Efficient residuals pre-processing for diagnosing multi-class faults in a doubly fed induction generator, under missing data scenarios

Roozbeh Razavi-Far [a,*], Enrico Zio [a,b], Vasile Palade [c]

[a] *Department of Energy, Politecnico di Milano, via Ponzio 34/3, 20133 Milan, Italy*
[b] *Systems Science and the Energetic Challenge, European Foundation for New Energy-Electricité de France, Ecole Centrale Paris and Supelec, Paris, 92295 Chatenay-Malabry Cedex, France*
[c] *Faculty of Engineering and Computing, Coventry University, Priory Street, Coventry, UK*

1. Introduction

The doubly fed induction generator (DFIG) is one of the most widely used classes of induction machines in the megawatt-class wind turbines (Hansen & Michalke, 2007). The DFIGs have shown a good performance in normal operation, but they are quite sensitive to particular classes of faults. The rate of failures in the sensors and the generator of wind turbines are reported to be approximately 14.1% and 5.5% of the total number of failures, that cause 5.4% and 8.9% of the system downtime (Ribrant & Bertling, 2007).

The sensor fault detection and isolation in the DFIGs has an important role to guarantee the safe and reliable operation of wind turbines. Since monitoring the generator entails processing the current and voltage sensor measurements, the first step is devoted to sensor fault diagnosis, which has been addressed in recent works (Boulkroune, Galvez-Carrillo, & Kinnaert, 2010; Galvez-Carrillo & Kinnaert, 2010).

Fault diagnosis can be performed in two major steps. Firstly, several signals, so-called residuals, reflecting faults in the process behavior, are generated. In the second step, the residuals are evaluated for decision making, to determine the time and the location of potential faults (Razavi-Far, Davilu, Palade, & Lucas, 2009a, 2009b).

Multiple observers schemes were developed in Boulkroune et al. (2010) and Galvez-Carrillo and Kinnaert (2010) to generate residuals associated to stator voltage and current sensors, as well as rotor sensors, respectively. These multiple observers were integrated in Razavi-Far and Kinnaert (2012, 2013) to reveal the mutual effects of the faults in each type of sensors on the residuals associated to another sensor type. This coupling prevented to develop a decision system by basic combination of the previously developed decision systems for each class of sensors. Thus, an effective classification technique has been used to design a suitable decision system in Razavi-Far and Kinnaert (2012, 2013).

The problem of fault classification can be tackled resorting to computational intelligence techniques. However, these approaches are usually based on time-series data of various signals in static environments (Razavi-Far, Davilu, Palade, & Lucas, 2009b). On the contrary, in dynamic environments, an incremental learning

* Corresponding author.
E-mail addresses: roozbeh.razavi@gmail.com, roozbeh.razavi@mail.polimi.it (R. Razavi-Far), enrico.zio@polimi.it, enrico.zio@ecp.fr, enrico.zio@supelec.fr (E. Zio), vasile.palade@coventry.ac.uk (V. Palade).

strategy is needed to update the decision system for fault classification. This has been done by resorting to ensemble of fault classifiers (Baraldi, Razavi-Far, & Zio, 2011b). In Baraldi, Razavi-Far, and Zio (2011a), a bagged ensemble of Fuzzy C-Means (FCM) classifiers was used for fault classification and its confidence for decision making has been studied in Baraldi, Razavi-Far, and Zio (2010).

The incrementally trained ensemble of classifiers in Baraldi et al. (2011b) can learn the new relations between the upcoming signals, while keeping the previously trained classifiers to preserve the existing knowledge. Albeit this has been successfully applied for decision making and fault classification in changing operating conditions (Baraldi et al., 2011b), the situation becomes more complicated when the datasets collected in subsequent installments have patterns of new classes of faults that were not included in previous datasets. Consequently, the base classifiers of the ensemble are doomed to misclassify patterns from faulty classes on which they were not trained.

The problem of new class fault diagnosis was firstly tackled by resorting to dynamic weighting ensembles (Razavi-Far, Baraldi, & Zio, 2012a), where a dynamic weighting ensembles algorithm was adopted for fault diagnosis in the feedwater system of a boil-ing water reactor (BWR). The algorithm is particularly developed for incremental learning of multiple new concept classes of faults. The detection of unseen classes in subsequent data was based on thresholding the normalized weighted average of the outputs (NWAO) of the base classifiers in the ensemble (Razavi-Far et al., 2012a, Razavi-Far, Baraldi, & Zio, 2012b).

Here a multiple observer scheme is used for residual generation, while for residual evaluation a dynamic weighting ensemble of classifiers is used. In the first step, a bank of observers generates a set of residuals that are robust to operating point changes. A so-called signal-based approach is used for residual generation of the stator current and voltage sensors, while two-stage filters exploiting the DFIG model and the balanced signal model are used for residual generation of the rotor currents, the same as in Razavi-Far and Kinnaert (2012, 2013).

In the second step, the pre-processed residuals are progressively fed into the dynamic weighting ensembles for fault classification. The algorithm incrementally learns the relation between projected residuals and faults, and dynamically classifies the faults including multiple new classes.

In Razavi-Far and Kinnaert (2012, 2013), prior to fault classification, the generated residuals ($r_i = r_1, r_2, \ldots, r_9$) were resampled (i.e., down-sampled) in the processing module and then forwarded to the fault classifier, i.e., the 'second step'. Each residual contains two vectors that form 18 features for the dynamic weighting ensemble of fault classifiers. The dynamic weighting ensemble of fault classifiers mapped these 18 features to 10 possible classes. These classes include the normal state '*ff* or fault-free' and 9 classes of faults (f_i=$f_1, f_2, \ldots, f_9$). The first three faults are sensor faults in the stator voltage at phase (a, b, c). Other faults correspond to sen-sor faults in stator and rotor currents at phase (a, b, c), respectively. In the preceding works (Razavi-Far & Kinnaert, 2012, 2013), it was shown that the decision module of the diagnostic system can isolate all classes with respect to the unavailability of patterns from all faulty classes during the training (i.e., new faults became avail-able dynamically in the course of time). The major focus was on detection and isolation of additive step-like faults, but additive drift-like faults were taken into account as well.

The generated residuals by multiple observers contain redundant or irrelevant residual vectors (i.e., features) that can degrade the fault classification performance. The pre-processing module in Razavi-Far and Kinnaert (2012, 2013) only resamples (i.e., down-samples) the residual vectors, which is a pattern-wise process.

To improve the fault classification performance, a pre-processing of the features is necessary. Feature selection is a task of pre-processing the data to select a subset of features. Feature extraction generates new features (e.g., latent residuals) from functions of the original features (i.e., generated residuals by multiple observers). This can improve the fault classification performance by improving the model interpretability, reducing overtraining, enhancing the generalization capability and shortening the training times.

The feature selection methods can be divided into three main categories: wrappers, filters and embedded methods (Guyon & Elisseeff, 2003). There exist different methods for feature extraction, such as those presented in Vong and Wong (2011), Vong, Wong, and Ip (2013) and Bruzzese (2014).

Moreover, the fault classifiers of the diagnostic system, like any other type of classifiers, fail to classify the incomplete patterns (i.e., containing some missing features). Thus, it is necessary to discard or impute the patterns with missing data/features before sending to the dynamic weighting ensemble. In the first case, the fault classification module cannot classify the fault for the missing patterns and the final decision is in question. In the latter case, the missing data are imputed in advance and, thus, the missing patterns can also be classified. There exists different number of missing data imputation techniques (Gheyas & Smith, 2010; Rassler, Rubin, & Zell, 2013). Although, these methods can impute the missing data, the outcome is the completed dataset that needs further pre-processing to reduce the size of features. This can be computationally expensive, and not feasible for online monitoring and diagnostic applications. Therefore, here a non-linear iterative partial least squares (NIPALS) algorithm along with the Wold cross-validation (Wold, 1978) has been used for pre-processing. This algorithm is fast and suitable for online application, it extracts the latent variables (i.e., latent residuals) from the residual data-sets, and it handles the missing data.

This paper aims to study the residuals and focus on the pre-processing module to provide more informative features of smaller size for fault classification. An efficient way to process the generated residuals is developed in order to extract latent information among residuals and provide informative features for the decision module of the diagnostic system. This is done by resorting to the principal component analysis (PCA), a popular data analysis technique.

The contribution of this work is in developing a non-linear iterative partial least squares (NIPALS) algorithm along with a dynamic weighting ensemble, for residual evaluation and new class fault diagnosis in dynamic environments. This algorithm is capable of reducing the number of the generated residuals, which incrementally become available, and projecting them onto the new feature space of smaller size, by extracting the latent information. The projected latent residuals allow faster incremental update of the ensemble of fault classifiers and improve the classification accuracy of some of the faults, while incomplete batches of residuals become available. The proposed classification scheme is validated on the problem of early diagnosis of new class faults in the sensors of a DFIG.

The rest of this paper is organized as follows. Section 2 describes briefly the system and presents the fault diagnostic scheme with a focus on the pre-processing module. Section 3 presents the NIPALS algorithm for the dimensional reduction of incomplete data. Next, the Wold cross-validation algorithm is used along with the NIPALS algorithm to estimate the number of principal components and extract the latent residuals. In Section 4, an application to the sensors of DFIG-based wind turbines is presented. First, the generated batch of residuals by multiple-observers are processed by the Wold cross-validation along with the NIPALS algorithm to form the latent

residuals. Then, the decision module of the diagnostic system incrementally learns the latent residuals and classifies multiple faults including new classes. Here, incomplete batches of residual data are used to validate the diagnostic system in the presence of missing data. Finally, conclusions are drawn in Section 5. The pseudo-codes of the NIPALS algorithms and the Wold cross-validation algorithm are presented in Appendix A.

2. System description and problem statement

The system description and the fault diagnosis scheme are first presented, as they are prerequisites for the problem statement.

2.1. The DFIG-based wind turbine

The considered system, shown in Fig. 1, is a DFIG-based wind turbine. The generator and the wind turbine rotor are coupled together via a gearbox. The stator of the DFIG machine is directly connected to the grid, while the rotor side is connected to a back-to-back converter via slip rings (Razavi-Far & Kinnaert, 2013). The back-to-back converter is composed of a rectifier con-nected to the rotor windings, that is called the rotor side converter (RSC), and an inverter connected to the power grid, namely, the grid side converter (GSC). A DC link has been devised between RSC and GSC to store energy and reduce the DC ripple. To reduce the harmonics injected by the GSC, a line filter has been placed between the GSC and the grid. The DFIG dynamics and notations are not described here for the sake of conciseness (the reader can refer to Razavi-Far & Kinnaert (2012, 2013) for a more detailed explanation).

2.2. Fault diagnosis scheme

The primary aim of this work is to detect and isolate single additive sensor faults in a controlled DFIG, as described in Razavi-Far and Kinnaert (2013). The faults are small additive faults

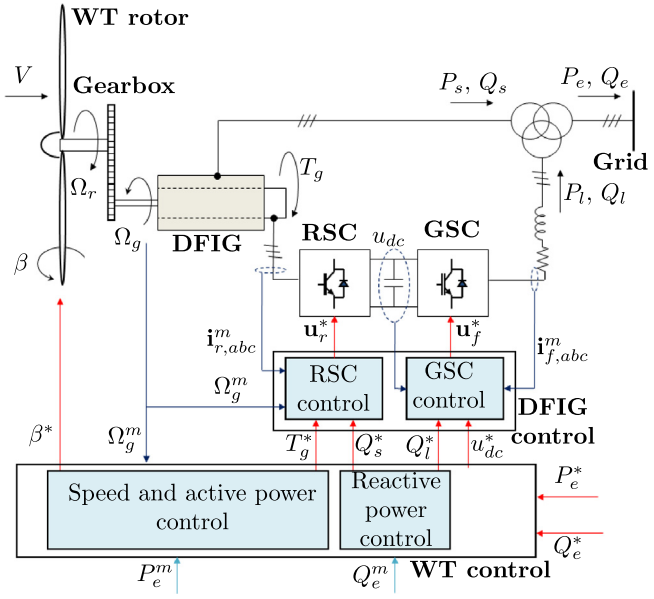

Fig. 1. Model of the DFIG-based wind turbine (Galvez-Carrillo & Kinnaert, 2010). Additional notations: V Wind speed, β pitch angle, $\Omega_{r,g}$ rotor and generator speed, P, Q active and reactive powers, lower indices e, s, r, l, f, dc respectively grid, stator, rotor, line filter, direct current, upper indices m, $*$ stand for measurements and references (Razavi-Far & Kinnaert, 2013).

in the stator voltage and current sensors as well as rotor current sensors, specifically step and drift-like faults.

Fault diagnosis is performed in two major steps. The proposed scheme is displayed in Fig. 2. This scheme has two main components: the residual generation and fault classification modules. Firstly, residual signals reflecting faults in the system are generated from sampled sensor measurements and command inputs. These residuals have zero mean in the fault-free mode, and some of them are subject to a change in the mean upon occurrence of a sensor fault. Then, the residuals are evaluated by the decision module, in order to determine the time and the location of faults.

2.2.1. Residual generation

The residual generator module contains multiple observers and complementary filters in three integrated sub-modules to detect all possible classes of faults in rotor and stator sensors. Signal-based observers are used for residual generation of stator current and voltage sensors, while two-stage filters utilizing the DFIG model and the balanced signal model are used for residual generation of rotor currents. The residual generation module (i.e., both approaches) is completely described in Razavi-Far and Kinnaert (2012, 2013).

These multiple observers generate a set of residuals $r(r_1, \ldots, r_n)$ that are robust against modeling uncertainties and change in oper-ating points (Razavi-Far & Kinnaert, 2012, 2013). Each residual is designed to be sensitive to a subset of faults. Besides, the residuals are designed to have different responses to different faults.

2.2.2. Fault classification

The fault classification module matches each pattern of the residual vectors with one of the pre-assigned classes, i.e., the fault-free or faulty classes.

Unlike conventional pattern recognition methods for fault clas-sification based on time-series data of various signals in static environments, it is assumed that the residuals successively become available in different batches and the signature trends of the residuals vary in different time intervals. The new signature trends can be related to new classes of faults that have not been seen previously during the training of the fault classifier. Therefore, the fault classifier needs to be incrementally updated over a period of time. This incremental learning can be performed by resorting to an ensemble of fault classifiers (Baraldi et al., 2011b).

Ensemble learning is a two-step algorithm: first, multiple diverse classifiers are trained; second, the outcomes of the individual classifiers are combined strategically to achieve higher classification performance from its individual-base classifiers. Albeit an ensemble of fault classifiers outperforms a single fault classifier (Baraldi et al., 2011a), and is more robust and confident (Baraldi et al., 2010), and can learn and diagnose in an incremental fashion (Baraldi et al., 2011b), it is doomed to misclassify patterns of new classes of faults (i.e., classes unseen during the training session) in subsequent datasets.

In Razavi-Far and Kinnaert (2013), a dynamic weighting ensemble algorithm, called *Learn*$^{++}$*.NC* (Muhlbaier, Topalis, & Polikar, 2009), is used to dynamically learn and diagnose the new classes of faults. This algorithm, which is used here for fault classification, creates and trains a new member of the ensemble $\{\mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3, etc\}$ (i.e., a pre-assigned number of MultiLayer Perceptron MLP-based classifiers) with each new batch of data $\{S^1, S^2, S^3, etc\}$. Each MLP is a three layer network in which the number of neurons in the input layer is equal to the number of features used for the diagnosis (i.e., the number of principal components or latent residuals that can vary for each dataset), and the number of neurons in the output layer is equal to the number of classes, here 10.

Each MLP network is trained on a different subset of the available training data $\{S^1, S^2, S^3, etc\}$. A training data subset is drawn

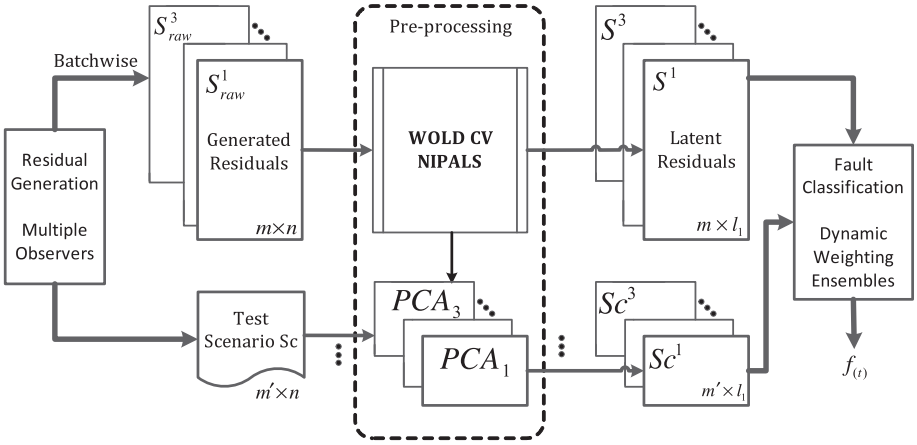

Fig. 2. Block diagram of the diagnostic system: the pre-processing bridges the gap between the residual generation and fault classification modules.

according to an iteratively updated distribution in order to train each individual base classifier.

The algorithm evaluates the ensemble on the current dataset and calculates the errors. The algorithm calls a subroutine, named the dynamically weighted consult and vote (*DW-CAV*), to create a weighted average of classifier errors on current and recent datasets, and assigns voting weights to each classifier based on age-adjusted weighted errors. This smart voting mechanism allows base classifiers to consult with each other (i.e., cross-reference their decisions with the class labels used during their training sessions) and dynamically adjust their voting weight for each pattern (Muhlbaier et al., 2009). The final decision is, then, obtained as the weighted majority voting of all classifiers. The detailed explanation of the algorithm and its pseudo-codes are formally presented in Razavi-Far and Kinnaert (2013).

2.3. Problem description

The pre-processing module, which is placed between the residual generation and decision modules, prepares the feature space for the fault classifiers by resampling and combining the residual components (Razavi-Far & Kinnaert, 2012, 2013). It resamples down-sampled residuals (r_i=r_1, r_2, ... , r_9) and, then, combined residual vectors (i.e., each residual contains two vectors) to form a feature space of size 18.

The pre-processed residuals were progressively fed to the dynamic weighting ensemble of fault classifiers to match each pattern from the feature space of size 18 with a pre-assigned class out of 10 (i.e., the fault-free *ff* and 9 classes of faults $f_1, f_2, \ldots, f_9$). The decision module in Razavi-Far and Kinnaert (2012, 2013), then, was able to dynamically learn and isolate all classes including unseen classes of faults in the subsequent residual datasets.

This work focuses on the pre-processing module to provide informative features of smaller size for fault classification. The pre-processing module is mainly the Wold cross-validation algo-rithm along with the NIPALS algorithm. Fig. 2 presents the fault diagnosis scheme with focus on the pre-processing module.

The batches of residuals are progressively fed to the pre-processing module. Each residual dataset is projected to the new feature space and the latent information among the residuals is extracted. The Wold cross-validation also estimates the optimal number of principal components latent residuals to form the new feature space. This reduces the dimension of the feature space by extracting latent information and canceling the noise among residual data, which allows faster incremental update of the ensemble of fault classifiers and improves the classification accuracy of some faults. This module can also process the incomplete batches of residuals (e.g., missing data in the generated residuals due to sensor failures).

Additionally, the WOLD cross-validation NIPALS algorithm creates and preserves a PCA model for each batch of residuals which projects the raw residuals $\{S^1_{raw}, S^2_{raw}, S^3_{raw}\}$ onto the latent residuals $\{S^1, S^2, S^3\}$.

During the test, these models $\{PCA_1, PCA_2, PCA_3\}$ will be used as pre-processing modules of the corresponding ensemble $(\mathcal{E}^1, \mathcal{E}^2, \mathcal{E}^3)$. Thus, any subsequent test pattern is firstly predicted by means of different preserved PCA models that project the test pattern onto different feature spaces (i.e., PCA models extract different numbers of principal components from an unprocessed test scenario *Sc*, to form Sc^1, Sc^2, Sc^3), and, then, feed them to the corresponding member of the ensemble.

3. Principal component analysis

Principal component analysis (PCA) is a popular method of data analysis. PCA is a simple, non-parametric method to extract relevant information from datasets. It can be used therefore to reduce the data dimension and reveal the latent knowledge and simplified underlying dynamics.

To perform the principal component analysis, the data is centered and scaled. The data is mean-centered, i.e., $x_{ij} = x_{ij} - \bar{x}_i$. By centering, the coordinate system is shifted to a new reference point which is the origin of the coordinate system in a *n*-dimensional space. Next, the data is simply scaled through dividing each column by its standard deviation. Thus variance and standard deviation of each column turn out to be 1, providing an equal opportunity of contribution to the model for each variable.

PCA is an orthogonal linear transformation of the data to a new coordinate system such that the maximum variance by any projection of the data lies on the first coordinate (the first principal component), the second greatest variance on the second coordinate, and so on.

The data matrix $X(m \times n)$ can be explained by the summation of a structure and a residual as follows:

$$X = TP' + E \tag{1}$$

The structure is the matrix product of $TP\prime$, where T and P denote the scores and the loadings, respectively, and E stands for residuals. The scores explain how the different rows of X (patterns) are

related to each other. The loadings are the weights of the variables in X on the scores T. The matrix residuals, $E(m \times n)$, is the part of X which is not explained by the PCA model TP'.

The principal components of X are, then, the eigenvectors of $X'X$ or the rows of P, where X is a centered $m \times n$ data matrix. The size of E is explained in terms of squared variance.

There exist different PCA algorithms, such as the singular value decomposition (SVD) and non-linear iterative partial least-squares (NIPALS) (Wold, 1966; Wold, Esbensen, & Geladi, 1987). The NIPALS algorithm is a sequential technique to compute the principal components.

Here the NIPALS algorithm is used to find principal components of the raw residual matrix and to select proper features for the dynamic weighting ensemble of fault classifiers. The reason for considering the NIPALS algorithm here is threefold: it handles missing data, it works well for huge data and it calculates the components sequentially.

The pre-processing of the residuals can benefit from the above stated features of the NIPALS algorithm, since the recorded data from residual vectors form a huge dataset due to high sampling frequency in wind turbine applications, and may contain missing values due to sensor failures.

The algorithm extracts each principal component successively, starting with the first component having the maximum variance, and then the second component, and so on.

3.1. The NIPALS algorithm

The NIPALS algorithm (Geladi & Kowalski, 1986) aims to find the principal components. It can be limited to finding the first l principal components of $X'X$, starting with the largest eigenvalue PC_1 and further. l must be less than or equal to the number of principal components n. The pseudo-code of the standard NIPALS is presented in Algorithm 1 (see Appendix A).

Upon convergence at each iteration of the NIPALS algorithm, the score t_k and loading p_k vectors are stored as the kth column in matrix T and P, respectively. Then, the final crucial step of the NIPALS algorithm deflates the residual data matrix; it takes away the captured variability by the component PC_k from X.

The last step, so called deflation, guarantees mutual orthogonality of the extracted components since each subsequent component can merely see the remained variations after eliminating all the preceding components. Thus, there exists no variability of the same type that could be explained by two principal components. Upon deflation, this procedure is repeated from step 1 to capture the next component. The algorithm can decide whether to keep that component or not.

The algorithm can be terminated once a certain number l of components are captured; this can be defined by rule of thumb. However, it is important to estimate a proper number of components since the captured components will be used as extracted features for the fault classification.

Undoubtedly, having a large number of principal components increases the number of extracted features but also the complexity. On the contrary, capturing few principal components decreases the number of extracted features and may lead to lower classification performance of the fault classifier.

3.2. How many principal components?

Indeed, it is crucial to know how many principal components should be retained to capture most of the data variability. Different methods have been proposed to find the actual dimensionality of the data (Bartlett, 1950; Cattel, 1996; Jackson, 1991; Jolliffe, 1986; Kaiser, 1960; Malinowski, 1977). However, these methods depend on subjective choices or non-realistic assumptions. Regardless of conventional criteria and methods, cross-validation is an objective method that does not need any assumption.

A cross-validatory PCA estimation is not based on the eigenvalues of the covariance matrix but on the predictive ability of the different principal components.

Cross-validation is an efficient and popular approach that has been successfully used in literature to determine the number of underlying features and to estimate the average prediction error. The basic principle of cross-validation is to split the data and leave out a portion, construct a model, and then predict the left-out patterns by means of the constructed model.

Cross-validation is a standard resampling technique. A cross-validation procedure has been used along with the SVD to estimate the number of principal components (Eastment & Krzanowski, 1982; Krzanowski, 1983, 1987) on which at each step the evaluation set is formed only by one item of the data matrix, so called the leave-one-out. In this method, the maximum amount of information is used for estimation, that is computationally expensive.

In Wold (1976, 1978), an efficient technique, so-called Wold cross-validation, has been proposed along with the NIPALS algorithm to identify the dimensions that best explain the systematic variations in data. In Diana and Tommasi (2002), different variation of cross-validation, single cross-validation (SCV) (Wold, 1976), double cross-validation (DCV) (Wold, 1978) and full cross-validation (FCV) (Forina, Lanteri, Boggia, & Bertran, 1993) are studied and compared. This comparative study shows that cross-validation is an efficient technique in determining the number of principal components, albeit the number of principal components captured by each method is slightly different (Diana & Tommasi, 2002). In Bro, Kjeldahl, Smilde, and Kiers (2008), the efficiency and performance of the Wold cross-validation in finding the num-ber of principal components is studied and compared with other cross-validation techniques.

Here, the Wold cross-validation is used to determine a proper number of components for the PCA model that best explain the variation in the data and if possible not the noise. The Wold cross-validation aims to determine the number of principal com-ponents that extract all systematic variance from X, leaving unstructured residual variance in E, in a way that fitting any addi-tional components will not improve variability of the data and, rather, start to fit this noise and unstructured variance in E.

Since the Wold cross-validation relies on the special property of the NIPALS algorithm to cope with missing data (Rubin, 1976; Nelson, Taylor, & MacGregor, 1996), the NIPALS algorithm is reformulated to handle the missing values.

3.3. The NIPALS algorithm with missing values

To find the principal components, the NIPALS algorithm minimizes the following objective function:

$$F = \sum_{ij} \left(x_{ij} - \sum_{k=1}^{n} t_{ik} p_{jk} \right)^2 \tag{2}$$

under the orthonormality (i.e., $\sum_j p_{jk} p_{jl} = \delta_{kl}$, the so called Kronecker delta) and orthogonality (i.e., $\sum_i t_{ik} t_{il} = 0 \ \forall \ k \neq l$) conditions (Grung & Manne, 1998).

The missing values of x_{ij} along with their model representations $\sum_k t_{ik} p_{jk}$ should be deflated from the objective function (2). Assume that Y is a full matrix and X is the known portion of Y. In the matrix X, the element with the same index of missing positions in Y are substituted with zeros. With this notation, the symbol of X can still be utilized for the matrix values applied as input to the computation.

These two matrices (i.e., X and Y) are related together by means of an incidence matrix Ψ. The matrix Ψ is, with the same dimension of X and Y, defined as follows:

$$\psi_{ij} = \begin{cases} 1 & \text{if } y_{ij} \text{ is known} \\ 0 & \text{if } y_{ij} \text{ is missing (i.e., unknown)} \end{cases} \quad (3)$$

These matrices are linked together as $x_{ij}=\psi_{ij}y_{ij}$ and $\psi_{ij}^2=\psi_{ij}$. Thus, the objective function in Eq. (2) can be rewritten as follows:

$$F = \sum_{ij} \psi_{ij} \left(y_{ij} - \sum_{k=1}^{n} t_{ik} p_{jk} \right)^2 = \sum_{ij} \left(x_{ij} - \sum_{k=1}^{n} t_{ik} \psi_{ij} p_{jk} \right)^2 \quad (4)$$

Thus, to find principal components, the modified objective function should be minimized (Grung & Manne, 1998). This can be done by setting the partial derivatives ($\partial F/\partial t_i$ and $\partial F/\partial p_j$) equal to zero. Consequently, the achieved equations for the score and the loading along with the normalization condition can be solved iteratively. Algorithm 2 presents the pseudo-code of the modified NIPALS with missing values (see Appendix A). In other words, the standard NIPALS algorithm without missing values is achieved with all $\psi_{ij}=1$.

3.4. Cross-validatory estimation of the number of components

The Wold cross-validation algorithm (Wold, 1978) along with the NIPALS steps is presented in Fig. A.1 (see Appendix A). This algorithm splits the data matrix X into Γ cancellation groups. In each iteration, one group is deleted from X to form the train and test subsets. Consequently, a model is fitted to the remaining data by means of the NIPALS algorithm and, then, the fit of the model to the corresponding left-out elements is evaluated.

Each cancellation matrix is achieved through deleting a sequence of individual elements x_{ij} in a diagonal scheme. For instance, in the γth group, $\gamma = 1, 2, \ldots, \Gamma$, the missing sequence is achieved through deleting the element numbers $\{\gamma, \gamma+\Gamma, \gamma+2\Gamma, etc\}$ in a row-wise scheme. This guarantees that all elements are left out once, upon completing Γ groups. These deleted individual elements are marked as 'missing values'. Thanks to the incidence matrix, as explained in the previous section, the NIPALS algorithm handles the generated missing values. The number of groups Γ is an arbitrary choice between 4 and 7 with the condition that Γ is not a divisor of m or n.

The Wold cross-validation finds one component at each recurrence. Next, the algorithm evaluates the first component and subtracts the valid component from the data. Consequently, the generated residuals are only used to evaluate the next principal component.

Table 1
Faulty situations.

ff	Normal	Normal or Fault-free situation
f_1	f_{u-sa}	Fault in stator voltage phase *a*
f_2	f_{u-sb}	Fault in stator voltage phase *b*
f_3	f_{u-sc}	Fault in stator voltage phase *c*
f_4	f_{i-sa}	Fault in stator current phase *a*
f_5	f_{i-sb}	Fault in stator current phase *b*
f_6	f_{i-sc}	Fault in stator current phase *c*
f_7	f_{i-ra}	Fault in rotor current phase *a*
f_8	f_{i-rb}	Fault in rotor current phase *b*
f_9	f_{i-rc}	Fault in rotor current phase *c*

4. Sensor fault diagnosis of a DFIG-based wind turbine

In this section, the capability of the proposed scheme is tested with respect to the early diagnosis of sensor faults in a DFIG-based wind turbine. These faults regard malfunctions in the sensors of the DFIG. The list of faults is reported in Table 1.

4.1. Design of the diagnostic system

The simulation environment and conditions for training is explained in Razavi-Far and Kinnaert (2012, 2013) in order to have a fair comparison.

Here, three steps of simulation consisting of different classes of faults have been performed. The generated residuals by the bank of observers are collected in different sets of residual data and, then, fed to the Wold cross validatin algorithm as input. The batch of residuals consists of three residual datasets. Each one includes patterns of simulated scenarios (i.e., normal and some faults). The first step of the simulations includes patterns of the normal status and three faults (i.e., f_1, f_2, f_3) in S_{raw}^1.

The second (third) simulation creates S_{raw}^2 (S_{raw}^3), made of different patterns of normal status and six (nine) classes of faults, i.e., $f_1, \ldots, f_6$ (f_9), introducing another three (six) new classes.

These residual datasets S_{raw}^1, S_{raw}^2, S_{raw}^3, become available progressively in the course of time. Each one consists of 9 residuals (r_i=r_1, r_2, …, r_9) and each residual is a two-dimensional vector which yields a feature space of 18 variables. Each residual dataset is firstly preceded by the pre-processing module (i.e., the Wold cross-validation and NIPALS) to project the residual to the new fea-ture space and extract the principal components to form the latent residual datasets in S^1, S^2, S^3. Each latent residual dataset is used to train a member of the ensemble $\mathcal{E}^1$, $\mathcal{E}^2$, $\mathcal{E}^3$.

The Wold cross-validation along with the NIPALS algorithm is used to reduce the dimension of the residual datasets $S_{raw}^1, S_{raw}^2, S_{raw}^3$ to smaller sets of features, i.e., the 'latent residuals'. Each set contains most of the information in the large residual set.

The number of principal components 'selected features' is less than or equal to the number of original variables 'residual vectors'. The NIPALS projects the patterns of the raw set of residuals in a way that the first principal component has the largest possible variance (i.e., variability in the raw set), and each subsequent principal component in turn has the highest variance possible under the orthogonality constraint, i.e., uncorrelated with the preceding principal components.

4.2. Latent residuals characteristics

The latent residuals are principal components extracted by the Wold cross-validation algorithm. The residuals successively become available at different snapshots (i.e., $S_{raw}^1, S_{raw}^2, S_{raw}^3$). Consequently, the number of extracted components (i.e., features) can vary for each incoming dataset.

This may lead to pre-processed datasets (i.e., S^1, S^2, S^3) with different number of features. Therefore the individual classifiers of each ensemble ($\mathcal{E}^1$, $\mathcal{E}^2$, $\mathcal{E}^3$) are trained with different datasets (i.e., S^1, S^2, S^3), which contain different numbers of features.

Thus, it is necessary to preserve the PCA model constructed for each dataset (i.e., S^1, S^2, S^3) as a pre-processing module of the corresponding ensemble ($\mathcal{E}^1$, $\mathcal{E}^2$, $\mathcal{E}^3$). Consequently, any upcoming test pattern (i.e., a test scenario Sc of size $m\prime \times n$) is firstly pre-dicted by means of different PCA models (e.g., PCA_1, PCA_2, PCA_3) that project the test pattern to different feature spaces (i.e., Sc^1, Sc^2, Sc^3) and feed to the corresponding ensemble of fault classifiers.

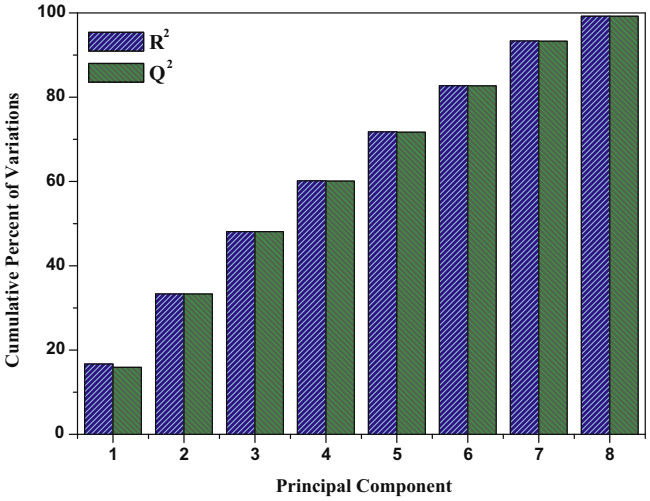

Fig. 3. The explained cumulative variance upon extraction of each principal component.

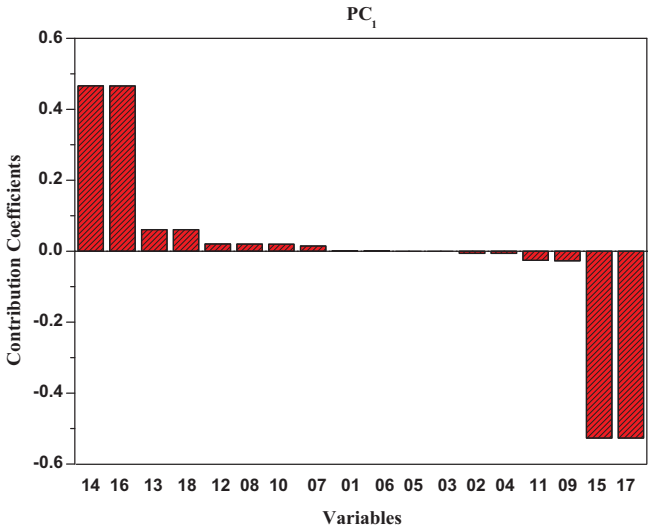

Fig. 4. Contribution plot for the first principal component t_1.

Each member of the ensemble, i.e., $\mathcal{E}^1$ ($\mathcal{E}^2$) ($\mathcal{E}^3$), which is trained by the dataset S^1 (S^2) (S^3) of size $m' \times l_1$ (l_2) (l_3) then evaluates the corresponding projection of the test scenario Sc^1 (Sc^2) (Sc^3) of size $m' \times l_1$ (l_2) (l_3) (Fig. 2).

The Wold cross-validation projects the residual datasets (i.e., S^1_{raw}, S^2_{raw} and S^3_{raw}, each one containing 18 features) to the new feature spaces of size 7, 7 and 8; stored in S^1, S^2 and S^3, respectively.

For instance, the Wold cross-validation extracts only 8 principal components from the third dataset S^3_{raw}. These 8 principal components explain 99.2% of the variability in S^3_{raw}. Fig. 3 displays the cumulative R^2 and Q^2 (see Eqs. (5) and (6)) for the S^3_{raw} dataset, after each principal component.

R^2_{cum} is the percent of the variation of all the data explained by the PCA model, i.e., the goodness of fit. Q^2_{cum} is the percent of the variation of all the data that can be predicted by the PCA model, i.e., the goodness of prediction.

R^2 and Q^2 for each component can be computed as follows:

$$R^2 = 1 - \frac{Var(\widetilde{E}^k)}{Var(X)} \tag{5}$$

$$Q^2 = 1 - \frac{Var(\text{predicted}\widetilde{E}^k)}{Var(X)} \tag{6}$$

where $Var(\text{predicted}\widetilde{E}^k)$ stands for the prediction error sum of squares $PRESS(k)$.

Consequently the cumulative R^2 and Q^2 of the $l-th$ component are calculated as follows:

$$R^2_{cum} = 1 - \prod_{k=1}^{l} \frac{Var(\widetilde{E}^k)}{Var(X)} \tag{7}$$

$$Q^2_{cum} = 1 - \prod_{k=1}^{l} \frac{PRESS(k)}{Var(X)} \tag{8}$$

R^2 is always larger than Q^2. If $R^2 = Q^2$, it means that the principal component is predictive in the PCA model, whereas small value of Q^2 indicates that principal component is likely fitting the noise.

The first extracted principal component explains 16.67% of the variability in S^3_{raw} (see Fig. 3). It is important to see how this component is affected by different residual vectors of S^3_{raw}. This issue can be studied by means of contribution plots.

A contribution plot shows the impact of each variable on each score and also identifies which variables are pushing the statistics out of their control limits (Kourti & MacGregor, 1996).

The contributions in Fig. 4 show the explained variation for the first principal component t_1, i.e., how different variables 'residual vectors' contribute to the first principal component 'projected residual'. Fig. 4 shows that variables 15, 17, 14 and 16 have the highest contribution in the first principal component. These values correspond to the rotor current residual vectors at phase a, b, b and c, respectively. These residuals vectors have the maximum variation and sensitivity to the faults at the rotor current sensor, which are introduced in the third dataset S^3_{raw}.

The projected latent residuals are also discussed in terms of component scores, i.e., the projected variable values corresponding to a particular pattern, and loadings, i.e., the weight by which each standardized original variable should be multiplied to get the component score (Shaw, 2003).

Fig. 5 shows the scores t_1 versus t_2, which are the most informative among projected residuals. This Figure shows how patterns of

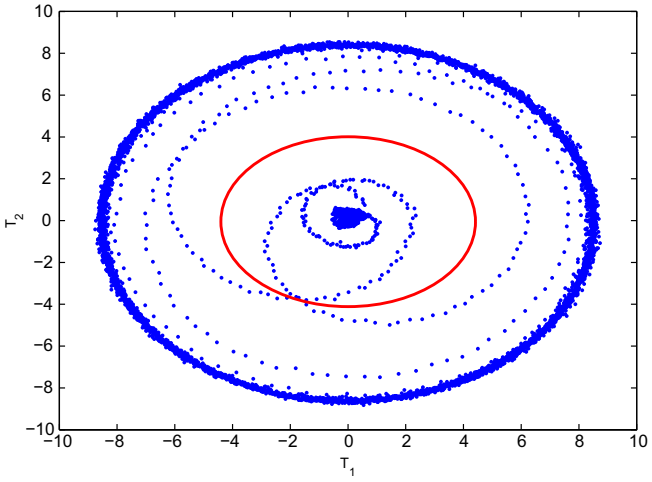

Fig. 5. Scores t_1 versus t_2; the solid ellipse represents a 95% confidence interval of the data.

the first two projected residuals summarize the variability of the original residual data. The score t_1 (first component) explains the largest variation of the residual space, followed by t_2, etc. Hence, the scatter plot of t_1 versus t_2 displays how the patterns are situated with respect to each other.

The score plot, Fig. 5, presents different groups, similarities, outliers and other patterns in the residual data. Patterns near each other are similar.

In PCA, an outlier is a pattern that lies outside the confidence limits of the PCA model. Outliers can be detected on the scores plot, i.e., the patterns outside of the red circle in Fig. 5. These outliers correspond to the faulty patterns.

The loadings explain the structure of the residual data in terms of variable correlations. Each variable has a loading on each principal component. It reveals how much the variable contributes to that principal component, and how well that principal component takes into account the variation of that variable over the residual data.

The loadings can be explained as the correlation of the variables 'residual vectors' with the scores T (i.e., $t_1, t_2, \ldots, t_n$). Fig. 6 shows the circle of correlations and the plot of the loadings of the variables with the first two principal components.

Variables (15, 17), (14, 16) and (13, 18) are mutually correlated (see their positions on the solid circle of correlation). These variables are of paramount importance for the first two principal components, since other variables are located around the origin and have low value with respect to p_1 and p_2.

Variables (15, 17) and (14, 16) have the highest values on p_1 (Fig. 6) and and correspondingly the most contribution on the first principal component (Fig. 4). Variables (15, 17) are negative on the first loading vector p_1 (Fig. 6) and and have negative contribution on the first principal component (Fig. 4). On the contrary, variables (14, 16) are positive on the first loading vector p_1 (Fig. 6) and, correspondingly, have positive contribution on the first principal component (Fig. 4).

Variables (13, 18) are highly correlated and placed on the solid circle (Fig. 6), and highly contribute to the first two princi-pal components. Their contributions on the first two principal components are positive, but their contribution on the second principal component is higher than the first principal component, since their values in the second loading vector p_2 is higher than in p_1.

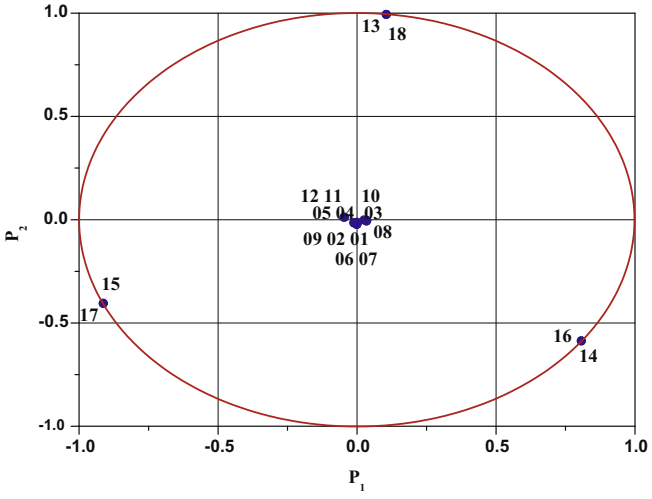

Fig. 6. Principle components loadings, p_1 versus p_2.

Table 2
Number of patterns in each dataset, for each class.

Dataset	Number of patterns										
	Total	*ff*	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
Train-set^1	232	58	58	58	58	–	–	–	–	–	–
Train-set^2	500	59	59	59	59	88	88	88	–	–	–
Train-set^3	1028	59	59	59	59	88	88	88	176	176	176
Test-set^1	48	12	12	12	12	–	–	–	–	–	–
Test-set^2	109	13	13	13	13	19	19	19	–	–	–
Test-set^3	223	13	13	13	13	19	19	19	38	38	38

4.3. Dynamic learning of the latent residuals

The latent residuals S^1 (S^2) (S^3) with 7 (7) (8) features are pattern-wise down-sampled and, then, fed to the dynamic weighting ensembles (*DWE*).

In the first step, while S^1 becomes available, the *DWE* splits the data into *Train-set*^{1} and *Test-set*^{1}. A summary of the datasets characteristics is reported in Table 2.

Train-set^1 and *Test-set*^{1} consist of different patterns of the normal status and three faults (f_1,f_2,f_3).

The *DWE* algorithm, first creates an ensemble $\mathcal{E}^1$ of ten classifiers which are trained on *Train-set*^{1} and then evaluated with *Test-set*^{1}.

Similarly, in the second (third) step, once S^2 (S^3) becomes available, the *DWE* splits the latent residuals into *Train-set*^{2} (*Train-set*^{3}) and *Test-set*^{2} (*Test-set*^{3}).

Train-set^2 (*Train-set*^{3}) and *Test-set*^{2} (*Test-set*^{3}) are made of different patterns of normal status and six (nine) classes of faults, introducing new classes. Upon data split, the *DWE* creates an ensemble $\mathcal{E}^2$ ($\mathcal{E}^3$) of ten new classifiers, which are trained on *Train-set*^{2} (*Train-set*^{3}) and, then, tested with *Test-set*^{2} (*Test-set*^{3}).

The MLP networks are weakly trained (i.e., error goal of 5%) to guarantee the diversity among the base classifiers, which leads to achieve a higher performance by the ensemble. Each MLP classifier in the first and second ensembles $\mathcal{E}^1$ and $\mathcal{E}^2$ has 70 neurons in the hidden layer. The *MLP* classifiers of the third ensemble $\mathcal{E}^3$ contains 80 neurons in the hidden layer.

Table 3 shows that the performance of the diagnostic system trained with latent residuals is improved compared with the performance of the diagnostic system, which was trained with residuals. The classification performance of the first step is reported in the first row of Table 3. The *DWE* at first step, i.e., $\mathcal{E}^1$ classifies the *Test-set*^{1} with high accuracy; however, the clas-sification performance is decreased with respect to the classifica-tion of the *Test-set*^{2} and *Test-set*^{3}. This is due to the presence of patterns of unseen classes during the training with *Train-set*^{1} in the test data.

The second row of Table 3 corresponds to the second step, while the second dataset S^2 becomes available.

The *DWE* at this step adds ten new classifiers to the ensemble $\mathcal{E}^2$. $\mathcal{E}^2$ classifies the *Test-set*^{1} and *Test-set*^{2} with high accuracy, but

Table 3
Performances of the ensembles to the test datasets by means of latent residuals (with raw residuals).

	Test-set^1	*Test-set*^{2}	*Test-set*^{3}
$\mathcal{E}^1$, *Train-set*^{1}	**98.3**% (97.9%)	47.3% (46.7%)	25.1% (22.4%)
$\mathcal{E}^2$, *Train-set*$^{1\&2}$	98.3% (97.9%)	**98.6**% (98.1%)	51.6% (48.8%)
$\mathcal{E}^3$, *Train-set*$^{1\&2\&3}$	100% (100%)	100% (100%)	**100**% (99.5%)

still the performance on the *Test-set*^{3} is low, which is due to unseen patterns of faults (f_7, f_8, f_9) in the *Test-set*^{3}.

The *DWE* evaluates the *Test-set*^{2} in a way that for the patterns of new classes (f_4, f_5, f_6), the *DW-CAV* subroutine decreases the voting weight of the classifiers of the first ensemble $\mathcal{E}^1$, which are trained with *Train-set*^{1} and did not learn any patterns of (f_4, f_5, f_6) during their training, and at the same time increases the voting weight of the new classifiers which are trained with *Train-set*^{2}. Adjusting the voting weights increases the classifica-tion performance.

The last row of the Table 3 corresponds to the third step, on which S^3 becomes available. The classification performances of the ensemble $\mathcal{E}^3$ with respect to all test datasets are high. This is due to the fact that in the last training session, $\mathcal{E}^3$ is trained on *Train-set*^{3}, which is made of patterns of all classes.

The bold entries in Table 3 are used for decision making. The entries inside the parentheses in the Table 3 are classification performances achieved by the diagnostic system without pre-processing, i.e., the pre-processing module dose not includes the Wold cross-validation and NIPALS (Razavi-Far & Kinnaert, 2012, 2013).

The reported results in Table 3 show that the classification performances are improved by means of the proposed pre-processing module (i.e., latent residuals). This improvement can be seen in almost all scenarios in the Table 3, particularly, when the faults in the rotor current sensor are introduced (the last row of the Table). This improvement is more significant for the *Test-set*^{3}, which contains patterns of f_7, f_8, f_9. This is due to high correlation of their corresponding residuals in the raw residual sets.

The final row of the Table shows a performance of 100% for all test-sets that cannot be achieved without pre-processing by all means. The reason for the slight improvement in the other cases is twofold:

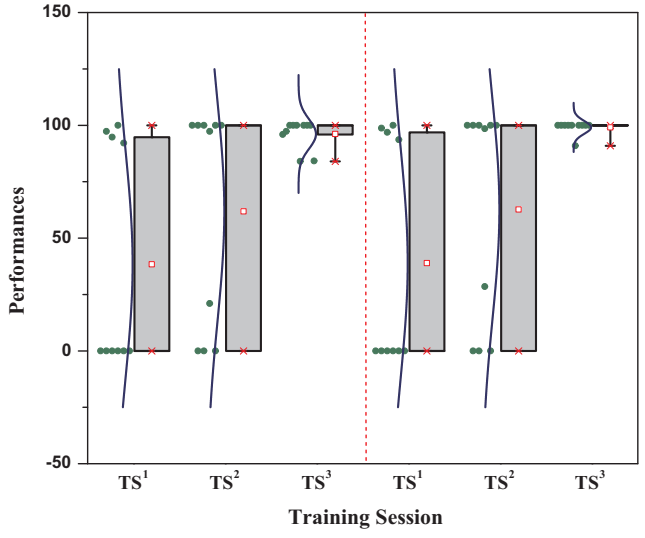

Fig. 7. Box plots representing the distribution of class-specific performances (%) with respect to the patterns of the *Test-set*$^{1+2+3}$ in different training sessions. The vertical line 'red dash' splits the training sessions, left boxes correspond to the method without latent residuals and right boxes correspond to the proposed method with latent residuals. Solid circles represent the distribution of class-specific performances in each training session. The red dashes stand for maximum and minimum performance values and the solid squares denote the average performance in each training session. The red crosses stand for 1 and 99 percentiles of the performance values. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

(a) It is mostly due to the fact that the patterns of transients corresponding to changes in operating conditions are assigned to the fault-free class, whereas their signature trends are very similar to those of the faulty scenarios. Since the data are randomly drawn into train and test sets, some of the classifiers are not trained with the patterns of transients or have seen only few of them during their training, whereas there exists a large number of transient patterns in the corresponding test-set and thus they are doomed to misclassify these patterns (all the entries below the main diagonal in the Table including the main diagonal entries).

(b) There is not any improvement on the patterns of unseen class, i.e., patterns of a new class of fault on which they have not been trained (all the entries above the main diagonal in the Table).

Fig. 7 presents class-specific performances, with respect to the combined *Test-set*$^{1+2+3}$ at the end of each training session $TS^{\mathcal{K}}$, where the current residual data $S^{\mathcal{K}}$ is merely used to train $\mathcal{T}_{\mathcal{K}}$ fault classifiers (without access to previously learned datasets). $\mathcal{K}$ is the number of residual datasets '3' introduced to the *DWE* and $\mathcal{T}_{\mathcal{K}}$ stands for the number of *MLP* classifiers '10' added to the ensemble for each residual dataset $S^{\mathcal{K}}$.

The first three boxes in the left side belong to the training sessions without pre-processing (i.e., raw residuals) and the last three boxes in the right side belong to the training session with pre-processing (i.e., with latent residuals).

In both methods, by moving from TS^1 to TS^2 more classes of faults are correctly classified since TS^2 and TS^3 have seen more classes during the training.

During the first training session TS^1, the first ten *MLP* classifiers are merely trained on *Train-set*^{1} and, thus, they are doomed to misclassify the patterns of class f_4 to f_9 (see the six zeros corresponding to the first box, TS^1). The patterns of the first three faults (f_1, f_2, f_3) are correctly classified with high performances: 94.7%, 92.1% and 97.3%, respectively.

The classification performance with respect to the patterns of fault-free *ff* is 100%, since the *ff* patterns in the *Train-set*^{1} include normal status in addition to patterns of transient due to changes in operating conditions.

Here, some patterns of transients are included in *Train-set*^{1} and used during the first training session TS^1, thus the classification performance of the first ten classifiers with respect to the *ff* fraction of the combined *Test-set*$^{1+2+3}$ is 100%.

The second box corresponds to the second training session TS^2 and shows that the number of correctly classified faults are increased (i.e., the classification performances with respect to f_1 to f_6 are high), however the second ten classifiers misclassify the patterns of f_7 to f_9.

Besides, the patterns of transients are not seen during the second training session, thus the performance of TS^2 with respect to the *ff* fraction of the combined *Test-set*$^{1+2+3}$ is decreased to 21%.

The third box TS^3 shows that the classification performances with respect to all classes of faults are high since the last ten classifiers are trained on *Train-set*^{3}, which includes the patterns of all faulty classes. The classification performances with respect to almost all classes are high, $\succ$ 95%, however the classification performances with respect to *ff* and f_8 are 84.2%.

The former is due to the fact that only few transient patterns are used for training in the last session and, consequently, the performance of TS^3 with respect to the *ff* class of the combined *Test-set*$^{1+2+3}$ is slightly better.

The same interpretation is valid for the next three boxes in the Fig. 7, that correspond to the training sessions $TS^{\mathcal{K}}$ in the proposed method, which include pre-processing (latent residuals).

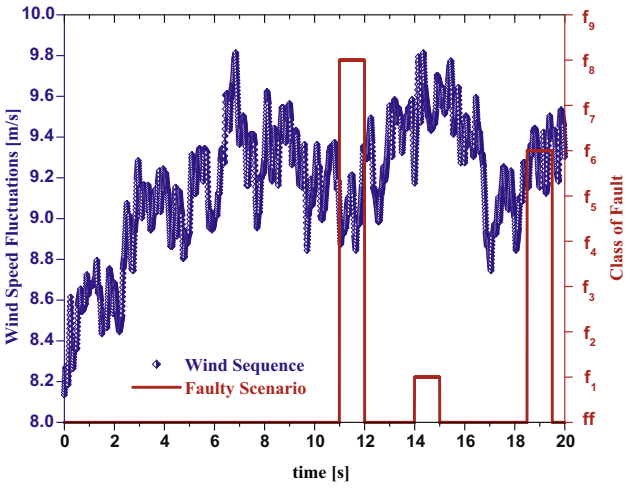

Fig. 8. The wind speed sequence (left vertical axis) and the simulated faulty scenario (right vertical axis) (Razavi-Far & Kinnaert, 2013).

The pair wise comparison between the first and second training sessions of each method, boxes (1, 4) and (2, 5), shows the same distribution of classification performances. Moreover, it shows that the performances achieved by the proposed method is slightly increased during the TS^1 and TS^2.

This improvement is significant for the last training session TS^3, boxes (3, 6), where the performances are improved about 3% on average. This is due to the high correlation between the residual vectors corresponding to the faults in the rotor current sensors (f_7, f_8, f_9).

4.4. Validation of the diagnostic system

The robustness of the diagnostic scheme with respect to wind speed fluctuations is studied in Razavi-Far and Kinnaert (2013). A realistic wind speed sequence has been used to generate the voltage, current and generator speed signals that enter the fault diagnosis system. This wind sequence is characterized by an average wind speed equal to 9.17 m/s and a turbulence intensity equal to 6.7%.

The simulation lasts for 20 s; the first 11 s correspond to normal mode *ff* and then a step-like fault appears at $t = 11$ s and disappears at $t = 12$ s in the measurement of i_{r-b} (corresponding to f_8). The fault has a magnitude of 5% of the rated rotor current (peak value). The second step-like fault is injected for a time interval between $t = 14$ s and $t = 15$ s in the measurement of u_{s-a} (corresponding to f_1). This fault also has a magnitude of 5% of the rated stator voltage (peak value). Lastly, a step-like fault appears at time $t = 18.5$ s and disappears at $t = 19.5$ s in the measurement of i_{s-c} (corresponding to f_6). The last fault has a magnitude of 5% of the rated stator current (peak value). This scenario is presented in Fig. 8 along with the wind speed sequence.

The test scenario *Sc* containing the raw residuals are firstly predicted by the preserved PCA models in the pre-processing module to generate different sets of latent residuals (Sc^1, Sc^2, Sc^3). Each set of latent residuals Sc^1 (Sc^2) (Sc^3) is fed to the corresponding member of the ensemble $\mathcal{E}^1$ ($\mathcal{E}^2$) ($\mathcal{E}^3$). The aggregated outcome of the base classifiers in the ensembles correctly classifies most of the patterns associated to the different classes during the considered scenario except some chattering in the output and misclassification due to unexpected fluctuations in residuals that have not been seen during the training sessions.

The missed alarm rate is equal to zero and false alarm rate is 3.43%. The percentage of the missed and false alarms shows a satisfactory generalization performance of the dynamic weighting ensemble of fault classifiers with respect to the test patterns of the wind sequence. The obtained results by means of the proposed pre-processing module show a significant reduction of false alarm rate, of 3.78% compared to the preceding work (Razavi-Far & Kinnaert, 2013) without pre-processing (i.e., the false alarm rate was 7.21%).

4.4.1. Incomplete scenario

The decision module relies on the trained ensembles using residual data, which is extremely dependent on data collection, storage and analysis. The collected residual data often have missing data due to disconnections, transient failures in the sensors, etc.

Sensors are subject to different risks and occasional failures due to wear and tear, severe environmental conditions (e.g., covered in water or snow) or exposure to physical damage; causing a sudden temporary failure until being replaced. Additionally, usually sensors rely on batteries, and are inaccessible during operation runtimes. Under such circumstances, it is possible to have miss reading from some sensors in different time intervals. This generally implies significant reduction in the diagnostic performance, since the base classifiers of the ensembles are doomed to classify the missing data, i.e., patterns with one or more miss-ing features. Concurrently, the diagnostic system needs to operate continuously and make decisions online without drastic reduction in the classification performance due to temporary missing sensor readings.

To analyze the performance of the diagnostic system in the presence of the missing data, an incomplete scenario has been

Table 4
Missing data pattern in the incomplete scenario.

	r_1		r_2		r_3		r_4		r_5		r_6		r_7		r_8		r_9	
	$r_{1,1}$	$r_{1,2}$	$r_{2,1}$	$r_{2,2}$	$r_{3,1}$	$r_{3,2}$	$r_{4,1}$	$r_{4,2}$	$r_{5,1}$	$r_{5,2}$	$r_{6,1}$	$r_{6,2}$	$r_{7,1}$	$r_{7,2}$	$r_{8,1}$	$r_{8,2}$	$r_{9,1}$	$r_{9,2}$
9250																		
250													■	■			■	■
250			■	■	■	■												
250							■	■	■	■								
10000	0	0	250	250	250	250	250	250	250	250	0	0	250	250	0	0	250	250

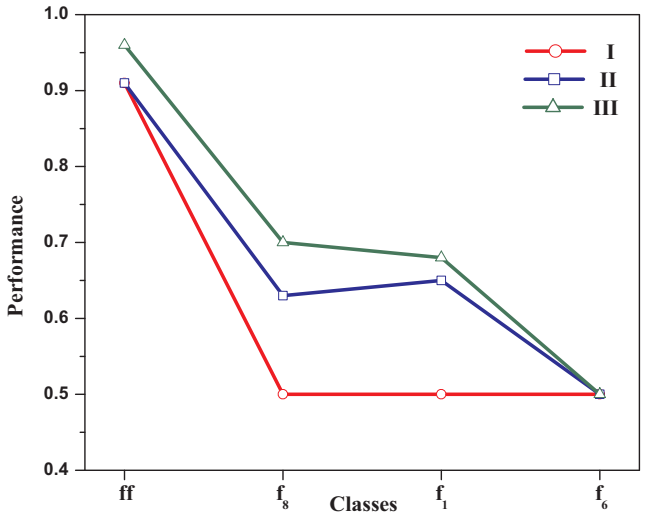

Fig. 9. Class-specific performance comparison on the faulty scenario *Sc*.

simulated. To form the incomplete scenario, the previous faulty scenario (see Fig. 8) has been simulated for 20*s*, but it is assumed that the sensor reading that measures the i_{r-b} (u_{s-a}) (i_{s-c}) is missing in the second half of the corresponding faulty interval, i.e., from $t = 11.5$ s (14.5 s) (19 s) to $t = 12$ s (15 s) (19.5 s).

Any failure in a sensor reading leads to missing data in the four corresponding residual vectors due to mutual relations between the sensor measurements and residual vectors (Razavi-Far & Kinnaert, 2013). For instance, failure in sensor reading of the u_{s-a} leads to missing data in r_2 and r_3 (i.e., corresponding residual vectors $r_{2,1}, r_{2,2}, r_{3,1}, r_{3,2}$).

Thus, the imposed failures in sensor reading induce a missingness in the residual data, and form 250 monotone missing patterns and 500 non-monotone missing patterns in the simulated scenario.

Table 4 illustrates the missing data pattern in the incomplete scenario. The grey squares represents the missing values. The incomplete scenario contains 10,000 patterns (the left most column in the last row), and Table 4 presents the missing data patterns into the minimum number of rows possible. Thus, the 10,000 rows of the data patterns are collapsed into 4 main rows (rows 3 to 6 in the Table). The last row of the Table presents the number of missing data of each residual vector in the scenario. For instance, the residual vector of $r_{2,1}$ has been missed in 250 patterns. The number of the complete patterns is 9250 (the third row of the Table). The 4th (5th) (6th) row shows a pattern where the residual vectors of $r_{7,1}, r_{7,2}, r_{9,1}, r_{9,2}$ ($r_{2,1}, r_{2,2}, r_{3,1}, r_{3,2}$) ($r_{4,1}, r_{4,2}, r_{5,1}, r_{5,2}$) are missing together and this type of missing data patterns occurs 250 (250) (250) times in the incomplete scenario, as can be seen from the left most column.

The incomplete set of residuals *Sc* cannot be fed to the dynamic weighing ensemble, since the base classifiers of the ensembles are not able to handle the missing data in *Sc*. Thus, the preceding diagnostic scheme in Razavi-Far and Kinnaert (2013) fails to classify the incomplete scenario *Sc*.

One alternative is to discard the incoming patterns that contain missing values (I) which leads to a very rapid deterioration of the performance, since there will be no classification for continuous periods of time.

Another remedy is to impute, the missing data by the mean value of the missing feature (II) which is a popular and fast technique for online applications. Data imputation techniques have been extensively used in different applications and offer an imperative avenue for future research in fault diagnosis under missing data assumption. However, the iterative multiple imputation techniques are computationally expensive and further research is needed to prove their efficiency for online monitoring applications.

Here, the pre-processing module projects the incomplete scenario (i.e., the raw residuals *Sc* with missing data) to complete sets of latent residuals (Sc^1, Sc^2, Sc^3), due to the inherent capability of the NIPALS algorithm in handling the missing data (see Section 3.3). Then, the complete sets of latent residuals Sc^1, Sc^2, Sc^3 are fed to their corresponding member of the ensemble $\mathcal{E}^1$, $\mathcal{E}^2$, $\mathcal{E}^3$, respectively, for fault classification purposes (III). Fig. 9 shows the class-specific performances, with respect to the faulty scenario, achieved by each technique (I, II, III).

The obtained results show a good classification performance on the fault-free class, since all the patterns of the *ff* class are complete (i.e., no missing feature in the *ff* class patterns). However, the proposed method (III) outperforms other two techniques, with more than 5% of improvement.

Method (I) has the lowest performance (50%) for the faulty classes since the missing data are totally discarded. The proposed method (III) significantly improves the classification performance for the faulty classes in the stator voltage and the rotor current and outperforms the mean imputation method (II) by about 3% and 7%, respectively. However, the classification performance with respect to the faults in the stator current is not improved.

The classification performance achieved by the methods I, II, III, with respect to all patterns of the test scenario (including missing patterns), are 84%, 86%, 91%, respectively. Thus, the proposed method not only can improve the classification accuracy when all the patterns are available, but also when readings from some sensors are missing for continuous periods of time.

The computational time needed by the pre-processing methods (i.e., down-sampling, discarding, mean imputation, and the Wold cross-validation NIPALS algorithm) is small and can be ignored.

The main difference is in the training of the diagnostic classifier and incremental update of the ensemble. All of the pre-processing methods except the Wold cross-validation NIPALS algorithm provide a full set of residual vectors for the dynamic weighting ensemble of fault classifiers, while the Wold cross-validation extract the latent residuals (i.e., less number of features) and, thus, the computational time to train the $\mathcal{T}_K$ MLP base classifiers of the dynamic weighting ensemble is reduced about 7 min.

5. Conclusions

This paper has proposed an efficient pre-processing method to generate the latent residuals for the diagnosis of new classes of faults in the controlled DFIG sensors. The proposed scheme has been validated on a three-phase simulation and the generated residuals by means of multiple observers are successively collected in different batches by performing each step of the simulation. These residuals correspond to the stator voltage, current and rotor current sensors. Each set of raw residual data contains patterns of new classes of faults, i.e., unseen classes in the previous datasets.

The raw residual datasets are firstly processed by the Wold cross-validation NIPALS algorithm, which estimates the optimal

number of principal components, in order to extract the latent information among the residuals and reduce the feature space. Consequently, the fault classification module, which is a Dynamic Weighting Ensemble algorithm, incrementally learns the latent residuals-faults relations, dynamically adjusts the voting weights of the base classifiers and diagnoses the faults including multiple new classes.

The novelty of the work stands in improving the fault classification performance and decision making under missing data assumption. The Wold cross-validation NIPALS algorithm is fast and suitable for online application, it extracts the latent residuals from the raw residual datasets, and it handles the missing data. The proposed pre-processing module projects the raw residuals onto the new feature space of a smaller size, i.e., latent residuals, and it preserves the constructed PCA models to project any upcoming test patterns onto the pattern with the same number of features for the corresponding member of the ensemble.

The attained results on three-phase simulation show that the pre-processing module improves the fault classification performance by extracting informative latent residuals. This improvement is substantial for the faults in the rotor current sensors, where there exists a high correlation between their corresponding residual vectors.

Moreover, the computational time is reduced, which is a vital factor in online monitoring systems. This fast online training and incremental update is due to the use of training sets of smaller sizes (i.e., set of latent residuals which includes fewer number of features).

More importantly, the Wold cross-validation NIPALS algorithm can handle the missing data and it allows the diagnosis of missing patterns in the incomplete scenarios and, thus, the decision module significantly outperforms diagnostic schemes which discards or substitutes the missing patterns with the mean values.

Since missing data are inevitable, there is a need for further study and implementation of accurate and fast data imputation techniques for online monitoring and fault diagnosis applications. Moreover, iterative multiple imputation techniques create several values for each missing data. The uncertainties and confidence intervals of the imputed variables can control the diversity in the ensemble learning and improve the classification performance, which looks to be a valuable direction for future research.

Acknowledgment

The authors would like to thank cordially to Professor Michel Kinnaert, Department of Control Engineering and System Analysis at Université Libre de Bruxelles, for valuable discussions in the early stage of this work and anonymous reviewers for their valuable comments.

Appendix A. The NIPALS and Wold cross-validation algorithms

For the completeness of the paper, the pseudo-codes of the NIPALS algorithm without missing values, the NIPALS algorithm with missing values, and the Wold cross-validation algorithm are presented in Algorithms 1, 2, and Fig. A.1, respectively.

Algorithm 1. The NIPALS algorithm without missing values.

INPUTS:
X is a mean centered data matrix of size $m \times n$
$X^0 = X$, the initial mean centered matrix
η is a threshold ($\approx 10^{-6}$) for the convergence check
n is the maximum number of PCs
DEFINITIONS:
t_k and p_k are the scores and loadings of PC_k
l is the number of PCs to be extracted $l \leq n$
λ_k and λ_{k-1} are the eigenvalues
it is the number of iteration
REQUIRE:
$\lambda_0 \leftarrow 3 \quad \lambda_1 \leftarrow 2 \quad it \leftarrow 0$
for $k = 1$ *to* l **do**
  INITIALIZE t_k, randomly assign to any column of X^{k-1}
  while $|\lambda_k - \lambda_{k-1}| > \eta\lambda_{k-1} \ \wedge \ it < 1000$ **do**
    UPDATE the convergence parameters

$$it \leftarrow it + 1$$
$$\lambda_{k-1} \leftarrow \lambda_k$$

    PROJECT X^{k-1} to t_k to find the related loading p_k

$$p'_k = (t'_k X^{k-1})/(t'_k t_k)$$

    NORMALIZE the loading vector p_k to length 1

$$p_k = p_k * (p'_k p_k)^{-0.5}$$

    PROJECT X^{k-1} to p_k to find the related score t_k

$$t_k = (X^{k-1} p_k)/(p'_k p_k)$$

    CALCULATE the eigenvalue as: $\lambda_k = t'_k t_k$
  end
  REMOVE the estimated principal component from X^{k-1}

$$X^k = X^{k-1} - t_k p'_k$$

end

Algorithm 2. The NIPALS algorithm with missing values.

INPUTS:
$X^0 = X$, related to Y via Ψ-matrix: $x_{ij} = \psi_{ij} y_{ij}$
η is a threshold ($\approx 10^{-6}$) for the convergence check
l is the number of PCs to be extracted $l \leq n$
DEFINITIONS:
t_k and p_k are the scores and loadings of PC_k
λ_k and λ_{k-1} are the eigenvalues
it is the number of iteration
REQUIRE:
$\lambda_0 \leftarrow 3 \quad \lambda_1 \leftarrow 2 \quad it \leftarrow 0$
for $k = 1$ *to* l **do**
  INITIALIZE t_k, randomly assign to any column of X^{k-1}
  while $|\lambda_k - \lambda_{k-1}| > \eta\lambda_{k-1} \ \wedge \ it < 1000$ **do**
    UPDATE the convergence parameters

$$it \leftarrow it + 1$$
$$\lambda_{k-1} \leftarrow \lambda_k$$

    PROJECT X^{k-1} to t_k to find the related loading p_k

$$p_{jk} = \frac{\sum_i x_{ij}^{k-1} t_{ik}}{\sum_i \psi_{ij}^{k-1} t_{ik}^2}$$

    NORMALIZE the loading vector p_k to length 1

$$p_{jk} = \frac{p_{jk}}{\sqrt{\sum_j p_{jk}^2}}$$

    PROJECT X^{k-1} to p_k to find the related score t_k

$$t_{ik} = \frac{\sum_j x_{ij}^{k-1} p_{jk}}{\sum_j \psi_{ij}^{k-1} p_{jk}^2}$$

    CALCULATE the eigenvalue as follows: $\lambda_k = t'_k t_k$
  end
  REMOVE the estimated principal component from X^{k-1}

$$X^k = X^{k-1} - t_k p'_k$$

end

INPUTS: X, k=0, Γ=7

1. CALCULATE the initial residual sum of squares that is the sum of squared deviations from the origin:

$$RSS(0) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{ij} - \bar{x}\right)^2, \qquad \text{where} \quad \bar{x} = \sum_{i=1}^{m}\sum_{j=1}^{n}\frac{x_{ij}}{mn}$$

2. SPLIT X randomly into Γ groups of the same size (row-wise)
3. DO FOR $\gamma = 1, 2, ..., \Gamma$
   3.1. DELETE the $\gamma - th$ group of rows E_γ, to form a reduced $\left[\tilde{m}_\gamma \times n\right]$ matrix $\tilde{X}_\gamma$ and compute $\tilde{\alpha}_j = \sum_{i=1:x_{ij}\in\tilde{X}_\gamma}^{\tilde{m}_\gamma} x_{ij}/\tilde{m}_\gamma$
   3.2. CALCULATE the prediction errors as the differences between $\tilde{\alpha}_j$ and the left-out elements of E_γ, to form the predictive residual sum of squares: $PRESS_\gamma(0) = \sum_{i,j:x_{ij}\in E_\gamma}\left(x_{ij} - \tilde{\alpha}_j\right)^2$

   END DO
4. COMPUTE the total $PRESS(0)$ by adding the partial prediction errors: $PRESS(0) = \sum_{\gamma=1}^{\Gamma} PRESS_\gamma(0)$
5. CALCULATE the ratio $R(0) = PRESS(0)/RSS(0)$. The predictions are improved if $R(0) < 1$
6. FORM the initial residual matrix of size $m \times n$:

$$\tilde{E}^0 = \begin{cases} \left[\tilde{e}_{ij}^0\right] & \text{if } R(0) \le 1 \quad \text{where} \quad \tilde{e}_{ij}^0 = x_{ij} - \hat{\alpha}_j = x_{ij} - \sum_{i=1}^{m}\frac{x_{ij}}{m} \\ X & \text{if } R(0) > 1 \end{cases}$$

7. $k = k + 1$
8. COMPUTE the residual sum of squares, $RSS(k) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\tilde{e}_{ij}^{k-1}\right)^2$
9. SPLIT the residual matrix $\tilde{E}^{k-1} = \left[\tilde{e}_{ij}^{k-1}\right]$ into Γ groups through a cancellation matrix in a diagonal scheme
10. DO FOR the left-out group $\gamma = 1, 2, ..., \Gamma$
    10.1. DELETE the $\gamma - th$ sequence of the elements $\{\gamma,\ \gamma + \Gamma,\ \gamma + 2\Gamma,\ etc\}$ of the matrix $\tilde{E}^{k-1}$, to form a cancellation matrix E_γ that holds only the left-out elements. The $\tilde{E}_\gamma^{k-1}$ includes all elements except the left-out ones
    10.2. ASSIGN proper values 'zeros' to the left-out elements of $\tilde{E}_\gamma^{k-1}$ and form the incidence matrix for NIPALS
    10.3. ESTIMATE the next principal component $\tilde{t}_{ik}^{\gamma}$ and $\tilde{p}_{jk}^{\gamma}$ by fitting a model to $\tilde{E}_\gamma^{k-1}$ with the NIPALS algorithm
    10.4. PREDICT the model of the cancellation matrix using the component: $\tilde{t}_{ik}^{\gamma}\tilde{p}_{jk}^{\gamma}$
    10.5. CALCULATE the prediction errors as the difference between the elements of the cancellation matrix E_γ and the predicted model $\tilde{t}_{ik}^{\gamma}\tilde{p}_{jk}^{\gamma}$. The partial predictive residual sum of squares can be calculated as follows:

$$PRESS_\gamma(k) = \sum\sum_{i,j:\tilde{e}_{ij}^{k-1}\in E_\gamma}\left(\tilde{e}_{ij}^{k-1} - \tilde{t}_{ik}^{\gamma}\tilde{p}_{jk}^{\gamma}\right)^2$$

    END DO
11. COMPUTE the total $PRESS(k) = \sum_{\gamma=1}^{\Gamma} PRESS_\gamma(k)$ and the ratio $R(k) = PRESS(k)/RSS(k)$
12. CHECK whether or not the inclusion of the $(k - th)$ component in the model improves the prediction errors:

$$\begin{cases} \text{if } R(k) \le 1 & \text{proceed to step 13} \\ \text{if } R(k) > 1 & \text{return the appropriate number of components } k - 1 \text{ and stop} \end{cases}$$

13. CALL the NIPALS algorithm to fit a PCA model to the complete matrix $\tilde{E}^{k-1}$ with one component $\hat{t}_{ik}^{\gamma}$ and $\hat{p}_{jk}^{\gamma}$
14. DEFLATE the component from the model to determine the residual variation $\tilde{E}^k = \left[\tilde{e}_{ij}^{k-1} - \hat{t}_{ik}^{\gamma}\hat{p}_{jk}^{\gamma}\right]_{m\times n}$
15. Go back to step (7)

Fig. A.1. The pseudo-code for the Wold cross-validation algorithm (Wold, 1978; Diana & Tommasi, 2002).

References

Baraldi, P., Razavi-Far, R., & Zio, E. (2010). A method for estimating the confidence in the identification of nuclear transients by a bagged ensemble of FCM classifiers. In *7th international topical meeting on nuclear plant instrumentation, control, and human-machine interface technologies (NPIC&HMIT)* (pp. 283–293). Las Vegas, Nevada: American Nuclear Society.

Baraldi, P., Razavi-Far, R., & Zio, E. (2011a). Bagged ensemble of FCM classifier for nuclear transient identification. *Annals of Nuclear Energy, 38*(5), 1161–1171. Baraldi, P., Razavi-Far, R., & Zio, E. (2011b). Classifier-ensemble incremental-learning procedure for nuclear transient identification at different operational conditions. *Reliabilty Engineering and System Safety, 96*(4), 480–488.

Bartlett, M. (1950). Tests of significance in factor analysis. *British Journal of Statistical Psychology, 3*, 77–85.

Boulkroune, B., Galvez-Carrillo, M., & Kinnaert, M. (2010). Robust sensor fault detection and isolation for a doubly-fed induction generator. In *50th IEEE conference on decision and control* (pp. 1600–1606).

Bro, R., Kjeldahl, K., Smilde, A., & Kiers, H. (2008). Cross-validation of component models: A critical look at current methods. *Analytical and Bioanalytical Chemistry, 390*(5), 1241–1251.

Bruzzese, C. (2014). Diagnosis of eccentric rotor in synchronous machines by analysis of split-phase currents; part II: Experimental analysis. *IEEE Transactions on Industrial Electronics, 61*, 4206–4216.

Cattel, R. (1996). The scree test for the number of factors. *Multivariate Behavioral Research, 1*, 245–276.

Diana, G., & Tommasi, C. (2002). Cross-validation methods in principal component analysis: A comparison. *Statistical Methods and Applications, 11*, 71–82.

Eastment, H., & Krzanowski, W. (1982). Cross-validatory choice of the number of component analysis. *Technometrics, 24*, 73–77.

Forina, M., Lanteri, S., Boggia, R., & Bertran, E. (1993). Double cross full validation. *Quimica Analitica, 12*, 128–135.

Galvez-Carrillo, M., & Kinnaert, M. (2010). Sensor fault detection and isolation in three-phase systems using a signal-based approach. *IET Control Theory and Applications, 4*(9), 1838–1848.

Geladi, P., & Kowalski, B. R. (1986). Partial least-squares regression: A tutorial. *Analytica Chimica Acta, 185*, 1–17.

Gheyas, I. A., & Smith, L. S. (2010). A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing, 73*, 3039–3065.

Grung, B., & Manne, R. (1998). Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems, 42*(1-2), 125–139.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.

Hansen, A., & Michalke, G. (2007). Fault ride-through capability of DFIG wind bines. *Renewable Energy, 32*(9), 1594–1610.

Jackson, J. (1991). *A user's guide to principal components*. New York: Wiley. Jolliffe,

I. (1986). *Principal component analysis*. Berlin Heidelberg New York: Springer.

Kaiser, H. (1960). The application of electronic computers to factor analysis. *Educational and Psychological Measurement, 20*, 141–151.

Kourti, T., & MacGregor, J. F. (1996). Multivariate SPC methods for process and product monitoring. *Journal of Quality Technology, 28*, 409–428.

Krzanowski, W. (1983). Cross-validatory choice in principal component analysis; some sampling results. *Journal of Statistical Computation and Simulation, 18*, 299–314.

Krzanowski, W. (1987). Cross-validation in principal component analysis. *Biometrics, 43*, 575–584.

Malinowski, E. (1977). Theory of error in factor analysis. *Analytical Chemistry, 49*, 606–612.

Muhlbaier, M., Topalis, A., & Polikar, R. (2009). Learn++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks, 20*(1), 152–168.

Nelson, P., Taylor, P., & MacGregor, J. (1996). Missing data methods in PCA and PLS: Score calculations with incomplete observations. *Chemometrics and Intelligent Laboratory Systems, 35*, 45–65.

Rassler, S., Rubin, D. B., & Zell, E. R. (2013). Imputation. *Wiley Interdisciplinary Reviews: Computational Statistics, 5*, 20–29.

Razavi-Far, R., & Kinnaert, M. (2012). Incremental design of a decision system for residual evaluation: A wind turbine application. In *8th IFAC conference fault detection, supervision and safety of technical processes* (Vol. 8(1), pp. 343–348).

Razavi-Far, R., Baraldi, P., & Zio, E. (2012a). Dynamic weighting ensembles for incremental learning and diagnosing new concept class faults in nuclear power systems. *IEEE Transactions on Nuclear Science, 59*(5), 2520–2530.

Razavi-Far, R., Davilu, H., Palade, V., & Lucas, C. (2009a). Model based fault detection and isolation of a steam generator using neuro-fuzzy networks. *Neurocomputing Journal, 72*, 2939–2951.

Razavi-Far, R., Davilu, H., Palade, V., & Lucas, C. (2009b). Neuro-fuzzy based fault diagnosis of a steam generator. In *7th IFAC conference on fault detection, supervision and safety of technical processes* (pp. 1180–1185).

Razavi-Far, R., Baraldi, P., & Zio, E. (2012b). Ensemble of neural networks for detection and classification of faults in nuclear power systems. In *World scientific proceedings series on computer engineering and information science 7; uncertainty modeling in knowledge engineering and decision making – proceedings of the 10th international FLINS conference* (Vol. 7, pp. 1202–1207). Istanbul, Turkey.

Razavi-Far, R., & Kinnaert, M. (2013). A multiple observers and dynamic weighting ensembles scheme for diagnosing new class faults in wind turbines. *Control Engineering Practice, 21*(9), 1165–1177.

Ribrant, J., & Bertling, L. (2007). Survey of failures in wind power systems with focus on swedish wind power plants during 1997–2005. In *IEEE power engineering society general meeting*, Tampa, USA.

Rubin, D. (1976). Inference and missing data. *Biometrika, 63*, 581–592.

Shaw, P. (2003). *Multivariate statistics for the environmental sciences*. 0-340-80763-6. Hodder-Arnold.

Vong, C., & Wong, P. (2011). Engine ignition signal diagnosis with wavelet packet transform and multi-class least squares support vector machines. *Expert Systems with Applications, 38*, 8563–8570.

Vong, C.-M., Wong, P.-K., & Ip, W.-F. (2013). A new framework of simultaneous-fault diagnosis using pairwise probabilistic multi-label classification for time-dependent patterns. *IEEE Transactions on Industrial Electronics, 60*, 3372–3385.

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*. NY: Academic Press.

Wold, S. (1976). Pattern recognition by means of disjoint principal components models. *Pattern Recognition, 8*(3), 127–139.

Wold, S. (1978). Cross-validatory estimation of the number of components in factor and principal component models. *Technometrics, 20*, 397–405.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems, 2*, 37–52.