# Decentralized and Dynamic Fault Detection Using PCA and Bayesian Inference

A. Sanchez-Fernandez, M.J. Fuente, G.I. Sainz-Palmero
*Department of System Engineering*
*and Automatic Control, EII*
*University of Valladolid*
Valladolid, Spain
{alvsan, mjfuente, gresai}@eii.uva.es

***Abstract*—This paper proposes a dynamic and decentralized fault detection method. The plant is divided in groups whose members are selected using linear and non-linear modelling techniques. In each group a Principal Component Analysis model does the fault detection, including delayed data to get a dynamic method. Then, a central node fuses the results of each group, using Bayesian Index Criterion (BIC), to get a global detection outcome. The method was tested on a widely used benchmark and compared with other proposal to check its effectiveness.**

***Index Terms*—Fault detection, Dynamic principal component analysis, Decentralized process monitoring, Decision fusion**

## I. INTRODUCTION

The objective of process control includes not only making the system work under desired conditions, keeping it stable and safe, but also detecting faults, i.e., any deviation from the expected behaviour. Any possible fault must be detected quickly, because any malfunction of the plant can lead to great losses in terms of reduced plant performance, product quality losses, unscheduled stops, etc. Furthermore, these faults can make the plant work under unsafe conditions for employees, other installations, etc., so it is crucial to perform this task effectively.

In general, data driven process monitoring methods such as principal component analysis (PCA) [1] or partial least squares (PLS) [2] have become very popular; this is because large amounts of data can been generated and collected from the plant, which can be used to extract useful information about the current state of the process, and detect faults. However, most multivariate statistical monitoring methods based on PCA assume implicitly that the observations at one time are statistically independent of observations at a past time, and this situation does not occur in industrial plants, where it is expectable that exist some degree of auto-correlation (or time-correlation) among the variables. In order to take into account this time-correlation and to capture the process dynamic, [3] proposes dynamic PCA (DPCA), which performs the PCA method using an augmented matrix with time-lagged variables. While [4]–[7] use Canonical variate analysis (CVA) for dynamic process monitoring where both past data and future measurements are used to estimate the process state space model and to build a fault detection scheme.

As industrial plants tend to increase their level of automation the number of sensors and, therefore, the amount of collected data is huge, this implies that the requested computational power to process this amount of data has grown significantly. For this reason, the fault detection have became highly time and resources consuming. In small plants fault detection can be developed using a central processor which collects all the information from the sensors, but in complex plants, with lot of sensors, this central processor must be able to deal with this vast amount of data, which is not always possible. One way to afford this problem is dividing the plant in blocks, each one containing a specific number of measured variables in it and assigning one small processor in every block, this is called decentralized approach. Here, each local processor does the fault detection task with its own sensors and send the result to a central processor which only has to fuse the local results to obtain a global detection result.

Several researches have been carried out regarding decentralized monitoring ( [8]–[12]). The most important step in this technique is to establish the criteria used to divide the plant. Some authors opted for one-variable blocks [13], while other authors constructed blocks that group more than one variable ( [8], [10], [12], [14]). In the second option (multivariable blocks), there are some possibilities: the previous knowledge of the plant can be used to group variables that have any physical connection ( [9], [15]), however, as not always is available enough information about the plant to do the decentralization, many authors have tested different methods to divide the plant only analysing measured data ( [8], [12], [14], [16]), obtaining significant relations between variables.

Once the local fault detection models obtain a result, it is necessary to fuse all these results in order to get an unique decision for the whole plant. There are different ways to do that as [8] that proposed Maximum Entropy, Multicriteria decision making (MCDM) methods as Ordered Weighted Average (OWA) operators [17] are another option, also ( [12], [14]) used Bayesian Inference Criterion (BIC) for their Fault Detection and Diagnosis (FDD) method.

In this paper we propose a decentralized and dynamic fault detection method. The plant first is decomposed into blocks

The authors would like to express their gratitude to the European Commission and the Spanish Ministry of Economy and Competitivity for financial support through the project DPI2015-67341- C2-2-R.

based on the study of the correlation between the system variables, using two methods, a linear one: the Sparse Partial Least Squares (SPLS) method and a non-linear one (a neural network). So, every measured variable has its own group where it is accompanied by those variables who have a high level of correlation with it. The local processors count in it with a dynamic PCA-based fault detection model which analyses new observations in order to detect anomalies. Finally, the central processor will fuse all the local detection results using the Bayesian Inference Criterion. This will lead to a global detection result.

This document is organized as follows: Section II contains the explanation of decentralized approach, the DPCA fault detection method, and the BIC fusion method. Section III presents the proposal of the distributed fault detection method. The tests developed using the proposed method over the Tennessee Eastman Plant and its results are included in Section IV. Finally, conclusions and future work appear in Section V.

## II. PRELIMINARIES

### A. Decentralized fault detection

As opposed to centralized fault detection, where all the tasks are carried out by an unique central processor, decentralized fault detection implies to divide the measured data in groups of variables, whose respective data are processed by separate units. The way in which the plant is divided is a crucial task and can be done using different strategies:

- Completely decentralized decomposition: each variable has its own group, without including any other variable, only it is possible to include lagged samples of this variable. In this case the information about the correlation between variables is lost [8].
- Multi-block decomposition:
  - One block per variable: here each measured variable has its own block. Each block includes one measured variable and those ones that share some relation with it.
  - Less blocks than variables: after analysing the relations between variables, some blocks are constructed with those variables that are more related. If there are variables not related with any other, they can be discarded or grouped in one extra block.

After the plant decomposition, the fault detection task is carried out in each block. This can be performed using different techniques, one of them is Dynamic PCA (DPCA).

### B. Dynamic PCA

*1) PCA:* After measuring the variables of a process in normal operation conditions, a matrix $\mathbf{X}(n \times m)$ can be arranged, whose rows represent the observations and the columns represent the variables. After doing a z-score normalization, the covariance matrix is obtained and decompose using singular value decomposition (SVD) as:

$$S = \frac{1}{(n-1)} X^T X = V \Lambda V^T \tag{1}$$

The eigenvalues of $\mathbf{S}$ are included in the diagonal of $\mathbf{\Lambda}$, sorted in decreasing order and representing the data variance included in each principal component. Matrix $\mathbf{V}$ contains the eigenvectors. The principal components are obtained as:

$$T = XP \tag{2}$$

where $\mathbf{T}$ are the principal components and the loading matrix $\mathbf{P}$ is formed using the first $a$ columns of matrix $\mathbf{V}$. The value of $a$ is selected as the one that maximizes the data variance included in the model and maximizes the dimensionality reduction. $\mathbf{T}$ is also known as the scores matrix, and its columns represent the $a$ principal components of $\mathbf{X}$.

*a) Fault detection:* When the PCA method is used to monitor the process, the $T^2$ and $Q$ statistics are used to detect anomalies [1].

$T^2$ statistic (or Hotelling's Statistic) is calculated as:

$$T^2 = x^T P \Lambda_a P^T x \tag{3}$$

where $\mathbf{\Lambda_a}$ includes the $a$ first rows and columns of $\mathbf{\Lambda}$. A fault is detected if:

$$T^2 > T^2_\alpha = \frac{(n^2-1)a}{n(n-a)} F_\alpha(a, n-a) \tag{4}$$

For a certain $\alpha$ significance level, where $F_\alpha(a, n-a)$ is the critical value $(100(1-\alpha))\%$ of the F distribution, with $a$ and $n$-$a$ degrees of freedom.

The $Q$ statistic (or squared prediction error, SPE), related with the goodness of fit and the system noise, is obtained as:

$$Q = [(I - PP^T)x]^T[(I - PP^T)x] \tag{5}$$

where $\mathbf{I}$ is the square identity matrix of $m$ dimension, and $x$ is a new measure which is been tested. The system is out of control if $Q$ value is greater than its threshold $Q_\alpha$, which is calculated, for a significance level of $\alpha$, as:

$$Q_\alpha = g\chi^2_\alpha(h) \tag{6}$$

where $g = \frac{S}{2\mu}$ and $h = \frac{2\mu^2}{S}$, being $\mu$ and $S$ the mean and the variance of $Q$ under non-faulty conditions [18].

In order to avoid false alarms caused by noise, disturbances, etc. a number of consecutive anomalous observations is required to indicate that the system is faulty. This value has to be adjusted looking for a value that reduces the false alarms occurrence without delaying the detection time too much.

*2) DPCA method:* Assuming that each variable has a certain level of time-correlation, that is, the current value of a variable is correlated with some past values of itself as well as with past values of the other variables, it is necessary to use a dynamic method. The Dynamic PCA (DPCA) is a PCA method that works with the augmented data matrix, $\mathbf{X_a}$ which is formed using current and past values of the variables:

$$X_a = \begin{bmatrix} X_{l+1}^T & X_l^T & \dots & X_1^T \\ X_{l+2}^T & X_{l+1}^T & \dots & X_2^T \\ \vdots & \vdots & \ddots & \vdots \\ X_n^T & X_{n-1}^T & \dots & X_{n-l}^T \end{bmatrix} \tag{7}$$

where $X_t$ is the vector of measures in time $t$, and $l$ is the number of lags included in the augmented matrix. $l$ can be selected doing tests with different values and selecting the one that gives the best performance, for example, using Akaike Information Criterion (AIC) ( [4], [19]).

## C. Sparse PLS

The Partial Least Square method (PLS) was introduced in [20] which is based in principal components and regression [21]. Let $\mathbf{Z}$ be a matrix with $n$ measures (rows) of $m$ variables (columns), this method considers two subsets of variables whose measures are included in matrices $\mathbf{X}$ (predictors) and $\mathbf{Y}$ (responses). A linear model is adjusted using least squares over new features that are linear combination of the original ones: $\mathbf{X}$ and $\mathbf{Y}$.

$$\begin{aligned} X &= TP^T + E \\ Y &= UQ^T + F \end{aligned} \tag{8}$$

where $\mathbf{X}$ is the predictors matrix $(n \times m)$, $\mathbf{Y}$ is the responses matrix $(n \times p)$, $\mathbf{T}$ is the projection of $\mathbf{X}$, and $\mathbf{U}$ is the projection of $\mathbf{Y}$ ($\mathbf{X}$ and $\mathbf{Y}$ scores, respectively), and their dimension is $(n \times l)$; $\mathbf{P}$ and $\mathbf{Q}$ are $(m \times l)$ and $(p \times l)$, respectively, the loading matrices. Finally, $\mathbf{E}$ and $\mathbf{F}$ represent the error terms. They are supposed to be independent and identically distributed random normal variables.

This model guarantees that the covariance between $\mathbf{T}$ and $\mathbf{U}$ are maximum. The way in which this method is used to perform a regression is:

$$Y = XB + G \tag{9}$$

where $\mathbf{G}$ is the residuals matrix, and $B$ is the vector of coefficients for the PLS-regression, and it is calculated using the well known algorithm of the non-linear iterative partial least squares (NIPALS) until all variance in the data structure is explained [22].

Sparse Partial Least Squares (SPLS) is a method based in Partial Least Squares (PLS) whose objective is to obtain a certain number of coefficients equal to zero in vector $B$. This means that the method selects the most relevant predictors in the regression, making easier the understanding of the model, as well as allowing to perform a variable selection. A detailed explanation of this method can be found in [23].

## D. Neural Networks

Neural networks are a useful non-linear modelling technique [24]. This model processes some inputs to return one or various outputs. An artificial neural network is formed by computing units, called neurons, which are connected to each other, forming a network. The strength of each connection, or weight, is updated iteratively as the network is trained, so in effect it learns patterns provided with the data. The neurons are organized in layers; the first layer consists of neurons that represent the inputs to the network. The hidden layer consists of a number of neurons that apply a nonlinear function to the inputs, and the output layer that consists of the predictions made by the artificial neural network. As an example, a neural net with three inputs, two hidden layers with four and three neurons, respectively, and two outputs is shown in Figure 1.

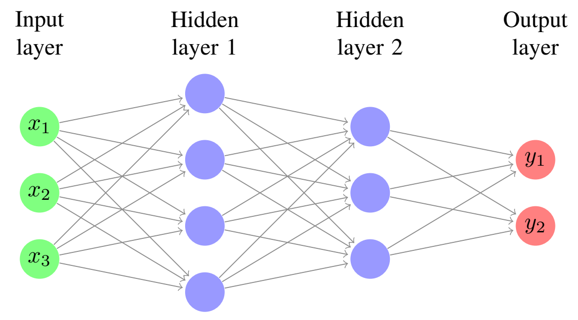

Figure 1. Neural network example

## E. Bayesian Inference Criterion (BIC)

In centralized approach, the fault detection task gives, for each statistic, one value per sample, but if the plant is divided in blocks, each one of them returns its own statistics. As the system is unique, it is necessary to fuse local results for each statistic to get one global result per statistic.

One option to fuse the results is the Bayesian Inference Criterion (BIC) [14], which provides a global value for each statistic. The fault posterior probability for a certain statistic $ST$ (which can be $T^2$ or $Q$) in the block $i$ ($i = 1, 2, \dots, m$) is:

$$P_{(F|x_i)} = P_{(x_i|F)} P_{(F)} / P_{(x_i)} \tag{10}$$

where

$$P_{(x_i)} = P_{(x_i|N)} P_{(N)} + P_{(x_i|F)} P_{(F)} \tag{11}$$

where $N$ and $F$ are the normal and abnormal state of the system, respectively. $P_{(N)}$ and $P_{(F)}$ represent the prior probabilities for the system working under normal and faulty conditions. The first one is set to an $\alpha$ value between 0 and 1, and the second is $1 - \alpha$. Moreover:

$$P_{(x_i|N)} = e^{-ST/ST_{i,lim}}, P_{(x_i|F)} = e^{-ST_{i,lim}/ST} \tag{12}$$

$ST_{i,lim}$ is the threshold of the statistic $ST$ in the $i$-th block.

Finally, the results of each block are fused using the $BIC$ index:

$$BIC_{ST} = \sum_{i=1}^{m} \frac{P_{(x_i|F)} P_{(F|x_i)}}{\sum_{i=1}^{m} P_{(x_i|F)}} \tag{13}$$

A fault is detected for a certain statistic if its respective $BIC$ value is over $(1-\alpha)$. This thresholds must be readjusted after applying the model with the test data.

## III. Distributed Fault Detection with PCA

The fault detection method presented in the above section is considered as a centralized DPCA method. In this case, all the sensor measurements are sent to a central node and processed to detect faults in the whole plant. Now, in this section, distributed DPCA methods are applied to detect faults. The first step is to decompose the plant into different blocks and then to perform a DPCA method to detect faults locally in each block, and, finally, a central processor fuse all the local detection results using the BIC index to take a global decision. The methodology adopted is outlined in the following steps.

*Step 1.- Plant decomposition*: in this paper two ways to do the decentralization of the plan, based only on data, are proposed: SparsePLS and Neural Nets.

*a) Sparse PLS:* The way in which the plant was decomposed here is obtaining a regression model for each variable using Sparse PLS method. So in this case, the matrix $\mathbf{X}$ is constructed, and a SPLS regression model is calculated taking as output each of the system variables, i.e., $\mathbf{Y} = x_i,\ i = 1, ..., m$ and as predictor the matrix $\mathbf{X}$ without the corresponding $i$-th variable that it is being modeled. In each regression model some values of sparsity must be tested, i.e., how many coefficients in the respective regression model $\mathbf{B_i}$ are considered equal to zero, such that at the same time selecting the sparse model that better fits the data. This problem can be solved using the Root Mean Squared Error (rMSE). So a block is created for each variable together with the variables with a non-zero coefficient in the respective regression coefficient vector $\mathbf{B_i}$. For a plant with $m$ variables, this results in a block per each measured variable, i.e., a decomposition of $m$ groups.

*b) Neural Nets:* As in the SPLS case, one model per measured variable is created. In this case a neural network is generated and trained, testing different values for parameters (number of hidden layers, number of neurons in each layer, etc.) and selecting the net which best fits to the data (as previous method comparing rMSE results). After that, the output is calculated using the inputs and the weights of the net. The proposed method tries to find the influence of each variable in the output. This is done analysing the path of each variable along the network: when a variable enters to a neuron is multiplied by a weight, and the output of this neuron is multiplied by another weight to enter into the next neuron in the net, so a score is assigned to each variable multiplying all the weights of the net that affect this variable. Finally, the variables with highest scores are included in the group. As a neural net is able to model non-linear relations, it is expected this method captures more information than SPLS.

*Step 2.- Distributed Dynamic PCA (DDPCA) training*: Each block has its own fault detection model based on Dynamic PCA. It is necessary to define how many lags, $l$, are going to be used to create the augmented matrix $\mathbf{X_a}$ of Eq. (7). A statistically justified method can be used for selecting this number, $l$, however it is possible to prove with different values, using faulty train data, and choosing the number of lags that gets the best performance in detection (lowest detection delay, highest number of faults detected, absence of false alarms). After doing the training process with selected value, local DPCA models are obtained.

Now, it is necessary to calculate the thresholds for the statistics $T^2$ and $Q$, i.e., to calculate $\{T^2_\alpha$ and $Q_\alpha\}$ for each block, taking into account that it is necessary to consider a consecutive number of observations exceeding the threshold in order to detect a fault. The thresholds of these statistics are calculated theoretically as equations (4) and (6) respectively, and after that these limits were tuned experimentally for an imposed significance level (ISL or $\alpha$) of $1\%$. This value is the expected percentage of alarms for the system under normal operation conditions.

These two steps are calculated off-line.

*Step 3.- DDPCA fault detection*. This step is calculated on-line. When new observations are collected from the plant, each local DPCA model calculates its current values for the statistics $T_i^2$ and $Q_i$, for $i = 1, ..., m$, the number of blocks. Then, a central processor collects all local results and processes them. A global outcome, for each statistic, is obtained fusing these local results using index BIC as it was explained in Section II-E. Thus, BIC index for $T^2$ and $Q$ are obtained, and the system is considered under control if both BIC indexes are under their thresholds for a certain confidence level $\alpha$.

An overview of the process is shown in Algorithm 1

## IV. Illustrative Example

To validate the proposed method it was tested on a widely used benchmark: Tennessee Eastman Process [25]. This is a well-known process in the fault diagnosis field, as it is used to test new methods about fault detection, fault identification, etc. [6], [16], [26]–[29]. The process represents a chemical plant with 52 variables measured each 3 minutes. A broad description of these variables can be found in [25]. The diagram of this plant is shown in Fig. 2. There are available faultless train and test data. Also, there are 21 faulty train and test datasets. Train and test datasets comprise 500 and 960 samples, respectively. These 21 faults are summarized in Table I.

### A. Experimental setup

The proposed methods were tested with different parameters: the number of lags, the significance value for BIC, the number of consecutive anomalous observations to detect a fault, the variance retained by DPCA, the $\alpha$ used to obtain PCA thresholds and the thresholds used in the variable assign task, in each method. These combinations of parameters were used with train data and then with test data, adjusting, if necessary, the $\alpha$ value for BIC to avoid the appearance of false alarms. The first step, Plant decomposition, was done with SPLS regression and Neural Networks modelling looking for the model with the lowest rMSE value for different values

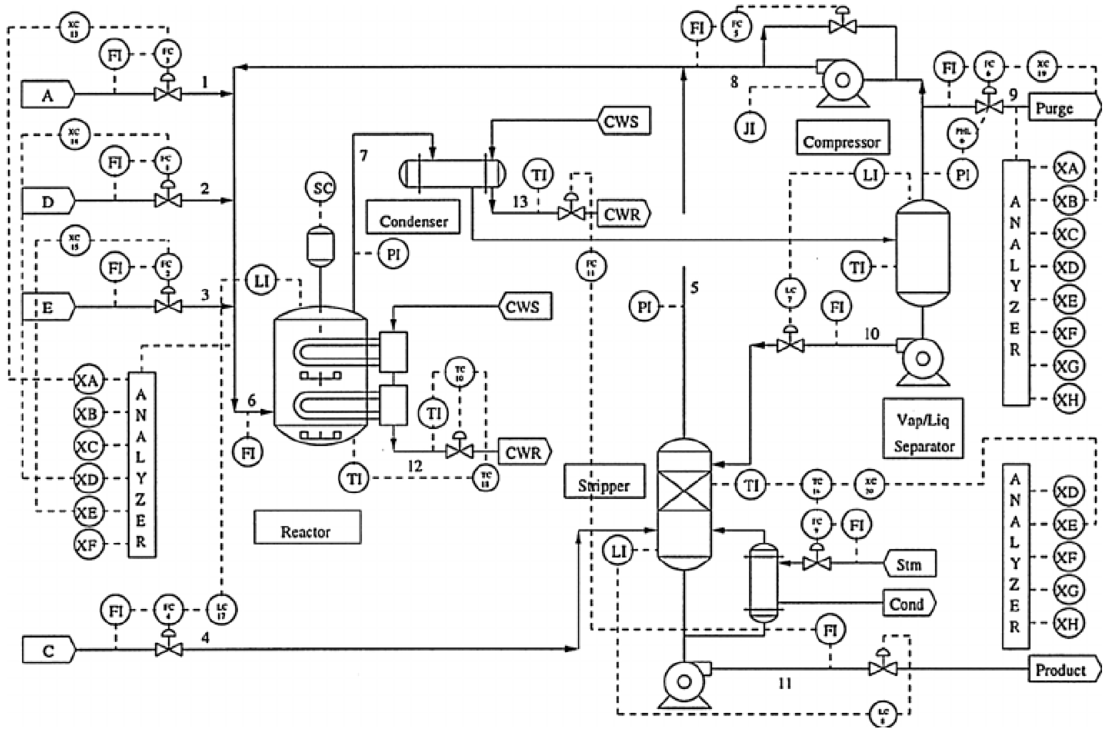

Figure 2. Tennessee Eastman Process diagram

Table I
TEP FAULTS

| Fault # | Description | Type |
|---|---|---|
| 1 | A/C feed ratio, B composition constant (Stream 4) | Step |
| 2 | B composition, A/C ratio constant (Stream 4) | Step |
| 3 | D feed (Stream 2) | Step |
| 4 | Reactor cooling water inlet temperature | Step |
| 5 | Condenser cooling water inlet temperature | Step |
| 6 | A feed loss (Stream 1) | Step |
| 7 | C header pressure loss-reduced availability (Stream 4) | Step |
| 8 | A, B and C compositions (Stream 4) | Random variation |
| 9 | D feed temperature (Stream 2) | Random variation |
| 10 | C feed temperature (Stream 4) | Random variation |
| 11 | Reactor cooling water inlet temperature | Random variation |
| 12 | Condenser cooling water inlet temperature | Random variation |
| 13 | Reaction kinetics | Slow drift |
| 14 | Reactor cooling water valve | Sticking |
| 15 | Condenser cooling water valve | Sticking |
| 16 | Unknown | - |
| 17 | Unknown | - |
| 18 | Unknown | - |
| 19 | Unknown | - |
| 20 | Unknown | - |
| 21 | Stream 4 valve | Sticking |

of parameters (non-zero elements in regression's vector for SPLS, and number of hidden neurons in Neural Net modelling) for each variable. Then, the second step and third step are carried out: DDPCA was applied looking for a combination of parameters that achieves the best fault detection performance.

So, after some tries, the configuration with the best result in terms of absence of false alarms, number of faults detected, and best detection times was, for SPLS method, 5 lags for the augmented matrix, 0.9 for $\alpha$ parameter in BIC criteria, 5 consecutive anomalous observations to detect a fault, 60% of variance inside the DPCA, local DPCA thresholds were adjusted for a confidence level of 99% and the SPLS regression coefficients of each variable must be over $10^{-5}$ to include this variable in a block. The BIC significance level was readjusted, as said before, using faultless test data, getting these values: 0.89 for $T^2$ and 0.88 for $Q$.

In case of NeuralNet DPCA method, the parameters used were 2 lags for the augmented matrix, 0.9 for $\alpha$ in BIC, 3 consecutive anomalous observations to detect faults, 60% of variance included in DPCA, local DPCA thresholds were obtained to get a confidence level of 99% and the coefficients of each variable inside the neural net must be over the mean of the maximum and minimum values of the whole variables to be included inside a block. The BIC $\alpha$ did not need to be readjusted using test data.

## B. Results

As the proposed method combines DPCA fault detection method with a decentralized approach, it was compared with a centralized approach that uses DPCA to detect faults. The results of this last method were taken from [4]. This comparison was done in terms of the fault missed detection rate (MDR), that denotes that faulty data are misclassified as faultless data, the fault detection delay, the false alarm rate (FAR), which measures the number of non-faulty observations identified as anomalous, and the number of faults detected.

First of all, the Missed Detection Rate (MDR) are shown in Tables II, and III. As this index counts, in percentage, the number of anomalous observations not detected as faults by each method, it acts as a sensitivity test.

**Algorithm 1** *Correlation based decentralization with DPCA fault detection*

```
1: **Off-line steps:**
2: Normalize train data (faultless)
3: **if** method: SPLS **then**
4:     **for** i=1 to m **do**                         ▷ For each variable
5:         Models SPLS with varied sparsity
6:         Select model with lowest rMSE
7:     **end for**
8: **end if**
9: **if** method: NeuralNet **then**
10:     **for** i=1 to m **do**                        ▷ For each variable
11:         Models Neural Nets with varied parameters (layers, neurons)
12:         Select model with lowest rMSE
13:     **end for**
14: **end if**
15: Group unselected variables in an extra group
16:
17: **for** i=1 to m **do**                  ▷ For each variable or block
18:     Train local DPCA models with different lags
19:     Select DPCA model with best performance
20: **end for**
21:
22: **On-line steps:**
23: **for** Each new observation **do**
24:     **for** i=1 to m **do**              ▷ For each variable or block
25:         Calculate $ST^i$ = { $T^2$, $Q$ }          ▷ Local statistics
26:     **end for**
27:     **for** s={$T^2$, $Q$}  **do**                 ▷ For each statistic
28:         $\boldsymbol{BIC}_s = f(ST_1, ST_2, \ldots, ST_m)$
29:         **if** $\boldsymbol{BIC}_s \geq (1-\alpha)$ **then**
30:             *Fault detected with statistic s*
31:         **else**
32:             *Normal condition with statistic s*
33:         **end if**
34:     **end for**
35: **end for**
```

In Tables IV and V are the results of the detection time for these methods. The time when each method points the start of the fault it is shown, but, as it is needed to detect a certain number of anomalous observations, the real detection time is higher than the results of these tables for the three methods.

As can be seen in the Tables II, III, IV and V. MDR results revealed the great results of Neural Net method with $T^2$, having the lowest value in 19 out of 21 faults. Also, SPLS is still better that central DPCA, being the best of the comparison in 6 faults. This method, SPLS, showed better outcome using $Q$, where it is the best in 19 faults. Neural Net performance was also good, as it is the best in 7 faults, and not far from SPLS in the remaining cases. Anyway, both methods have better behaviour than central DPCA.

In the delay results, the situation is very similar, Neural Net is the best with $T^2$ (it had the fastest time in 18 faults), the second method is SPLS (which is the best in 13 fault). Central DPCA got poor results with this statistic. SPLS, once again, is the best using $Q$ (in 19 faults it got the fastest time), and Neural Net achieved the second best result (12 faults with the best delay).

In summary, NeuralNet method gave the best MDR and Delay results with $T^2$, obtaining a good result with $Q$, always ahead of Central DPCA. This shows the greater performance of this method with $T^2$ statistic. This may be due to the ability of this method to extract non-linear characteristics of the plant to construct the groups. Also, as $T^2$ statistic monitors the behaviour of the model, while $Q$ monitors the noise, disturbances, etc. [1], if the model is better, it is expected

Table II
MISSED DETECTION RATE (MDR), IN %. $T^2$

| Fault | SPLS DPCA | NNET DPCA | DPCA |
|---|---|---|---|
| 1 | **0** | 0.13 | 0.6 |
| 2 | **1.13** | **1.13** | 1.9 |
| 3 | 96.61 | **94.11** | 99.1 |
| 4 | 63.74 | **27.44** | 93.9 |
| 5 | 73.15 | **70.30** | 75.8 |
| 6 | **0** | **0** | 1.3 |
| 7 | **0** | **0** | 15.9 |
| 8 | 2.26 | **0.88** | 2.8 |
| 9 | 97.99 | **93.61** | 99.5 |
| 10 | 44.54 | **38.22** | 58 |
| 11 | 55.33 | **43.98** | 80.1 |
| 12 | 0.50 | **0.38** | 1 |
| 13 | **4.27** | 4.39 | 4.9 |
| 14 | **0** | **0** | 6.1 |
| 15 | 88.58 | **83.08** | 96.4 |
| 16 | 60.73 | **48.75** | 78.3 |
| 17 | 12.30 | **8.77** | 24 |
| 18 | 11.04 | **9.77** | 11.1 |
| 19 | 99.75 | **97.74** | 99.3 |
| 20 | 47.30 | **42.11** | 64.4 |
| 21 | 55.83 | **55.51** | 64.4 |

Table III
MISSED DETECTION RATE (MDR), IN %. $Q$

| Fault | SPLS DPCA | NNET DPCA | DPCA |
|---|---|---|---|
| 1 | **0** | **0** | 0.5 |
| 2 | **0.753** | 1.25 | 1.5 |
| 3 | **96.49** | 98.50 | 99 |
| 4 | **0** | **0** | **0** |
| 5 | **73.53** | 75.56 | 74.8 |
| 6 | **0** | **0** | **0** |
| 7 | **0** | **0** | **0** |
| 8 | **1.51** | 2.01 | 2.5 |
| 9 | **97.11** | 98.37 | 99.4 |
| 10 | **36.01** | 38.72 | 66.5 |
| 11 | 10.16 | **8.90** | 19.3 |
| 12 | **0.38** | 0.50 | 2.4 |
| 13 | **4.52** | 4.76 | 4.9 |
| 14 | **0** | **0** | **0** |
| 15 | **90.46** | 96.24 | 97.6 |
| 16 | **50.06** | 54.39 | 70.8 |
| 17 | **2.38** | 2.88 | 5.3 |
| 18 | **9.16** | 9.65 | 10 |
| 19 | **31.74** | 75.31 | 73.5 |
| 20 | **32.99** | 38.97 | 49 |
| 21 | 51.82 | **47.49** | 55.8 |

Table IV
DETECTION DELAY, IN SAMPLES. $T^2$

| Fault | SPLS DPCA | NNET DPCA | DPCA |
|---|---|---|---|
| 1 | **0** | 1 | 6 |
| 2 | **9** | **9** | 16 |
| 3 | 81 | **40** | |
| 4 | 58 | **0** | 151 |
| 5 | **0** | **0** | 2 |
| 6 | **0** | **0** | 11 |
| 7 | **0** | **0** | 1 |
| 8 | 18 | **7** | 23 |
| 9 | **0** | **0** | |
| 10 | 33 | **22** | 101 |
| 11 | **9** | **9** | 195 |
| 12 | **0** | **0** | 3 |
| 13 | **34** | 35 | 45 |
| 14 | **0** | **0** | 6 |
| 15 | **572** | 573 | |
| 16 | 30 | **0** | 199 |
| 17 | **20** | **20** | 28 |
| 18 | 91 | **83** | 93 |
| 19 | | **8** | |
| 20 | **78** | **78** | 87 |
| 21 | 450 | **415** | 522 |

Table V
DETECTION DELAY, IN SAMPLES. $Q$

| Fault | SPLS DPCA | NNET DPCA | DPCA |
|---|---|---|---|
| 1 | **0** | **0** | 5 |
| 2 | **7** | 10 | 13 |
| 3 | | **86** | |
| 4 | **0** | **0** | 2 |
| 5 | **0** | **0** | 2 |
| 6 | **0** | **0** | 1 |
| 7 | **0** | **0** | 1 |
| 8 | **12** | 16 | 21 |
| 9 | 358 | **3** | |
| 10 | **31** | 32 | 50 |
| 11 | **3** | **3** | 7 |
| 12 | **0** | **0** | 8 |
| 13 | **36** | **36** | 40 |
| 14 | **0** | **0** | 1 |
| 15 | **233** | 571 | |
| 16 | **13** | 16 | 196 |
| 17 | **17** | 18 | 24 |
| 18 | **77** | 78 | 84 |
| 19 | **7** | 79 | 82 |
| 20 | **78** | **78** | 84 |
| 21 | **254** | 265 | 286 |

that $T^2$ will be more effective than $Q$.

Also, SPLS obtained the best results in MDR and Delay with $Q$, being the second using $T^2$. In this case, as this method is linear, did not take in account non-linear relations to construct groups thus they are not modelled by local DPCAs. This made that these non-linearities had more influence in $Q$ than in $T^2$.

Finally, Table VI shows that the proposals were able to detect more faults than the central DPCA, being NeuralNet the method with the highest figures, 21 faults detected with $T^2$ and $Q$. Also, SPLS was capable of detecting 20 faults out of 21 with both statistics, while central DPCA only gets to 18 in the best case. The fault alarm rate (FAR) for test data vary from 0 to 0.2004, a bit lower than DPCA with $T^2$ but clearly below than central DPCA with $Q$.

Table VI
FALSE ALARMS RATES AND FAULTS DETECTED (IN %)

| | SPLS DPCA | NNET DPCA | DPCA |
|---|---|---|---|
| FAR $T^2$ | 0 | 0.2004 | 0.6 |
| FAR $Q$ | 0.2004 | 0 | 28.1 |
| Detected faults $T^2$ | 20 | **21** | 17 |
| Detected faults $Q$ | 20 | **21** | 18 |

In these tests it is clear that the decentralized approach, using good block-construction methods, works better than the centralized method. This implies that, prior to do the fault detection, it is better to group the variables that share any relation between them.

## V. CONCLUSIONS

This paper proposed a fault detection method based on dividing the plant in blocks of variables, setting local fault detection models in each block and fusing all the local results to obtain a global fault detection outcome. The decentralization task was developed with two methods: Sparse PLS regression and Neural Net modelling. The first one is a linear method while the second is a non-linear one. In each group the fault detection method was based on Dynamic PCA which can model the time and space-correlations between variables. Finally, the global result was obtained using Bayesian Inference Criterion to fuse local results.

The objective was to check the advantages of the decentralized approach using linear and non-linear methods, and to test the performance of DPCA fault detection method in such a distributed plant. To do that, the proposals were compared with other method: a centralized DPCA [4], a dynamic, but not distributed method.

As can be seen in section IV, both proposals gave better results than centralized DPCA in Missed Detection Rate, detection delay and number of faults detected. Also, the results showed that SPLS works better using $Q$ statistic, while Neural Net is the best of the comparison with $T^2$.

In the case of SPLS, $Q$ statistic, which analyses information not captured by the model, worked better, suggesting that SPLS, as it is a linear method, was not able to capture non-linear relations to create the groups, thus local DPCA models were not as good as they should, giving worse results with $T^2$. On the other hand, Neural Net decentralization, as it is based on a non-linear modelling method, seemed to be able to capture non-linear relations between variables to create groups, allowing local modelling to obtain better models. This lead to better results with $T^2$, as it is a statistic that analyses the information retained by the model.

After this work, some questions remain opened. The first one is the selection of the decentralization method, known that there is not an unique and best option. So, it is necessary to do more tests with other techniques, linear and non-linear, having in mind that each type of plant will need one or other option. Also, it would be interesting to test the idea of combining two or more fault detection methods and fuse the results of them to obtain a final diagnostic. Another question is the selection of the fault detection method applied in each block. The DPCA method is effective but not perfect, so more techniques should be investigated and developed in order to approach to a Missed Detection Rate of 0%, 0 samples detection delay and 100% of faults detected.

## References

[1] T. Kourti and J.F. MacGregor. Multivariate SPC methods for process and product monitoring. *Journal of Quality Technology*, 28:409–428, 1996.

[2] R. Muradore and P. Fiorini. A PLS-based statistical approach for fault detection and isolation of robotic manipulators. *IEEE Transactions on Industrial Electronics*, 59(8):3167–3175, 2012.

[3] W. Ku, R.H. Storer, and C. Georgakis. Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and intelligent laboratory systems*, 30:179–196, 1995.

[4] E. L. Russell, L. H. Chiang, and R. D. Braatz. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2000.

[5] A. Simoglou, E.B. Martin, and A.J. Morris. Statisical performance monitoring of dynamic multivariate processes using state space modeling. *Computers & Chemical Engineering,*, 26(6):909–920, 2002.

[6] P.P. Odiowei and Y. Cao. State-space independent component analysis for nonlinear dynamic process monitoring. *Chemometrics and Intelligent Laboratory Systems*, 103:59–65, 2010.

[7] Z. Chen, K. Zhang, H. Hao, S. X. Ding, M. Krueger, and Z. He. A canonical variate analysis based process monitoring scheme and benchmark study. *IFAC Proceedings Volumes*, 47(3):10634–10639, 2014. 19th IFAC World Congress.

[8] M. Grbovic, W. Li, P. Xu, A.K. Usadi, L. Somg, and S. Vucetic. Decentralized fault detection and diagnosis via sparse PCA based decomposition and maximum entropy decision fusion. *Journal of Process Control*, 22:738–750, 2012.

[9] Y. Zhang, H. Zhou, S.J. Qin, and T. Chai. Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares. *IEEE Transactions on Industrial Informatics*, 6(1):3–10, 2010.

[10] A. Sanchez-Fernández, M. J. Fuente, and G. I. Sainz-Palmero. Fault detection in wastewater treatment plants using distributed pca methods. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–7, Sept 2015.

[11] A. Sánchez-Fernández, M. J. Fuente, and G. I. Sainz-Palmero. Fault detection with distributed PCA methods in water distribution networks. In *2015 23rd Mediterranean Conference on Control and Automation (MED)*, pages 156–161, June 2015.

[12] C. Tong, T. Lan, and X. Shi. Fault detection and diagnosis of dynamic processes using weighted dynamic decentralized PCA approach. *Chemometrics and Intelligent Laboratory Systems*, 161(Supplement C):34 – 42, 2017.

[13] Q. Cheng, P.K. Varshney, J. Michels, and C.M. Belcastro. Distributed fault detection via particle filtering and decision fusion. In *2005 7th International Conference on Information Fusion*, volume 2. IEEE, 2005.

[14] C. Tong and X. Shi. Decentralized monitoring of dynamic processes based on dynamic feature selection and informative fault pattern dissimilarity. *IEEE Transactions on Industrial Electronics*, 63(6):3804–3814, June 2016.

[15] W. Li, W. H. Gui, Y. F. Xie, and S. X. Ding. Decentralised fault detection of large-scale systems with limited network communications [brief paper]. *IET Control Theory Applications*, 4(9):1867–1876, September 2010.

[16] Z. Ge and Z. Song. Distributed PCA model for plant-wide process monitoring. *Industrial and Engineering Chemistry Research*, 52:1947–1957, 2013.

[17] R.R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

[18] P. Nomikos and J F. MacGregor. Multivariate SPC charts for monitoring batch processes. *Technometrics*, 37(1):41–59, 1995.

[19] W. E. Larimore. *Statistical methods in control and signal processing*. Marcel Dekker, 1997.

[20] H. Wold. *Estimation of principal components and related models by iterative least squares*, pages 391–420. Academic Press, New York, 1966.

[21] C. Colombani, P. Croiseau, S. Fritz, F. Guillaume, A. Legarra, V. Ducrocq, and C. Robert-GraniÃ©. A comparison of partial least squares (pls) and sparse pls regressions in genomic selection in french dairy cattle. *Journal of Dairy Science*, 95(4):2120–2131, 2012.

[22] S. Wold, M. Sjostrom, and L. Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

[23] H. Chun and S. Keleş. Sparse partial least squares regression for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(1):3–25, 2010.

[24] M. Kuhn and K. Johnson. *Applied predictive modeling*. 2013.

[25] J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17:245–255, 1993.

[26] J. Liu. Fault diagnosis using contribution plots without smearing effect on non-faulty variables. *Journal of Process Control*, 22:1609–1623, 2012.

[27] B. Jiang, D. Huang, X. Zhu, F. Yang, and R. D. Braatz. Canonical variate analysis-based contributions for fault identification. *Journal of Process Control*, 26:17–25, 2015.

[28] Sankar Mahadevan and Sirish L. Shah. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of Process Control*, 19:1627–1639, 2009.

[29] Funa Zhou, Ju H. Park, and Yajuan Liu. Differential feature based hierarchical PCA fault detection method for dynamic fault. *Neurocomputing*, 202:27–35, aug 2016.