CHAPTER 1: BACKGROUND ..... 1
1.1 Overview ..... 1
1.2 Principal component analysis ..... 2
1.2.1 Overview ..... 2
1.2.2 Algorithm ..... 4
1.2.3 Monitoring statistics and control limits ..... 6
1.2.4 Fault detection method ..... 6
1.3 Recursive principal component analysis ..... 7
1.3.1 Overview ..... 7
1.3.2 Algorithm ..... 7
1.3.2.1 Adaptation of the correlation matrix ..... 7
1.3.3 Monitoring statistics and adaptation of control limits ..... 8
1.3.4 Fault detection method ..... 8
1.3.5 Conditions to update model ..... 8
1.4 Moving window PCA ..... 9
1.4.1 Overview ..... 9
1.4.2 Algorithm ..... 10
1.4.3 Monitoring statistics and adaptation of control limits ..... 11
1.4.4 Fault detection method ..... 11
1.5 Gaussian mixture model ..... 11
1.5.1 Overview ..... 11
1.5.2 Algorithm ..... 12
1.5.2.1 Estimation of Gaussian mixture density ..... 13
1.5.2.2 EM Algorithm ..... 14
1.5.2.3 Covariance structure model ..... 16
1.5.2.4 Model Selection ..... 18
1.5.3 Monitoring statistics and control limits ..... 19
1.5.4 Fault detection method ..... 20
1.6 PCA-based GMM ..... 20
1.6.1 Overview ..... 20
1.6.2 Algorithm ..... 21
1.6.3 Monitoring statistics and control limits ..... 21
1.6.4 Fault detection ..... 21
CHAPTER 2: METHODOLOGY ..... 22
2.1.1 Application of recursive PCA ..... 23
2.1.1.1 Model update and condition to update ..... 24

2.1.2 Application of MWPCA ..... 25
2.1.3 Application of GMM ..... 26
2.1.4 Application of PCA-based GMM ..... 27
2.2 Adaptive PCA-based GMM ..... 28
2.2.1 Overview ..... 28
2.2.2 Algorithm ..... 29
2.2.3 Monitoring statistics and control limits ..... 30
2.2.4 Fault detection ..... 30
2.2.5 Application ..... 30
2.2.5.1 Model update and condition to update ..... 31
ABBREVIATIONS AND SYMBOLS ..... 34
REFERENCES ..... 36

# CHAPTER 1: BACKGROUND

*This chapter presents the techniques identified to address each stated objective. The methodology is divided into two sections. The first section describes PCA and adaptive PCA approaches. The second section describes Gaussian mixture models (GMMs) and PCA-based GMM approaches.*

## 1.1 Overview

The adaptive PCA 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 normal operating conditions (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 adaptive PCA methods.

Figure 1.1 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 1.2 presents a simple flow diagram showing the individual monitoring models of GMM and PCA.

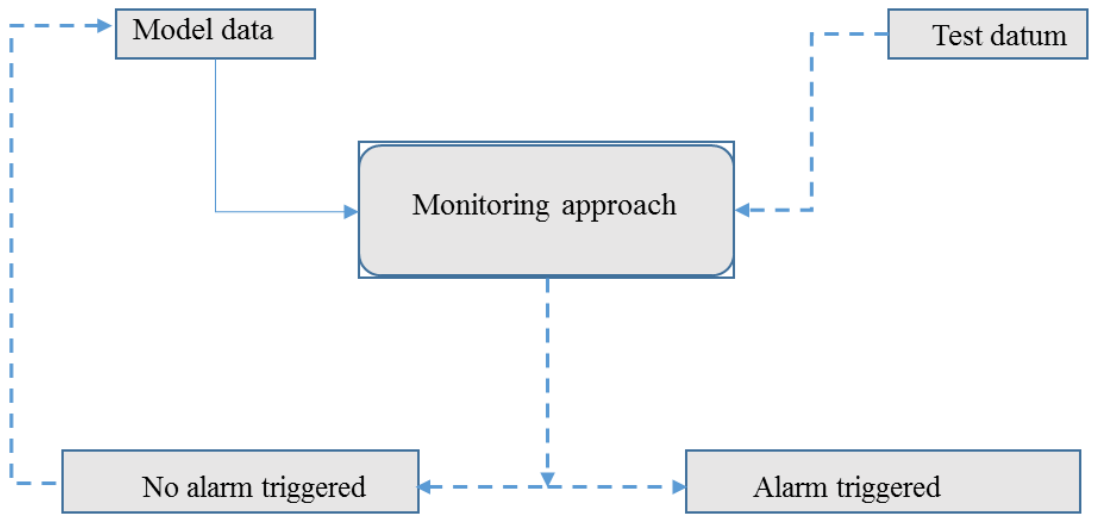


Figure 1.1: Adaptation of the monitoring model (for an approach) by the addition of new observations test datum) to the model data.

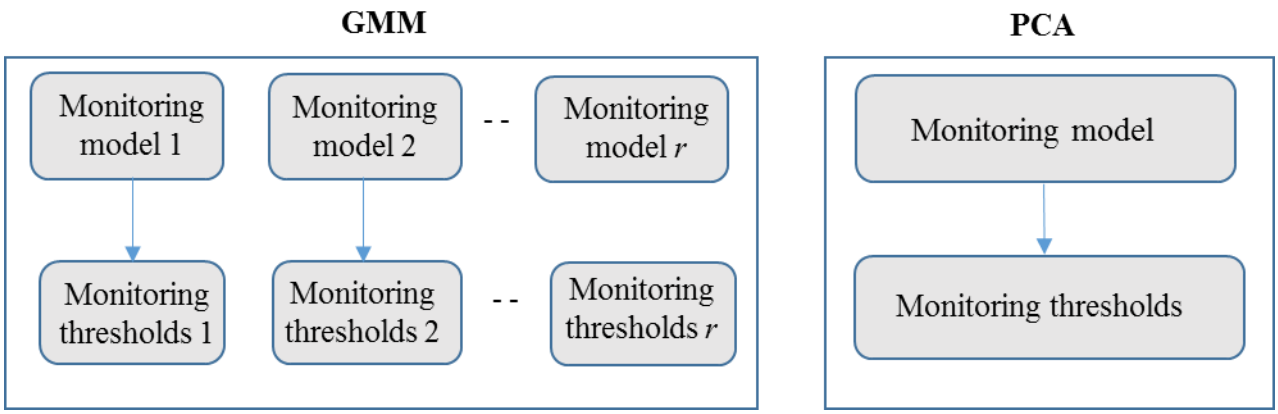


Figure 1.2: Monitoring approaches for GMM (left) (for $r$ number of modes) and PCA (right) showing the individual monitoring models.

## 1.2 Principal component analysis

### 1.2.1 Overview

Principal component analysis (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 normal operating conditions 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 access 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 1.4.

The monitoring strategies involve using the scores and the reconstructed data. This is done via the modified Hotelling's $T^2$ statistic and squared prediction error (SPE) respectively. Figure 1.4 provides a view of how the monitoring statistics are applied in combination with PCA for a simple 2-dimensional example. The figure shows how the normalized data in the original input space is transformed into the lower dimensional feature space by finding the principal components and retaining the ones with the highest explained variance (which is highlighted by the dashed-red line in this case). The modified Hotelling's $T^2$ measures the distance of an observation from the centre of the feature space while the SPE measures the distance between the input space and the feature space of an observation. The derivation of the model parameters and relevant monitoring statistics are presented in the next subsection.

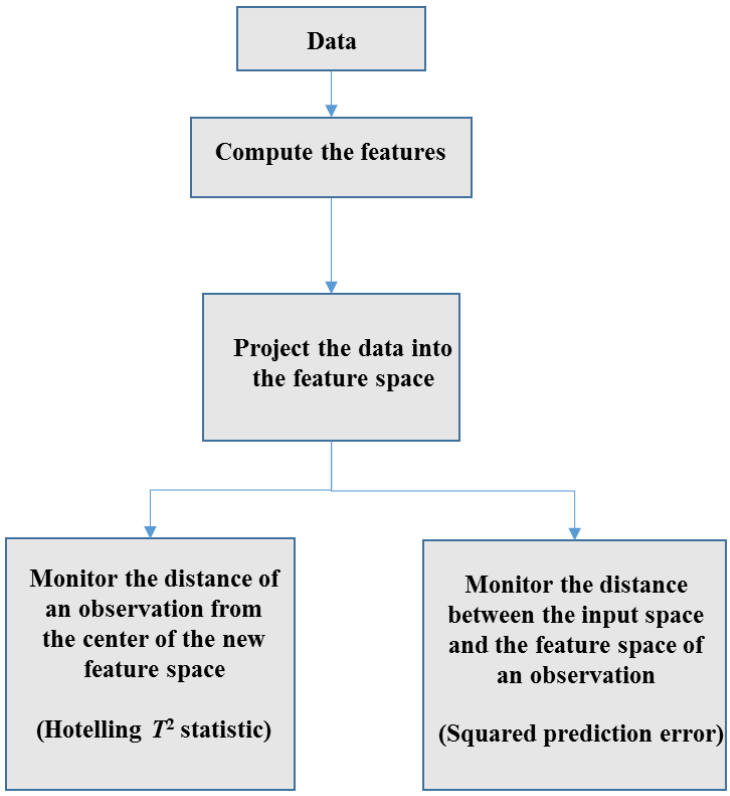


Figure 1.3: Conceptual diagram for fault detection using PCA.

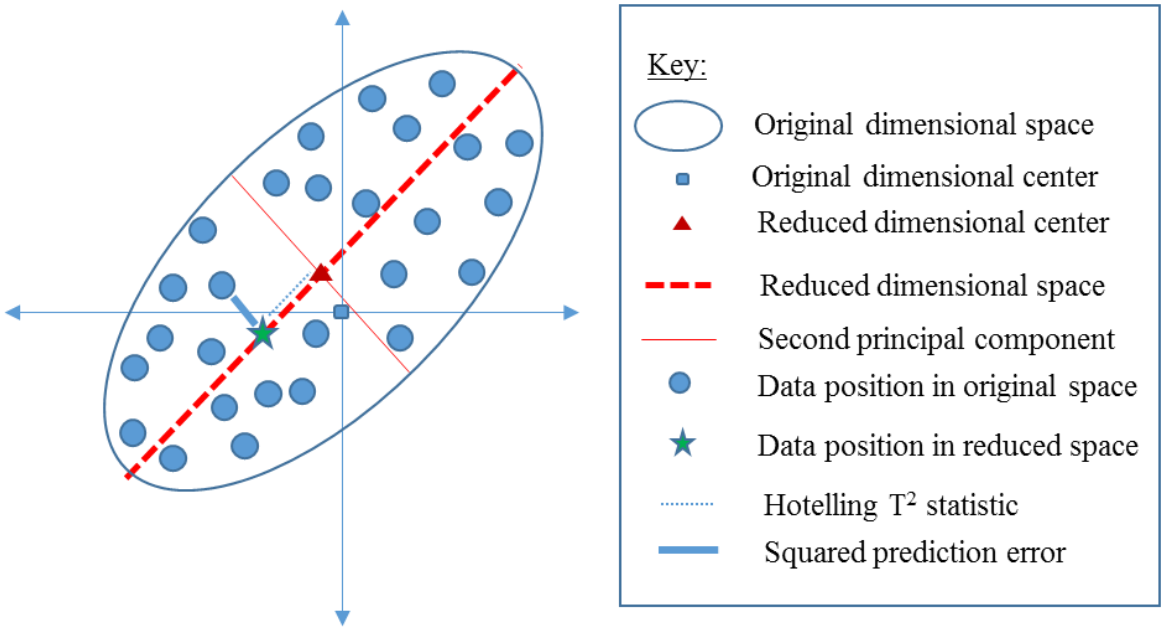


Figure 1.4: Pictorial view of dimension reduction of normalized 2-D data to 1-D by projection onto the first principal component.

### 1.2.2 Algorithm

The derivation of the model parameters for PCA involves the calculation of the principal components 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 $\mu_j$ denote the mean of the j-th variable:

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

2. Calculate the standard deviation of the j-th variable, $\sigma_j$:

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

3. Each variable is normalized. The variable value for each observation is centred by subtracting the variables mean $\mu_j$ and scaling the result by dividing it by its standard deviation $\sigma_j$:

$$x_{i,j} = \frac{d_{i,j} - \mu_j}{\sigma_j} \qquad \textbf{1-3}$$

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

**Computing the correlation matrix**

The correlation matrix, $\boldsymbol{C}$, for the normalized data, $\boldsymbol{X}$, is computed as:

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

**Computing the loading vectors by using eigen-decomposition**

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

$$\boldsymbol{CP = \lambda P} \qquad \textbf{1-5}$$

$\boldsymbol{P}$ and $\boldsymbol{\lambda}$ respectively represent the matrix of eigenvectors and the vector of eigenvalues produced as solutions.

**Computing the scores**

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

$$\boldsymbol{T = XP} \qquad \textbf{1-6}$$

**Retaining first $\upsilon$ loading vectors with largest eigenvalues**

The fraction of variance in the normalized data matrix accounted for by a principal component is computed as shown in Equation **1-7**.

$$\frac{\lambda_i}{\Sigma\lambda_i}. \qquad \textbf{1-7}$$

Thus using the υ principal components corresponding to the $\upsilon$ largest eigenvalues collectively account for some fraction $L_\upsilon$:

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

$\widehat{\boldsymbol{P}} \in R^{m\times\upsilon}$, $\widehat{\boldsymbol{T}} \in R^{n\times\upsilon}$ and $\widehat{\boldsymbol{\lambda}} \in R^{l\times\upsilon}$ respectively represent the principal components, scores, and variances retained once is υ decided.

**Reconstruction**

The transformation of the score matrix back into the original dimensional observational space can now be computed as in Equation **1-9**.

$$\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{T}}\left(\widehat{\boldsymbol{P}}\right)^T \qquad \textbf{1-9}$$

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 $\boldsymbol{E}$:

$$\boldsymbol{E = X - \widehat{X}} \qquad \textbf{1-10}$$

### 1.2.3 Monitoring statistics and control limits

Monitoring using PCA involves the use of the squared prediction error (SPE or Q) and Hotelling's $T^2$ statistics for the retained loadings, which will be called the modified Hotelling's $T^2$ statistic (T-squared) 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:

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

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

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

The modified Hotelling's $T^2$ statistic control limit, $(\hat{t}^2)_\alpha$, is determined using the probability distribution (Russell, Chiang & Braatz, 2000):

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

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

The detection limit for Q statistic with a significance level $\alpha$ as approximated by Jackson & Mudholkar (1979) is:

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

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 $z_\alpha$ is the normal deviate corresponding to the $(1 - \alpha)$ percentile.

### 1.2.4 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 principal components identified during training.

The modified Hotelling's $T^2$ and SPE statistics for the new point are computed as shown in Equations **1-12** and **1-11** respectively and checked against the calculated thresholds, $(t_A^2)_\alpha$

and $q_\alpha$. 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 (Ayech, Chakour & Harkat, 2012; Choi, Martin, Morris & Lee, 2005; Choi, Park & Lee, 2004).

The next section looks at recursive PCA which is an adaptive PCA approach.

## 1.3 Recursive principal component analysis

### 1.3.1 Overview

Recursive PCA (Li, Yue, Valle-Cervantes & Qin, 2000) and moving window PCA (Wang, Kruger & Irwin, 2005) (presented in Section 1.4) are the adaptive PCA methods we consider in this work. Recursive PCA 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. Recursive PCA 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 normal operating conditions data.

### 1.3.2 Algorithm

The recursive PCA 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 as listed in Section 1.2.2. 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.

#### 1.3.2.1 Adaptation of the correlation matrix

As the data window (initially set at the outset) is augmented with a new observation, changes occur in the mean 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 $w_{k-1}$,$w_k$, $w_{k+1}$ respectively. Also, let the diagonal matrix of some mean vector $\boldsymbol{\sigma_\bullet}$ be denoted by $\boldsymbol{\Sigma_\bullet}$. For the data window, the initial mean vector $\boldsymbol{\mu}_k$, initial variance $\boldsymbol{\sigma}_k^2$ and initial correlation matrix $\boldsymbol{C}_k$ are updated for a new observation $\boldsymbol{d}_{k+1}$ as:

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

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

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

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

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.

### 1.3.3 Monitoring statistics and adaptation of control limits

Recursive PCA 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 1.2.3 but with updated model parameters.

### 1.3.4 Fault detection method

Recursive PCA follows the same fault detection procedure as in the conventional PCA (see Section 1.2.4). 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 normal operating conditions, the monitoring statistics at each interval is checked against their corresponding thresholds to make sure they are not beyond their respective thresholds.

### 1.3.5 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:

First update method (UM-1) allows the model update if no alarm is triggered for a consecutive number of observations. For example, if $z = 3$, and at least one observation is below the threshold for both the SPE and T-squared statistic, 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-squared 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* number of observations for both of the monitoring statistics (SPE and T-squared) must be in control before an update can occur. To put things in perspective, UM-3 is similar to UM-2 when $z = 1$ (Tien, 2005; Zhou, Ye, Zhang & Li, 2016).

The next section looks at the moving window PCA which is the other adaptive PCA approach considered.

## 1.4 Moving window PCA

### 1.4.1 Overview

In contrast to recursive PCA, Moving window PCA (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 1.5 shows how the training data window gets updated with new observations for the two approaches.

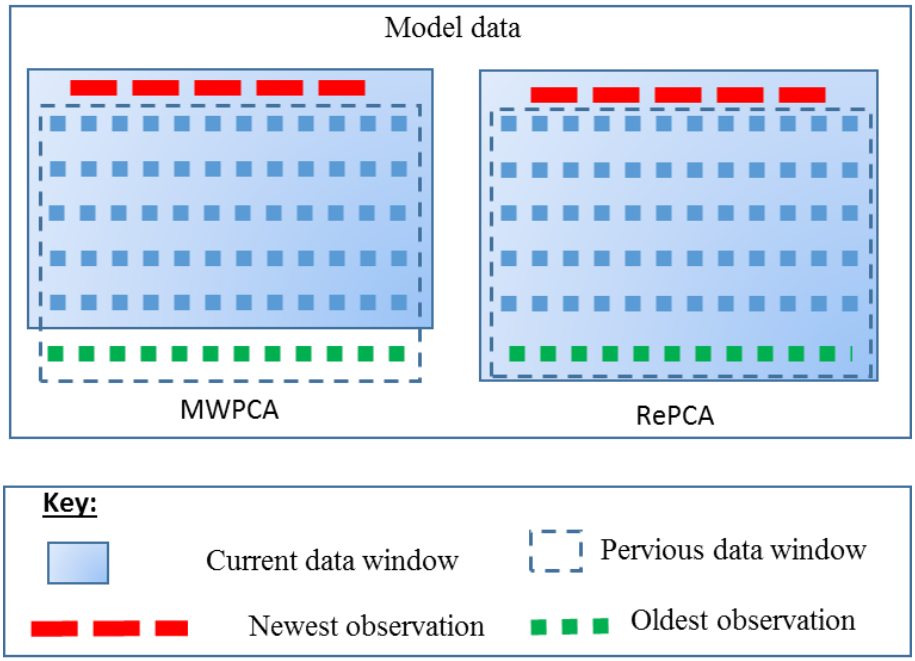


Figure 1.5: Differences in model update methods for recursive PCA (RePCA) and MWPCA.

### 1.4.2 Algorithm

The MWPCA algorithm follows the approach described for recursive PCA 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:

**Remove the oldest observation (down-dating)**

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

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

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

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

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 vector $\boldsymbol{\mu}_{k+1}$, variance vector $\boldsymbol{\sigma}_{k+1}^2$ and correlation matrix $\boldsymbol{C}_{k+1}$ which are computed as:

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

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

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

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

Same as recursive PCA, 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.

### 1.4.3 Monitoring statistics and adaptation of control limits

MWPCA implements the same monitoring statistics as that of the conventional and recursive PCA. The computation of monitoring statistics for the updated window follows the same method as outlined for conventional PCA in Section 1.2.3 but with changing retained components υ over time.

### 1.4.4 Fault detection method

MWPCA follows the same fault detection procedure as recursive PCA in Section 1.5.4.

The next section looks at the Gaussian mixture model which is a multimodal approach.

## 1.5 Gaussian mixture model

### 1.5.1 Overview

Gaussian mixture models (GMM) (Yan, Hyewon & Soohyun, 2008; Yu, 2012) are common machine intelligence technique used for modelling data. GMMs describes complex process data as a mixture 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 normal operating conditions data into a training and a validation data set. 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 are presented in the next sections.

### 1.5.2 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:

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

Here, $\boldsymbol{S}_j$ and $\boldsymbol{\mu}_j$ are respectively the covariance matrix and the mean of the $j^{\text{th}}$ 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 **1-26**.

$$\boldsymbol{\theta} = (\varrho_1, \boldsymbol{\mu}_1, \boldsymbol{S}_1, \ldots, \pi_1, \boldsymbol{\mu}_r, \boldsymbol{S}_r) \qquad \textbf{1-26}$$

The mixture weights of the $j^{\text{th}}$ component are $\varrho_j$, whereby $0 \leq \varrho_j < 1$ is true for all components, and $\sum_{j=1}^{r} \varrho_j = 1$. The mixture weight $\varrho_j$ represents the probability that a new observation

belongs to the cluster *j*. Figure 1.6 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 **1-27**.

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

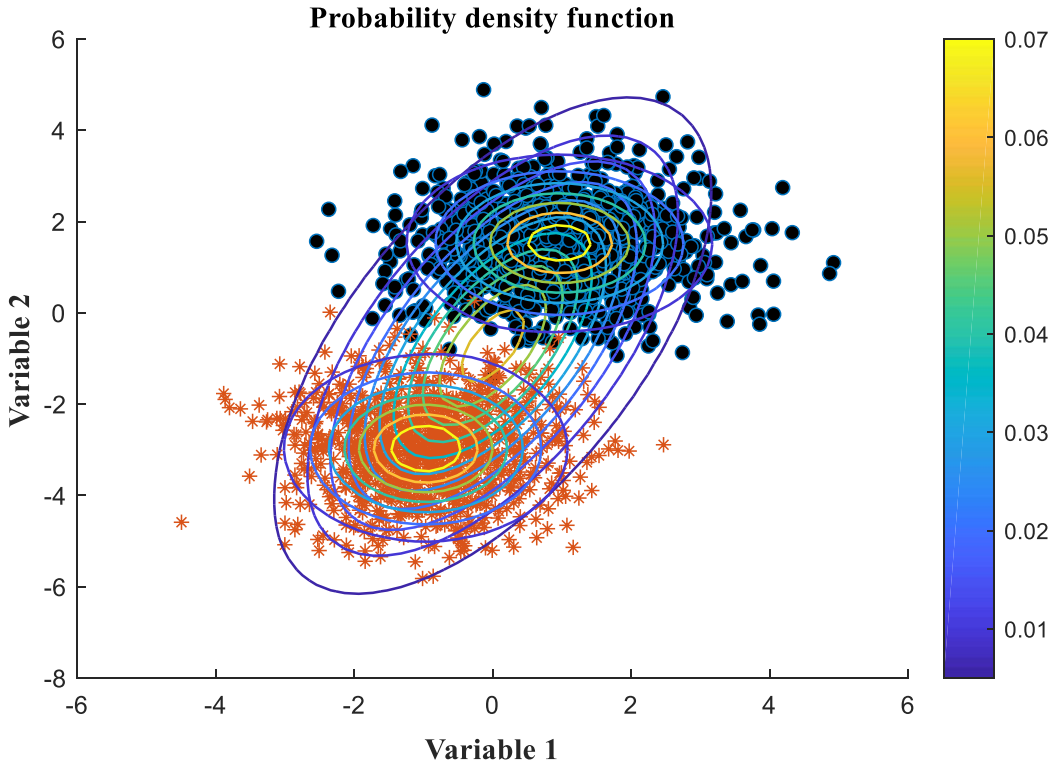


Figure 1.6: An example of Gaussian mixture density fitted to bi-modal data with two clusters.

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.

#### 1.5.2.1 Estimation of Gaussian mixture density

Estimation and selection of model parameters involve checking how the distribution built by the estimated parameters fit the data (i.e. how close the estimated parameters match the true parameters). 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 **1-28**.

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

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

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\boldsymbol{D}) \qquad \textbf{1-29}$$

Taking the log of the likelihood function in Equation **1-28**, yields Equation **1-30** transforming the product of potentially small likelihoods into a sum of logs, which is easier to distinguish from 0 in computation. The Equation in **1-30** is therefore maximized instead of the likelihood function in Equation **1-28** because it is computationally easier to handle.

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

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.

#### 1.5.2.2 EM Algorithm

The EM algorithm (Dempster, Laird & 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 log-likelihood is subsequently formulated as shown in Equation **1-31**.

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

The iterative process of EM algorithm consists of two procedure, 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 $\delta_{ij}$ denote the expectation of observation $\boldsymbol{d}_i$ belonging to the $j^{\text{th}}$ cluster for the current model parameter estimates. The $\delta_{ij}$ called "responsibilities" is computed as:

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

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

$$\varrho_j{}^{t+1} = \frac{1}{n}\sum_{i=1}^{n} \delta_{ij} \quad \textbf{1-33}$$

$$\mu_j{}^{t+1} = \frac{\sum_{i=1}^{n} \delta_{ij}\ \boldsymbol{d}_i}{\sum_{i=1}^{n} \delta_{ij}} \quad \textbf{1-34}$$

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

Equation 1-33 can be interpreted as updating the mixture weights $\varrho_j$ 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 **1-27**) and then calculating the average of the posterior probabilities of each sample point belonging to the component *j* (see Equation **1-32**). Equation **1-34** can be interpreted as updating the mean $\mu_j$ of a cluster by weighting the observations by their probability of being part of that cluster. Equation **1-35** 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:

$$r_{max} = n^{0.3} \quad \textbf{1-36}$$

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 & 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 ($\tau$) 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 $\tau$. 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.

**1.5.2.3 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 & 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.

Apart from the orientation of the ellipsoids in terms of the axes, the volume and orientations is also a consideration to be made (Erar, 2011). 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 un-identical. 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 1.7 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 **1-37**.

$$ɦ = \gamma + ɣ \qquad \textbf{1-37}$$

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

Table 1: Parameters of various covariance shapes and the number of parameters in the covariance matrix.

| **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.

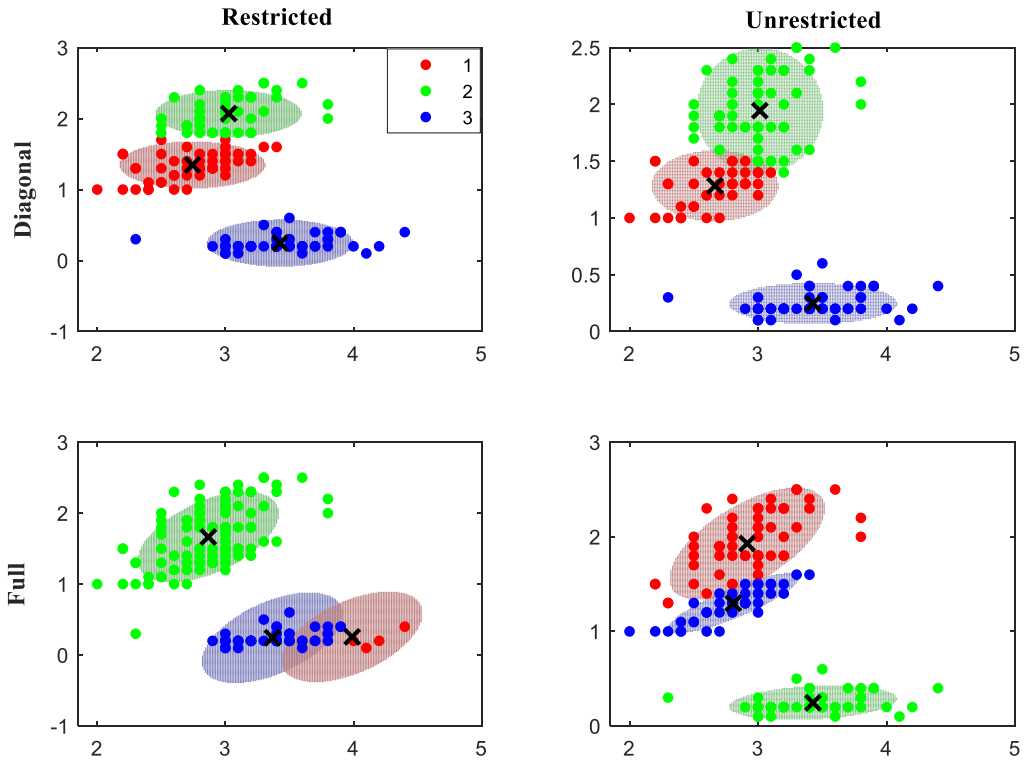


Figure 1.7: An example of different clustering results for the listed covariance structures.

### 1.5.2.4 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 **1-38** and **1-39** respectively.

$$\mathrm{AIC} = -2\log L(\boldsymbol{\theta}^*|\boldsymbol{D}) + 2ɦ \qquad \textbf{1-38}$$

$$\mathrm{BIC} = -2\log L(\boldsymbol{\theta}^*|\boldsymbol{D}) + ɦ \log(n) \qquad \textbf{1-39}$$

In the formulae, $n$ is the number of observations, ɦ is the number of independent parameters to be estimated (see Equation **1-37**) 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 & Buckley, 2004).

As mentioned by Yu (2011), deciding on the best criterion involves using prior experimental work which looks at 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 **1-40** and **1-41**.

$$h_1 = |BIC_b - AIC_c| \qquad \textbf{1-40}$$

$$h_2 = |AIC_b - BIC_c| \qquad \textbf{1-41}$$

The best model selected thereafter is then the best model producing the minimum value out of $h_1$and $h_2$ values. That is if min $\{h_1, h_2\}$ equals $h_1$, the model with the lowest BIC score ($BIC_b$) is selected.

### 1.5.3 Monitoring statistics and control limits

Monitoring using GMM involves the use of the estimated PDF (see Equation **1-25**). The PDF indicates how close an observation follows the obtained 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.

### 1.5.4 Fault detection method

GMM fault detection procedure involves checking if an observation is unusual for to 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 1.5.2.4) from a number of plausible GMM models obtained by the training data for the various covariance structures options and 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 looks at the PCA-based GMM which is a combination of PCA and GMM.

## 1.6 PCA-based GMM

### 1.6.1 Overview

PCA combined with GMM involves the use of principal component scores $\boldsymbol{T}$ (see Section 1.2.2) 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 data set ($\boldsymbol{D}$) is replaced with the scores ($\boldsymbol{T}$). For example,

the probability function of the raw data (shown in Equation **1-25**) is updated from the raw data formulation to the scores formulation as shown in Equation 1-42.

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

### 1.6.2 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 1.2.2.

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 1.5.2.

### 1.6.3 Monitoring statistics and control limits

The PCA-based GMM employs the same monitoring statistics and detection thresholds as shown for GMM in Section 1.5.3.

### 1.6.4 Fault detection

PCA-based GMM employs the same fault detection method as shown for GMM in Section 1.5.4.

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

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

# CHAPTER 2: METHODOLOGY

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

### 1.1.1 Application of PCA

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

**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 **1-1** to **1-3**].
3. Compute the correlation matrix $\boldsymbol{C}$ of the normalized data $\boldsymbol{X} \in R^{n\times m}$ [see Equation **1-4**].
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 **1-5**].
5. Retain first (major) $\upsilon$ principal components $\widehat{\boldsymbol{P}}$ and their corresponding variances $\hat{\boldsymbol{\lambda}}$ [see Equation **1-7**].
6. Compute the retained scores, $\widehat{\boldsymbol{T}} \in R^{n\times\upsilon}$ [see Equation **1-6**].
7. Reconstruct the data from, $\widehat{\boldsymbol{T}}$ [see Equation **1-9**].
8. Calculate the squared prediction errors $\boldsymbol{q}$ and its detection threshold $q_\alpha$ of the reconstructed data $\widehat{\boldsymbol{X}}$ [see Equations **1-11** and **1-14**].
9. Compute the modified Hotelling's $T^2$ statistics $\hat{\boldsymbol{t}}^2$ and its detection threshold $(\hat{t}^2)_\alpha$ [see Equations **1-12** and **1-13**].
10. Obtain the validation data $\boldsymbol{D_v} \in R^{\acute{n}\times m}$ (obtained by splitting the original data).
11. Normalize the validation data using the retained means and standard deviations from procedure 2.
12. Compute the scores by projecting the normalized data $\boldsymbol{X_v} \in R^{\acute{n}\times m}$ on $\widehat{\boldsymbol{P}}$ [ $\widehat{\boldsymbol{T}}_{\boldsymbol{v}} = \boldsymbol{X_v} \times \widehat{\boldsymbol{P}}$].
13. Compute $\hat{\boldsymbol{t}}^2{}_{\boldsymbol{v}}$ using the retained principal components $\widehat{\boldsymbol{P}}$ and eigenvalues $\hat{\boldsymbol{\lambda}}$.
14. Reconstruct the data and compute the squared prediction error $\boldsymbol{q_v}$ [see Equations **1-9** and Equations **1-11**].
15. Test how the derived model parameters ($\widehat{\boldsymbol{P}}$ and $\hat{\boldsymbol{\lambda}}$) and thresholds ($q_\alpha, (\hat{t}^2)_\alpha$) perform for unseen NOC data by verifying with the validation data monitoring statistics and readjust the retained parameters and thresholds if possible.

16. Save model for online deployment.

**Online (Deployment)**

1. Obtain new observation ***d***.
2. Normalize the sample using the retained means and standard deviations from the training stage [see Equations **1-1** to **1-3**].
3. Compute the 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 squared prediction error *q* of the reconstructed datum $\widehat{\boldsymbol{x}}$ [see Equation **2-1**].
6. Compute $\hat{t}^2$ using the retained principal components and eigenvalues [see Equation **2-2**].
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 looks at the application of recursive PCA for fault detection.

### 2.1.1 Application of recursive PCA

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

To access if 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* ¦ *w, μ, σ*) denote the normalization of some variable *w* using *μ*, *σ* as some mean and standard deviation respectively to produce some variable *b* as shown below:

$$b = \frac{w - \mu}{\sigma} \qquad \textbf{2-3}$$

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

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

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

5. Compute old score by Proj $(\hat{\boldsymbol{t}}_{old}|(\boldsymbol{x}_{k+1})_{old},(\widehat{\boldsymbol{P}})_k)$
6. Compute the reconstructed data for all the scores by Proj $((\hat{\boldsymbol{x}}_{k+1})_{new}|\hat{\boldsymbol{t}}_{new},(\widehat{\boldsymbol{P}})_{k+1}^T)$, Proj $((\hat{\boldsymbol{x}}_{k+1})_{int}|\hat{\boldsymbol{t}}_{int},(\widehat{\boldsymbol{P}})_k^T)$ and Proj $((\hat{\boldsymbol{x}}_{k+1})_{old}|\hat{\boldsymbol{t}}_{old},(\widehat{\boldsymbol{P}})_k^T)$
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 **2-5**.

$$(\boldsymbol{e}_{k+1})_{\bullet} = (\boldsymbol{x}_{k+1})_{\bullet} - (\hat{\boldsymbol{x}}_{k+1})_{\bullet} \qquad \textbf{2-5}$$

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

8. Compute the squared prediction errors for all the reconstructed data as shown in Equation **2-6**.

$$(q_{k+1})_{\bullet} = (e_{k+1})_{\bullet}.(e_{k+1})_{\bullet} \qquad \textbf{2-6}$$

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

Compute the modified Hotelling's $T^2$ statistic for all the scores. Using: $\boldsymbol{\Lambda}_{new}=\boldsymbol{\Lambda}_{k+1}$, $\boldsymbol{\Lambda}_{int}=\boldsymbol{\Lambda}_{old}=\boldsymbol{\Lambda}_k$, $\boldsymbol{P}_{new}=(\widehat{\boldsymbol{P}})_{k+1}$, and $\boldsymbol{P}_{int}=\boldsymbol{P}_{old}=(\widehat{\boldsymbol{P}})_k$, the values for $\hat{t}_{new}$, $\hat{t}_{old}$ and $\hat{t}_{int}$ are calculated as shown in Equation **2-7**.

$$((\hat{t}^2)_{k+1})_{\bullet}=(\boldsymbol{x}_{k+1})_{\bullet}\boldsymbol{P}_{\bullet}\boldsymbol{\Lambda}_{\bullet}^{-1}\boldsymbol{P}_{\bullet}^T(\boldsymbol{x}_{k+1}^T)_{\bullet} \qquad \textbf{2-7}$$

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*.

#### 2.1.1.1 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 $((t_A^2)_{k+1})_{old}$ 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 put 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 **1-15** to **1-18**].
3. Compute and retain the principal components $(\widehat{\boldsymbol{P}})_{k+1}$ and variances $(\hat{\boldsymbol{\lambda}})_{k+1}$ of $\boldsymbol{C}_{k+1}$.
4. Compute the retained scores $(\widehat{\boldsymbol{T}})_{k+1}$ [see Equation **1-6**].
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 Procedure to 1 to 8 of Section 2.1.1].
7. Update model parameters if appropriate (using Procedure 1 to 2 of the model update condition in Section 2.1.1.1).

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, looks at the application of moving window PCA which is the other considered adaptive PCA approach for fault detection.

### 2.1.2 Application of MWPCA

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

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

**Offline (Training)**

The offline application for MWPCA follows the same procedure as listed in the offline application for recursive PCA in Section 2.1.1.

**Online (Deployment)**

The online application of recursive PCA 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 **1-19** to **1-21**].
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-dated window $\boldsymbol{D}_{k+1} \epsilon R^{n\times m}$ [see Equations **1-22** to **1-24**].

Once the updated window is created, Procedure 1 to 7 of the online application of recursive PCA as listed in Section 2.1.1 are followed. Hyperparameter tuning is done using the same approach as that for recursive PCA.

The next section looks at the application Gaussian mixture model which is a multimodal approach for fault detection.

### 2.1.3 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 number (see Sections 1.5.2.3 and 1.5.2.4). The best model is then selected using the mAB criterion (see Section 1.5.2.4) 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 **1-36**].
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 1.5.2.2].
4. Compute and retain the best GMM using the mAB method [see Section. 1.5.2.4].
5. Compute the monitoring statistics **NLPDF** and the global critical value $\text{NLPDF}_\alpha$ and the individual (local) critical values $\textbf{NLPDF}_{\boldsymbol{\alpha}}$ [see Section 1.5.4].

6. Obtain the validation data $\boldsymbol{D_v} \epsilon R^{\acute{n} \times m}$ (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 $\text{NLPDF}_{\alpha}$, trigger an alarm.

### 2.1.4 Application of PCA-based GMM

The initial stages involve computing the scores of the training data as outlined in PCA (Section 1.1.1). 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 2.1.3). The sequence of implementation is shown in the subsequent offline and online procedure.

**Offline (Training)**

1. Obtain the training data $\boldsymbol{D} \epsilon R^{n \times m}$.
2. Follow Procedure 2 to 5 outlined in PCA offline to compute the retained scores $\widehat{\boldsymbol{T}}$.
3. Follow Procedure 2 to 5 outlined in GMM offline on $\widehat{\boldsymbol{T}}$ 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 Procedure 11 to 12 outlined in PCA to compute the scores.
5. Follow Procedure 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 Procedure 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 $\text{NLPDF}_{\alpha}$, trigger an alarm.

The next section looks at adaptive PCA-based GMM which is the combination of adaptive PCA and GMMs.

## 2.2 Adaptive PCA-based GMM

### 2.2.1 Overview

Adaptive PCA-based GMM combines adaptive PCA methods with multimodal methods to make provision for slow and natural process changes that occur in a multimodal process. Like adaptive PCA, 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 principal components 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 $\boldsymbol{D}_b \in R^{c \times m}$ 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.

$\boldsymbol{D}_b \in R^{c \times m}$ is therefore obtained by accumulating new observations that exhibit NOC data over time. For a time instance, let $\boldsymbol{D}_k \in R^{n \times m}$ be the data window (initially set at the outset). Let $\boldsymbol{D}_1 \in R^{c \times m}$ 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 2.1 provides a conceptual diagram of how the PCA model is updated for new observations while Figure 2.2 shows how the initial GMM model is updated with the new block of observations.

(The illustrations are made in the input data space while implementation 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.

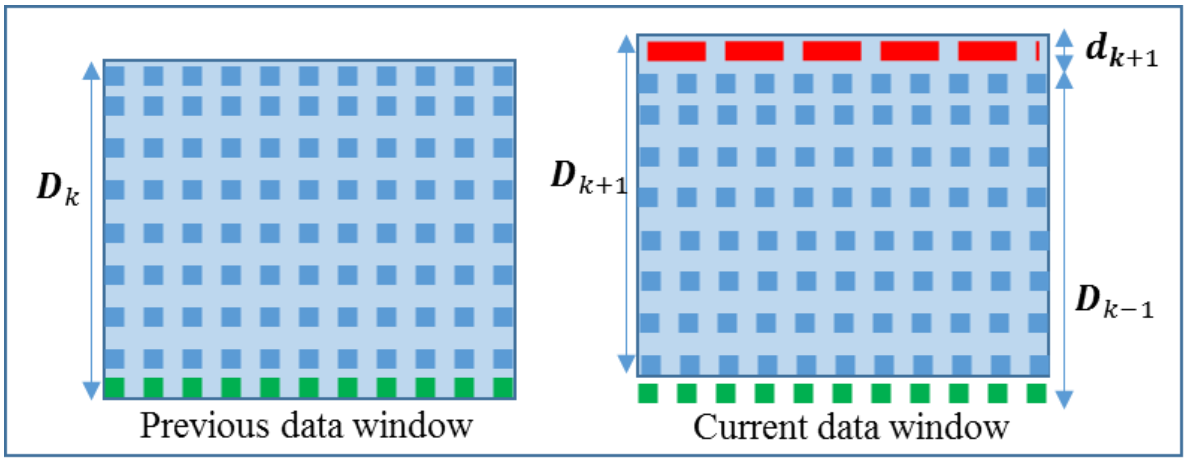


Figure 2.1: Sample-wise update of the PCA model for new observations.

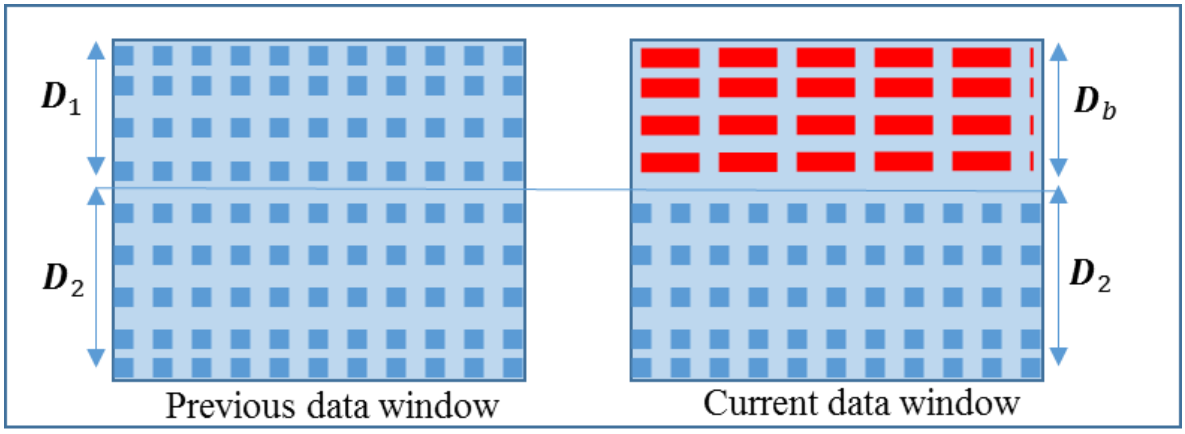


Figure 2.2: Block-wise update of the GMMs for new observations.

### 2.2.2 Algorithm

Adaptation of the PCA model parameters of the updated window follows the procedure outlined in MWPCA algorithm in Section 1.4.2 to compute the updated scores $\boldsymbol{T}_{k+1}$. The retained scores $(\widehat{\boldsymbol{T}})_{k+1}$ 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 $(\widehat{\boldsymbol{T}})_{k+1}$. The

monitoring statistics and thresholds $(\mathbf{NLPDF}_{\boldsymbol{\alpha}})_{k+1}$ are consequently updated as outlined in Section 1.6.2.

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.

### 2.2.3 Monitoring statistics and control limits

Adaptive PCA-based GMM employs the same monitoring statistics and detection thresholds as the GMM in Section 1.5.3 but with changing score matrix.

### 2.2.4 Fault detection

Adaptive PCA-based GMM employs the same fault detection procedure as shown for the GMM in Section 1.5.4.

### 2.2.5 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 observation 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 1.5.3)

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 $(\hat{\boldsymbol{t}}_{new} \mid (\boldsymbol{x}_{k+1})_{new}, (\widehat{\boldsymbol{P}})_{k+1})$
4. Using $(\boldsymbol{x}_{k+1})_{int} = (\boldsymbol{x}_{k+1})_{new}$, compute an intermediate score by Proj $(\hat{\boldsymbol{t}}_{int} \mid (\boldsymbol{x}_{k+1})_{int}, (\widehat{\boldsymbol{P}})_k)$
5. Compute old score by Proj $(\hat{\boldsymbol{t}}_{old} \mid (\boldsymbol{x}_{k+1})_{old}, (\widehat{\boldsymbol{P}})_k)$
6. Compute the NLPDF for all the scores (old, new and intermediate). Denoting $new$, $int$, or $old$ by **•,** the NLPDF for $\hat{\boldsymbol{t}}_{new}$, $\hat{\boldsymbol{t}}_{int}$ and $\hat{\boldsymbol{t}}_{old}$ are calculated as:

$$\text{NLPDF}(\hat{\boldsymbol{t}}_{\bullet}) = -\log \sum_{j=1}^{r} \varrho_j g_j \left( \hat{\boldsymbol{t}}_{\bullet} \,\middle|\, \boldsymbol{\mu}_j, \boldsymbol{S}_j \right) \qquad \textbf{2-8}$$

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:

$$p(g_j | \hat{\boldsymbol{t}}_{\bullet}) = \frac{\varrho_j g_j(\hat{\boldsymbol{t}}_{\bullet} | \boldsymbol{\mu}_j, \boldsymbol{S}_j)}{\sum_{p=1}^{r} \varrho_p g_p(\hat{\boldsymbol{t}}_{\bullet} | \boldsymbol{\mu}_p, \boldsymbol{S}_p)} \qquad \textbf{2-9}$$

Here, • denotes $new$, $int$, or $old$.

#### 2.2.5.1 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$\left(\boldsymbol{t}_{\boldsymbol{A}_{int}}\right)$ 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 $\boldsymbol{D}_{\boldsymbol{b}}$. $\boldsymbol{D}_k = \boldsymbol{D}_{k+1}$ for application to next observation.
5. Size the accumulated data $\boldsymbol{D}_{\boldsymbol{b}}$. 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 procedures.

**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 **1-36**].
3. Run PCA on $\boldsymbol{D}_k$ to produce $(\widehat{\boldsymbol{T}})_k$.
4. Use $(\widehat{\boldsymbol{T}})_k$ 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 1.5.2.2].
6. Compute and retain the best GMM using the mAB method [see Section. 1.5.2.4].
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 **1-19** to **1-21**].
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 **1-22** to **1-24**].
9. Compute and retain the principal components $(\widehat{\boldsymbol{P}})_{k+1}$ and variances $(\widehat{\boldsymbol{\lambda}})_{k+1}$ of $\boldsymbol{C}_{k+1}$ [Equation **1-8**].
10. Compute the retained scores $(\widehat{\boldsymbol{T}})_{k+1}$ of the new window [Equation **1-6**].
11. Compute the monitoring statistic $(\mathbf{NLPDF})_{k+1}$ and cluster membership of each $\boldsymbol{t}$ in $(\widehat{\boldsymbol{T}})_{k+1}$ 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 **2-10** to **2-11**].
14. Update model parameters if appropriate (using the stated Procedure in 1 to 5 of the model update condition).

Hyperparameter tuning is done using the same approach as that for recursive PCA (in Section 2.1.1).

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

## 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 |
| 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 |
| 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 |
|---|---|
| $\boldsymbol{D}$ ($\boldsymbol{d}$) | Input data matrix (vector) |
| $\boldsymbol{X}$ ($\boldsymbol{x}$) | Normalized data matrix (vector) |
| $\boldsymbol{C}$ | Correlation matrix |
| $\boldsymbol{S}$ | Covariance matrix |
| $m$ | Number of process variables |
| $\boldsymbol{P}$ | Matrix of eigenvectors |
| $\boldsymbol{T}$ | PCA score matrix |
| $\boldsymbol{E}(\boldsymbol{e})$ | Reconstruction error matrix (vector) |
| $\hat{\boldsymbol{t}}^2(\hat{t}^2)$ | Modified Hotelling T-square statistic vector (scalar) |
| $\boldsymbol{q}(q)$ | SPE statistic vector (scalar) |
| $\upsilon$ | Number of retained principal components |
| $L_\upsilon$ | Cumulative variance explained by $\upsilon$ principal components |
| $r$ | Number of modes |
| $r_{\max}$ | Maximum plausible number of modes |
| $\hat{\boldsymbol{Z}}(\hat{\boldsymbol{z}})$ | Latent variables matrix (vector) |
| $\boldsymbol{\mu}$ | Mean vector |
| $z_\alpha$ | Normal deviate corresponding to the $(1-\alpha)$ percentile |
| $z$ | Consecutive number of observations required to trigger an alarm |
| $\boldsymbol{\sigma}$ | Standard deviation vector |
| $n$ | Number of observations |
| $p(.)$ | Probability density function |
| $\boldsymbol{\theta}^t$ | Initial GMM parameter estimates |
| $\boldsymbol{\theta}^*$ | Maximum likelihood estimates for GMM parameters |
| $\varrho$ | Mixing weight of a Gaussian cluster |
| $\boldsymbol{\lambda}$ | Vector of eigenvalues |
| $\eth_{ij}$ | Responsibility of cluster $j$ for observation $i$ |
| $\tau$ | Tolerance level for EM |
| $\alpha$ | Significance level of probability distribution |
| $\widehat{\boldsymbol{P}}$ | Retained principal components vector |
| $\hat{\boldsymbol{\lambda}}$ | Retained eigenvalues vector |
| $\widehat{\boldsymbol{T}}$ | Retained score matrix |
| $\widehat{\boldsymbol{X}}(\hat{\boldsymbol{x}})$ | Reconstructed data matrix (vector) |
| $\boldsymbol{\Sigma}_\bullet$ | Diagonal matrix of $\boldsymbol{\sigma}_\bullet$ |
| $F_\alpha(\upsilon, n-\upsilon)$ | The upper $100\alpha$ % critical point of the F-distribution |
| $g_j$ | Gaussian distribution of cluster $j$ |
| $L(.)$ | Likelihood function |
| $\boldsymbol{d}_1$ | Oldest observation of in input matrix |
| $\gamma$ | Number of parameters of the means and mixing proportions |
| $\mathfrak{g}$ | Number of parameters in the covariance matrix |
| $(\hat{t}^2)_\alpha$, | Modified Hotelling T-square statistic critical value |
| $\mathbf{NLPDF}_{\boldsymbol{\alpha}}$ | Negative-logarithm PDF critical value vector |
| $\dot{n}$ | Number of observations for validation data |
| $(.)_{k-1}$ | Previous ( or down-dated) value |
| $(.)_k$ | Current value |
| $(.)_{k+1}$ | Next (or updated) value |

## 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.

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.

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.

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

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 & chemical engineering*, *28*(8), pp.1377-1387.

Choi, S.W., Martin, E.B., Morris, A.J. and Lee, I.B., 2005. Fault detection based on a maximum-likelihood principal component analysis (PCA) mixture. *Industrial & engineering chemistry research*, *44*(7), pp.2316-2327.

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.

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

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.

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. Graphical representation of data using principal components. *Principal component analysis*, pp.78-110.

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 & Sons.

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.

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.

Russell, E.L., Chiang, L.H. and Braatz, R.D., 2000. Data-driven techniques for fault detection and diagnosis in chemical process. *Advances in Industrial Control*.

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.

Schwarz, G., 1978. Estimating the dimension of a model The Annals of Statistics 6 (2), 461–464. *URL: http://dx. doi. org/10.1214/aos/1176344136*.

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

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

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

Wang, Y., Seo, H. and Jeon, S., 2008, September. Automatic Transition Detection of Segmented Motion Clips Using PCA-based GMM Method. In *Cyberworlds, 2008 International Conference on* (pp. 567-572). IEEE.

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.

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.