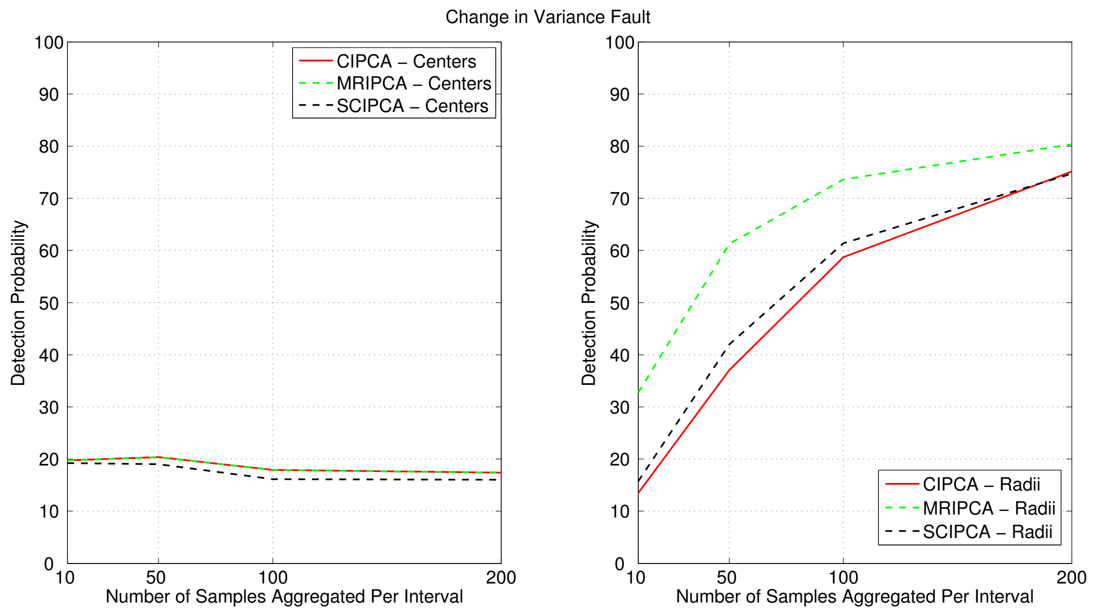


Figure 3.24: Detection probability vs m for 5% false alarm probability - change in variance fault - case study 3.

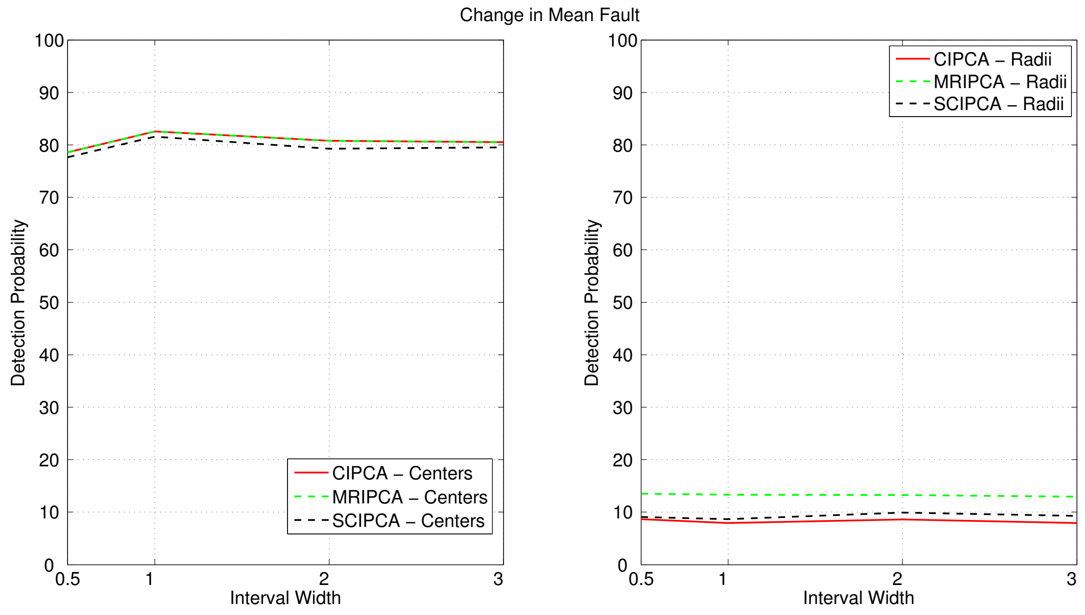


Figure 3.25: Detection probability vs β for 5% false alarm probability - change in mean fault - case study 3.

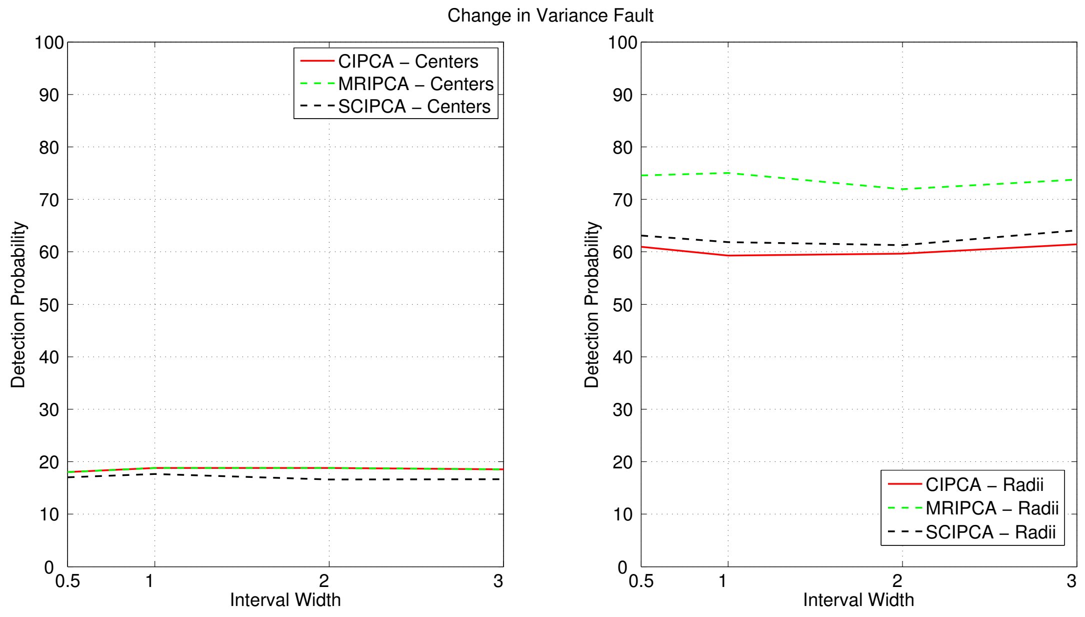


Figure 3.26: Detection probability vs β for 5% false alarm probability - change in variance fault - case study 3.

Consequently, m is set to 100, while β is set to 1 for all IPCA methods. The data modeling results are shown in Figure 3.27. Even though the graphs show that the optimal number of PCs to retain is five, only 3 PCs were retained for all methods, matching the number of independent variables in the model. The reason here is that since there are only six variables in the dataset, retaining more PCs also guarantees that less noise is filtered out, which in turn adversely affects the fault detector's performance.

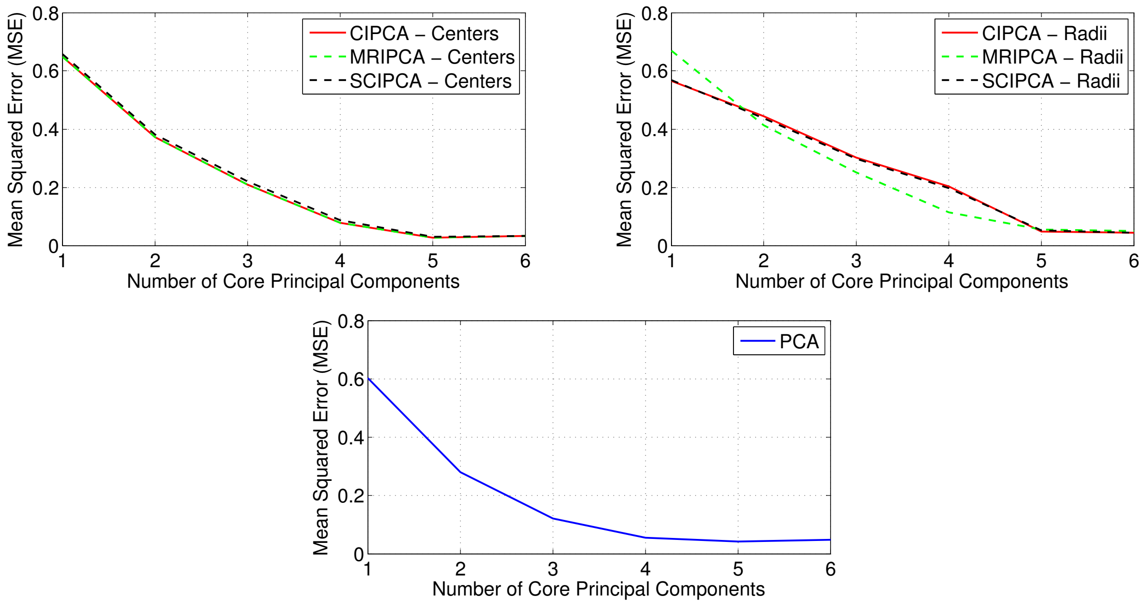


Figure 3.27: MSE for IPCA methods versus the number of retained principal components for the interval centers (top left) and interval radii (top right), and for classical PCA (bottom) - case study 3.

Figure 3.28 shows the results for a change in mean fault, and Figure 3.29 shows the results for a change in variance fault. All IPCA methods had near identical performances in the case of a change in mean fault, and were better fault detectors than classical PCA. On the other hand, MRIPCA had the best performance for a change in variance fault. As expected, the results show that the interval centers were better suited to detecting changes in mean, and interval radii were better suited to detecting changes in variance respectively.

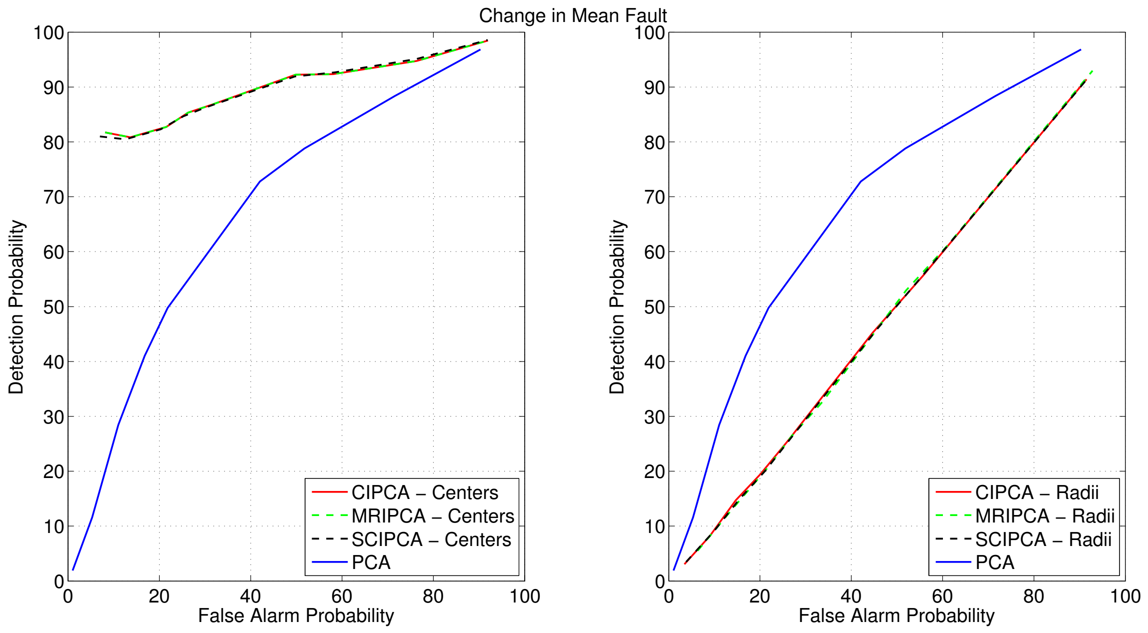


Figure 3.28: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in mean fault - case study 3.

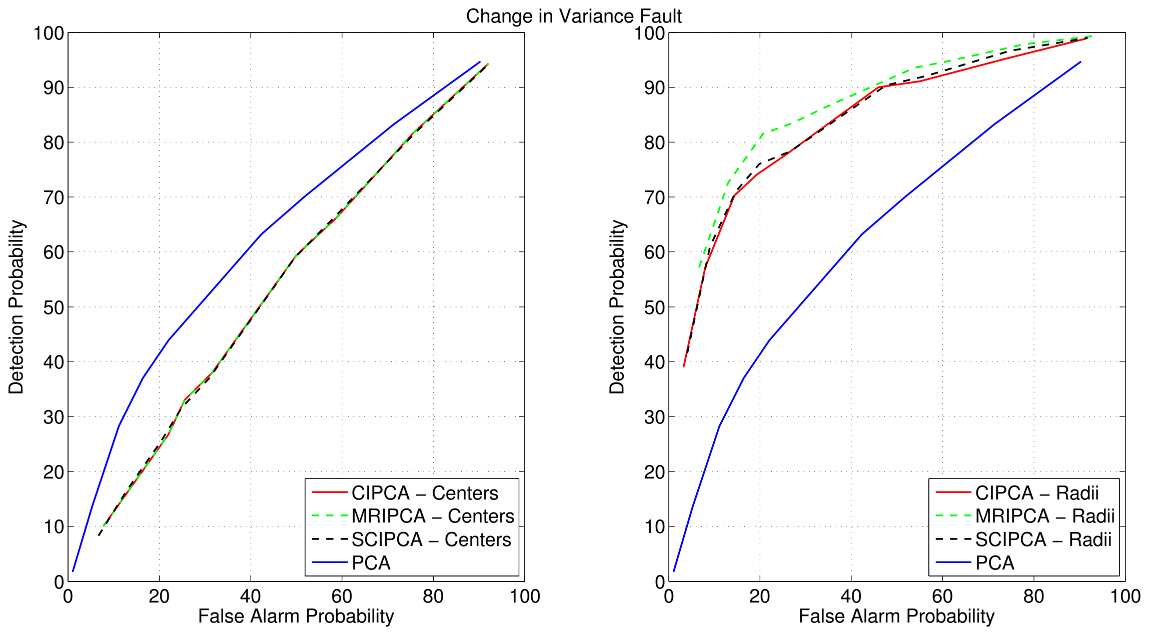


Figure 3.29: ROC curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - change in variance fault - case study 3.

3.4 Case Study 4: Gas Sensor Drift Real Data Example

In the fourth case study, the real data example is taken from the UCI data repository [25]. It contains 13910 classical measurements recorded by 16 metal-oxide (MOX) gas sensors of 6 gases at different concentrations. Each sensor reading is 8-dimensional, resulting in 128 variables per sample. For IPCA methods, 50 classical samples were aggregated per interval (i.e. $m = 50$), and the interval width coefficient was set to 1 (i.e. $\beta = 1$). The gases are labeled as follows: 1-Ethanol, 2-Ethylene, 3-Ammonia, 4-Acetaldehyde, 5-Acetone, and 6-Toluene. The precision of classifying each gas is plotted for all methods in Figure 3.30.

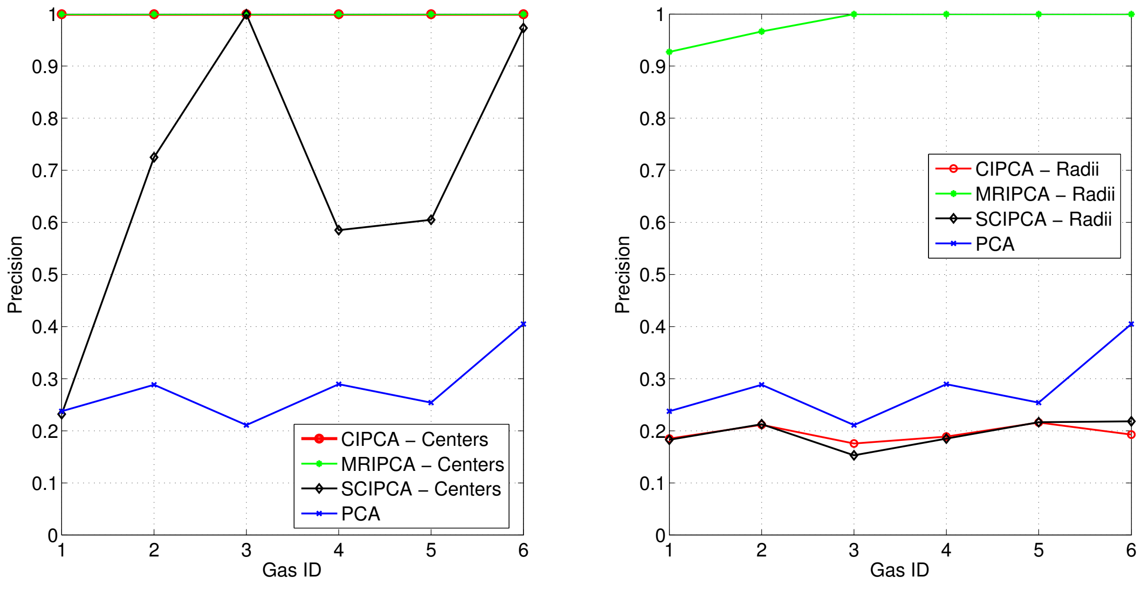
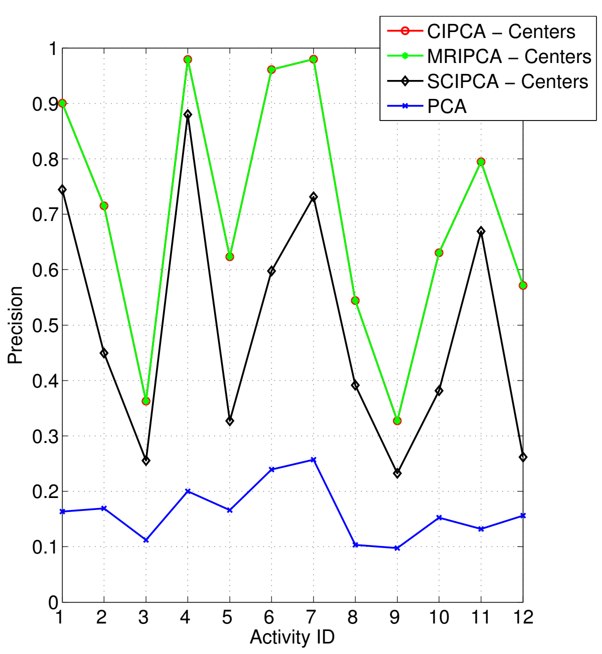


Figure 3.30: Precision curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - case study 4.

The results show that for interval centers, MRIPCA and CIPCA had a precision value of 1 for all gas types. For the interval radii, MRIPCA had the best performance, with a near perfect precision for all gas classes. As a result, MRIPCA was the most robust classifier method.

3.5 Case Study 5: Physical Activity Mapping Real Data Example

In the fifth case study, the real data example is taken from the UCI data repository [26]. The dataset contains over 3 million samples and, after removing redundant and/or invalid variables, 31 variables per sample. Due to the varying sampling rates, the heart rate variable had missing samples. For classical PCA, the missing information was padded with the previous/nearest known classical value. For IPCA methods, $m = 1000$ and $\beta = 1$. The dataset was split into six separate matrices, one for each of the following classes: 1-Cycling, 2-Going down the stairs, 3-Ironing, 4-Lying down, 5-Nordic walking, 6-Rope Jumping, 7-Running, 8-Sitting, 9-Standing, 10-Going up the stairs, 11- Vacuum cleaning, and 12-Walking. The precision plots of each physical activity is shown in Figure 3.31.



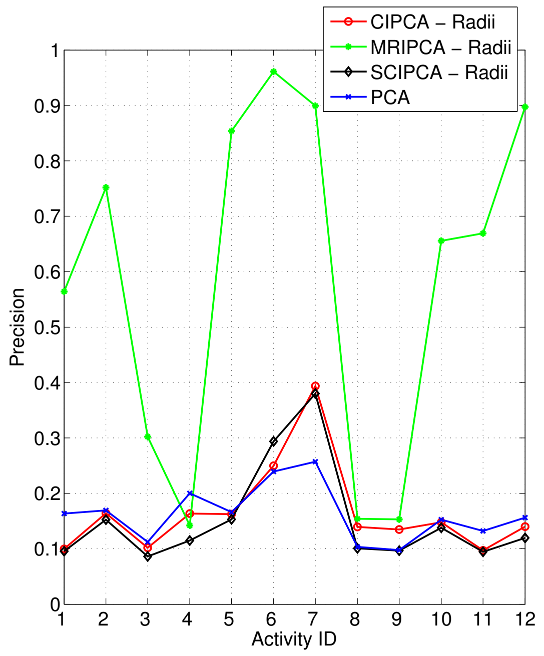
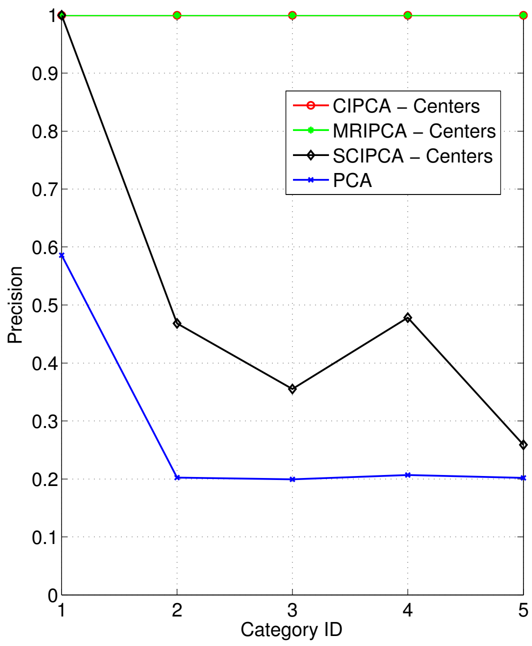


Figure 3.31: Precision curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - case study 5.

The results show once more that MRIPCA and CIPCA had the highest precision for interval centers, while, for the interval radii, MRIPCA had the highest precision by a wide margin in all categories except Category 4, where classical PCA was best.

3.6 Case Study 6: Epileptic Seizure Recognition Real Data Example

In the sixth case study, the dataset is taken from the UCI data repository [29]. There are 13800 samples in the dataset, with 178 variables per sample. It is split into five categories: 1-Recorded seizure activity, 2-Recorded the EEG from the area where a tumor was located, 3-Recorded the EEG from the healthy brain area, 4-Recorded the EEG signal when the patient had their eyes closed, 5-Recorded the EEG signal when the patient had their eyes open. For IPCA, 100 classical samples were aggregated per interval, and the interval width coefficient was set to 1. The plot of precision versus each category is shown in Figure 3.32 for all methods.



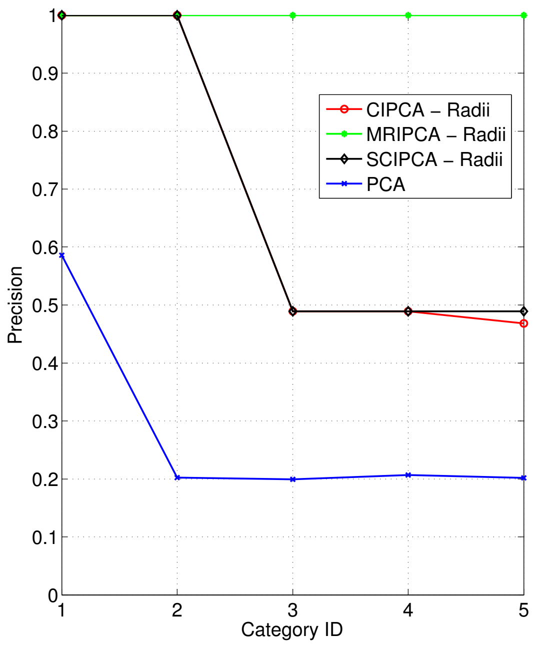


Figure 3.32: Precision curve for the interval centers (left) and radii (right) of all methods, with PCA taken as control - case study 6.

Similar to the fourth and fifth case studies, MRIPCA and CIPCA had the best classification precision across all classes for the interval centers, while MRIPCA was the best classifier for the interval radii.

4. SUMMARY AND CONCLUSIONS

Statistical process control, or process monitoring, is a crucial part of today's world for the sake quality control and safety purposes. Maintaining a high standard of quality control is not only beneficial from a business perspective, since it reduces cost, minimizes waste and streamlines the manufacturing process, it is also important for the detection of hazardous scenarios before they take place, such as runaway reactions in chemical plants.

One of the most powerful methods for process monitoring is classical Principal Component Analysis (PCA). PCA is a linear data analysis tool used to reduce the dimensionality of a multivariate dataset, while retaining most of the variability found in its cross-correlated variables. It transforms the original variables in a dataset to a new set of variables, known as the principal components (PC), which have unique mathematical properties. PCs of real datasets are necessarily decorrelated, which is crucial for the purposes of fault detection and data classification. Furthermore, Interval Principal Component Analysis (IPCA) is an extension of classical PCA, and is used to apply PCA to large datasets through the use of interval data. Three IPCA methods were explored in this thesis: the Centers IPCA (CIPCA), the Midpoint-Radii IPCA (MRIPCA), and the Symbolic Covariance IPCA (SCIPCA).

All IPCA methods were developed for statistical monitoring and their performance was compared to that of classical PCA. In addition, the methods used for fault detection are re-applied for the purposes of data classification, in order to take advantage of the IPCA methods' unique fault detection performance. The first three case studies are of synthetic models of different types.

The first case study is of a linear model, where the centers fall in a tight range of values due to the Gaussian distribution of its variables. The second case study is also of a linear model, but the intervals are generated such that their interval centers fall in a wider range of values. Finally, the third case study is of a non-linear model. Intuitively, it is expected to use non-linear methods to monitor non-linear models, such as Kernel PCA. However, for high dimensional systems with a high number of samples, using non-linear methods can be computationally expensive and time-consuming. Therefore, the use of a robust linear method can be more appealing if it provides a good degree of accuracy.

The results for the first three case studies show that all IPCA methods were much better suited to fault detection, where it had a consistently higher detection probability for the same false alarm probabilities, as shown in their respective ROC curves. Moreover, the results show that the interval centers were uniquely capable of detecting changes in the mean, while the interval radii were capable of detecting changes in the variance. This is interesting because it adds a new layer to the process monitoring applications using IPCA methods, which are now capable of detecting and differentiating the type of fault.

On the other hand, for data classification, the results show that MRIPCA was the most robust classifier. MRIPCA had a near perfect classification precision for the fourth and sixth case study for both the interval centers and radii. For the fifth case study, MRIPCA had the highest precision for the interval centers and radii by a wide margin, except for a single category.

In conclusion, the results show that IPCA methods provide more robust models for fault detection and data classification than classical PCA, with MRIPCA providing the most promising results. Moreover, the linear nature of IPCA, which make it computationally inexpensive, easily warrants further research into their application.

REFERENCES

[1] J. Oakland, *Statistical Process Control*. Taylor and Francis Group, 2011.

[2] K. Dunn, “Process improvement using data,” 2016. [Online]. Available: `https://learnche.org/pid/` [Accessed: Nov-2017].

[3] K. Pearson, “On lines and planes of closest fit to systems of points is space,” *Philosophical Magazine*, vol. 6, pp. 559–572, 1901.

[4] H. Hotelling, “Analysis of a complex of statistical variables into principal components,” *Journal of Educational Psychology*, pp. 417–441,498–520, 1933.

[5] I. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[6] P. Sanguansat, *Principal Component Analysis: Engineering Applications*. InTech, 2012.

[7] G. Strang, *Introduction to Linear Algebra*. Wellesley-Cambridge Press and SIAM, 2016.

[8] M. Masnan, A. Zakaria, A. Shakaff, N. Mahat, H. Hamid, N. Subari, and J. Saleh, “Principal component analysis: A realization of classification success in multi sensor data fusion,” in *Principal Component Analysis: Engineering Applications* (P. Sanguansat, ed.), ch. 1, pp. 1–24, InTech, 2012.

[9] P. Shenai, Z. Xu, and Y. Zhao, “Applications of principal component analysis (pca) in materials science,” in *Principal Component Analysis: Engineering Applications* (P. Sanguansat, ed.), ch. 2, pp. 25–40, InTech, 2012.

[10] P. Sanguansat, *Principal Component Analysis: Multidisciplinary Applications*. InTech, 2012.

[11] E. Belasco, B. P. Jr., and G. Gong, “The health care access index as a determinant of delayed cancer detection through principal component analysis,” in *Principal Component Analysis: Multidisciplinary Applications* (P. Sanguansat, ed.), ch. 9, pp. 143–166, InTech, 2012.

[12] L. Billard and J. Le-Rademacher, “Principal component analysis for interval data,” *WIREs Comput Stat*, vol. 4, pp. 535–540, 2012.

[13] A. Benaicha, G. Mourot, J. Ragot, and K. Benothman, “Fault detection and isolation with interval principal component analysis,” *International Conference on Control, Engineering and Information Technology*, vol. 1, pp. 162–167, 2013.

[14] T. Izem, W. Bougheloum, M. Harkat, and M. Djeghaba, “Fault detection and isolation using interval principal component analysis methods,” *International Federation of Automatic Control*, pp. 1402–1407, 2015.

[15] P. Cazes, A. Chouakria, E. Diday, and Y. Schektman, “Extension de l’analyse en composantes principales à des données de type intervalle,” *Revue de Statistique Appliquée*, vol. 3, pp. 5–24, 1997.

[16] J. Le-Rademacher, *Principal Component Analysis for Interval-Valued and Histogram-Valued Data and Likelihood Functions and Some Maximum Likelihood Estimators for Symbolic Data*. Ph.D. dissertation, The University of Georgia, Athens, 2008.

[17] F. Palumbo and N. Lauro, “A pca for interval-valued data based on midpoints and radii,” *New Developments in Psychometrics*, pp. 641–648, 2001.

[18] N. Lauro and F. Palumbo, “Principal component analysis of interval data: A symbolic data analysis,” *Computational Statistics*, pp. 73–87, 2000.

[19] N. Lauro, R. Verde, and A. Irpino, "Principal component analysis of symbolic data described by intervals," *Symbolic Data Analysis and the SODAS Software*, pp. 279–311, 2008.

[20] N. Lauro and F. Palumbo, "Principal component analysis for non-precise data," *Studies in Classification, Data Analysis, and Knowledge Organization*, pp. 173–184, 2005.

[21] J. Le-Rademacher and L. Billard, "Symbolic covariance principal component analysis and visualization for interval-valued data," *Journal of Computational and Graphical Statistics*, vol. 2, pp. 413–432, 2012.

[22] Y. Tharrault, G. Mourot, J. Ragot, and M. Harkat, "Sensor fault detection and isolation by robust principal component analysis," in *Fault Detection* (W. Zhang, ed.), ch. 17, pp. 369–392, InTech, 2010.

[23] C. Aggarwal, *Data Classification Algorithms and Applications*. CRC Press, 2015.

[24] A. Vergara, S. Vembu, T. Ayhan, M. Ryan, M. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," *Sensors and Actuators B: Chemical*, pp. 320–329, 2012.

[25] A. Vergara, "Gas sensor array drift dataset data set," 2012. [Online]. Available: `http://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset` [Accessed: Oct-2017].

[26] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," *The 16th IEEE International Symposium on Wearable Computers (ISWC)*, 2012.

[27] A. Reiss, "Pamap2 physical activity monitoring data set," 2012. [Online]. Available: `http://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+`

`Activity+Monitoring` [Accessed: Oct-2017].

[28] R. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," *Physical Review E*, vol. 64, 2001.

[29] Q. Wu and E. Fokoue, "Epileptic seizure recognition data set," 2017. [Online]. Available: `http://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition` [Accessed: Oct-2017].