Nonlinear process monitoring and fault isolation using extended maximum variance unfolding

Yuan-Jui Liu, Tao Chen, and Yuan Yao

Department of Chemical Engineering, National Tsing Hua University, Hsinchu, 31003, Taiwan, R.O.C

Department of Chemical and Process Engineering, University of Surrey, Guildford GU2 7XH, UK

Abstract

Kernel principal component analysis (KPCA) has become a popular technique for process monitoring, owing to its capability of handling nonlinearity. Nevertheless, KPCA suffers from two major disadvantages. First, the underlying manifold structure of data is not considered in process modeling. Second, the selection of kernel parameters is problematic. To avoid such deficiencies, a manifold learning technique named maximum variance unfolding (MVU) is considered as an alternative. However, such method is merely able to deal with the training data, but has no means to handle new samples. Therefore, MVU cannot be applied to process monitoring directly. In this paper, an extended MVU (EMVU) method is proposed, extending the utilization of MVU to new samples by approximating the nonlinear mapping between the input space and the output space with a Gaussian process model. Consequently, EMVU is suitable to nonlinear process monitoring. A cross-validation algorithm is designed to determine the dimensionality of the EMVU output space. For online monitoring, three different types of monitoring indices are developed, including squared prediction error (SPE), Hotelling-$T^2$, and the prediction variance of the outputs. In addition, a fault isolation algorithm based on missing data analysis is designed for EMVU to identify the variables contributing most to the faults. The effectiveness of the proposed methods is verified by the case studies on a numerical simulation and the benchmark Tennessee Eastman (TE) process.

Key words: process monitoring, fault isolation, nonlinear, manifold learning, maximum variance unfolding.

* Corresponding author: Tel: 886-3-5713690, Fax: 886-3-5719408, Email: yyao@mx.nthu.edu.tw
1. Introduction

In manufacturing industry, efficient process monitoring and fault isolation are critically important due to the increasing demands on economy, safety and environmental protection. Multivariate linear projection methods, such as principal component analysis (PCA), have been widely used [1-5], for their abilities to handle high dimensional and highly correlated data. However, many industrial processes are in essence nonlinear. The performance of PCA degrades when dealing with such processes. The early idea of nonlinear process monitoring comes from the field of artificial intelligence. Kramer designed a nonlinear version of PCA based on auto-associative neural networks [6]. However, such model is difficult to train. Another well known nonlinear PCA algorithm was developed by Dong and McAvoy, by combining principal curves and neural networks [7]. This algorithm also suffers from computational complexity, since a convoluted nonlinear optimization problem has to be solved. Kernel principal component analysis (KPCA) [8] handles the problem of nonlinear data analysis in another way, which maps the data in the original input space into a higher- or even infinite-dimensional feature space before performing PCA. In 2004, Lee et al. extended the applications of KPCA from the field of machine learning to systems engineering [9]. From then on, KPCA becomes a popular method in process monitoring.

Comparing to nonlinear PCA based on neural networks, KPCA provides a much simpler computational framework, since it only employs linear algebra to solve an eigenvalue problem. In addition, KPCA is able to handle a wide range of nonlinearities, attributing to its flexibility of using different kernels. However, KPCA has two major disadvantages. The first is that the underlying manifold structure of data is not considered in the process modeling procedure. Hence, the performance of the KPCA-based monitoring largely depends on the empirically selected kernel function. Secondly, for applications of unsupervised kernel methods like KPCA, the selection of kernel parameters is always problematic. As discussed by Shao and Rong [10], it is even difficult to specify an appropriate number of principal components (PCs) retained in the final KPCA model, if the parameters of the kernel function are not properly determined.

In recent years, manifold learning techniques have been intensively researched in the field of machine learning [11-15]. Such techniques can be divided into two categories that are global and local, respectively. Comparing to the local techniques, the global techniques, such as maximum variance unfolding (MVU) [14, 15] and Isomap [12], provide more faithful representations of the global data structure and have metric-preserving properties which are easier to understand [16]. Unlike Isomap, MVU does not need the assumption of convexity and therefore works in more general cases. Furthermore, in MVU, the kernel matrix is automatically learned from training data. Hence, MVU avoids both deficiencies of KPCA. It is natural to think about utilizing MVU in nonlinear process monitoring. However, such application is not as straightforward as it seems. In MVU, only a data-dependent kernel matrix is calculated, while the kernel function remains unknown. As a result, this technique is merely able to deal with training data, but cannot be extended to "out-of-sample" data points. Here, the "out-of-sample" extension does not necessarily mean "extrapolation". Instead, it refers to unfolding any new samples that are not included in the training set. Such new samples can be either "interpolation" or "extrapolation" of the training data. Obviously, such characteristic is not desirable in process monitoring. Recently, Shao and Rong [10] attempted to model the mapping between the input space and the MVU output space with a multivariate linear regression model. However, the use of linear regression is questionable, since the relationship between the high dimensional inputs and the low dimensional outputs is usually nonlinear.
In this paper, an extended MVU (EMVU) method is proposed to model and monitor the nonlinear process data, which inherits the advantages from both manifold learning and Gaussian process. EMVU firstly applies MVU to the training data collected from normal operations. Thus, a lower dimensional output space is constructed. Then, a Gaussian process regression (GPR) model is utilized to approximate the nonlinear mapping between the inputs and the outputs. Distinct from the conventional MVU method, EMVU has the ability to perform dimensionality reduction of new samples. Consequently, the EMVU model can be used in online process monitoring. To determine the number of reduced dimensions in EMVU, a cross-validation algorithm is developed. For process monitoring, the statistics of Hotelling-$T^2$ and squared prediction error ($SPE$) are calculated to assess the operational status of the investigated process. Different from those used in linear PCA, both statistics are calculated in the reduced output space instead of the original input space. In addition, the predictive variance of the EMVU outputs is also monitored, a quantity measuring the uncertainty of the prediction. As shown, such measurement is a good complement to $T^2$ and $SPE$. For fault isolation, a missing-data based algorithm is developed to identify the variables contributing most to the fault.

The organization of this paper is as follows. In section 2, the EMVU method is proposed, including manifold learning with MVU, modeling the nonlinear mapping between the input and the output space using GPR, and the determination of the output dimensionality by cross-validation. Section 3 presents the EMVU-based process monitoring, including the introduction of the monitoring statistics, and the major steps of offline modeling and online fault detection based on EMVU. The issue of fault isolation is discussed in Section 4. Then, the effectiveness of the proposed method is evaluated and verified by the case studies with a numerical simulation and the benchmark Tennessee Eastman (TE) process in Section 5. Finally, conclusion is drawn in Section 6.

2. Extended maximum variance unfolding
2.1. Maximum variance unfolding

In the first step of EMVU, MVU [14, 15], also known as semi-definite embedding (SDE) [17], is utilized for nonlinear dimensionality reduction of the training data that are collected under normal operation. The basic idea of MVU is based on the finding that high dimensional data usually lie on a low dimensional manifold. Different from KPCA, MVU does not project the input data to a higher dimensional feature space by subjectively specifying certain kernel function. Instead, MVU constructs a kernel matrix directly by maximizing the overall variance in the reduced output space while preserving the angles and distances between nearby observations.

Supposing that the input data are denoted as $x_i \in R^D (i = 1,2,\cdots,n)$, MVU unfolds them to the low dimensional outputs $y_i \in R^d (i = 1,2,\cdots,n$ and $d << D)$ by solving the following optimization problem subject to some local constraints.
Maximize $\sum_{i,j} \| y_i - y_j \|^2$ subject to

$$\sum_{i} y_{i} = 0,$$

$$\| y_i - y_j \|^2 = \| x_i - x_j \|^2 \text{ for all } (i, j) \text{ with } \eta_{ij} = 1,$$

where $\eta_{ij} = 1$ indicates that inputs $x_i$ and $x_j$ are k-nearest neighbors. Such optimization presents that the sum of the pairwise distances between the outputs is maximized. The first constraint centers the outputs on the origin. The second constraint enforces the distances between neighboring outputs equal to the distances between corresponding inputs, so that the local geometric properties are preserved during the unfolding. By doing so, a unique solution can be achieved.

To avoid local minima, the above optimization problem is reformulated as a semidefinite program (2) by defining the kernel matrix $K$ with entries $K_{ij} = y_i \cdot y_j = y_i^T y_j$.

Maximize $\text{Tr}(K)$ subject to

$$\sum_{i,j} K_{ij} = 0,$$

$$K_{ii} - 2K_{ij} + K_{jj} = \| x_i - x_j \|^2 \text{ for all } (i, j) \text{ with } \eta_{ij} = 1,$$

$K \succeq 0$.

The first two constraints have the same meaning with those in (1), while the $K \succeq 0$ makes the kernel matrix $K$ to be positive semidefinite. Such a semidefinite program is convex and can be efficiently solved with many off-the-shelf solvers, e.g. the CSDP library [18].

After the kernel matrix $K$ is learned, the outputs $y_i$ are calculated by singular value decomposition (SVD). Let $v_{a\alpha}$ be the $i$-th element of the $\alpha$-th eigenvector, corresponding to the $\alpha$-th largest eigenvalue $\lambda_{a\alpha}$. Then $K$ can be decomposed as:

$$K_{ij} = \sum_{a=1}^{n} \lambda_{a\alpha} v_{a\alphai} v_{a\alpha j}.$$

Accordingly, the $\alpha$-th element $y_{ia\alpha}$ of the output $y_i$ is obtained as:

$$y_{ia\alpha} = \sqrt{\lambda_{a\alpha}} v_{a\alphai}.$$

By reasonably truncating the elements of $y_i$, the low dimensional outputs are calculated.

There are only two parameters to be specified in the above algorithm, the number $k$ of the nearest neighbors and the reduced dimensionality $d$ of the output space. The optimal number of $k$ is related to the characteristics of data. There are some guidelines to be followed in the determination of $k$. First, the number of neighbors should be large enough to preserve the topology of the data set. Second, $k$ cannot be too large, so that the basic assumption of locality is not violated. An alternative to setting a fixed number $k$ is to choose all points within an $\varepsilon$-ball as the neighbors of a data point. By doing so, different numbers of neighbors can be selected for each point. For the specification of $d$, eigenvalue spectrum is commonly utilized in the conventional MVU, where a
“large” gap between the $d$-th and $(d+1)$-th eigenvalues indicates the most desired dimensionality $d$. Such approach can also be adopted in EMVU. However, the definition of “large” is sometimes ambiguous. In such situation, selecting $d$ based on eigenvalue spectrum may be subjective. Enlightened by the cross-validation approach [19] widely used in model parameter selection, a cross-validation algorithm will be proposed for EMVU. Therefore, a most suited $d$ can be automatically determined for process monitoring. The details about the cross-validation procedure will be provided in the later part of this paper.

2.2. Modeling the nonlinear mapping between input and output spaces

The procedures in the last section show that the conventional MVU only computes a data-dependent kernel matrix $K$ instead of a kernel function, suggesting that it can only unfold training samples and has no ability dealing with samples different from training ones. Therefore, such method cannot be directly applied to process monitoring which requires continuous projection of new process measurements to the reduced output space, unless the mapping between the input and output spaces can be correctly modeled. Shao and Rong [10] proposed to model such mapping with multivariate linear regression. However, such modeling is inappropriate, since the mapping between two spaces in MVU is usually nonlinear.

A numerical example, originally reported in [20], can illustrate this point. In this example, data are generated by the following equations:

\begin{align}
    x_1(t) &= 0.5t^2 - 2t + 0.5, \\
    x_2(t) &= t^2 + t + \sin(\pi t), \\
    x_3(t) &= 2t^2 - t - 2\cos(\pi t).
\end{align}

With $t$ taking random values in the range [-1, 1], the data comprising 200 samples are generated and stored in an input matrix $X = [x_1 \ x_2 \ \cdots \ x_{200}]^T$, where $x_i (i = 1, 2, \cdots, 200)$ is a sample vector with three variables. Random noise in the range (-0.3, 0.3) are added into the data. MVU is conducted on these data for dimensionality reduction, leading to a single output vector $y$. According to Shao and Rong's method [10] (denoted as "SR method" in this paper), a linear regression model is built to relate the inputs $X$ and the output $y$:

\begin{align}
    y &= Xb + e,
\end{align}

where $b$ is the regression coefficient vector, and $e$ is the residual vector. The nonlinearity of the relationship between $X$ and $y$ can then be tested by plotting the real output from MVU against the predicted output values estimated as $\hat{y} = Xb$. If the points are not symmetrically distributed around a diagonal line, the relationship is nonlinear. Alternatively, the nonlinearity can also be detected by a plot of residuals versus the predicted output values. Only if the points are symmetrically distributed around a horizontal line in such plot, the relationship is linear. Both types of plots are shown in Fig. 1. It is clear that the mapping from the MVU input and the output is nonlinear and can only be approximated reasonably with a nonlinear regression model.

As the second step in EMVU, Gaussian process regression (GPR) [21], a method preferred in nonlinear regression for its good performance in practice and desirable analytical properties, is
adopted to model the nonlinear mapping between two spaces. As a Bayesian regression method, GPR assumes that the target function is Gaussian process prior and performs prediction on the basis of Bayesian inference.

Consider \( \{ x_i, y_{ia} \} \), where \( x_i \) is the \( i \)-th sample in the training data, \( y_{ia} \) is the \( \alpha \)-th MVU output corresponding to \( x_i \), calculated by equation (4), \( i = 1, 2, \ldots, n \). To model the mapping from \( x_i \) to \( y_{ia} \), \( x_i = [x_{i1}, x_{i2}, \ldots, x_{ip}] \) is considered as a \( D \)-dimensional predictor vector, and \( y_{ia} \) is the corresponding scalar response. Thus, for each \( \alpha \), a Gaussian process for regression is defined such that the response, \( y = [y_{1\alpha}, y_{2\alpha}, \ldots, y_{n\alpha}]^T \), has a Gaussian prior distribution with zero mean and an \( n \times n \) covariance matrix, i.e.

\[
y \sim N(0, C),
\]

where \( C \) is the covariance matrix whose \( ij \)-th entry \( C_{ij} \) is defined by a covariance function (kernel) \( C \) as \( C_{ij} = C(x_i, x_j) \). The kernel functions commonly used in KPCA, such as the Gaussian radial basis function (RBF) kernel, polynomial kernel, and etc., can also be adopted in GPR modeling. Besides, a widely used covariance function in GPR is:

\[
C(x_i, x_j) = a_0 + a_1 \sum_{k=1}^{D} x_{ik} x_{jk} + v_0 \exp(-\sum_{k=1}^{D} w_k (x_{ik} - x_{jk})^2) + \delta y \sigma^2,
\]

where \( \delta_{ij} = 1 \) if \( i = j \), otherwise \( \delta_{ij} = 0 \). The non-negative parameters \( \theta \) defining the covariance function are denoted as hyper-parameters, where \( \theta = [a_0, a_1, v_0, w_1, \ldots, w_D, \sigma^2] \). There are four terms in such covariance function. The first term represents constant bias, while the second term models the linear correlation. The exponential term capture the strong correlation information between the outputs and the nearby inputs. The last term considers the random noise. By including both linear and nonlinear terms in the covariance function, the Gaussian process has the ability to handle both linear and nonlinear data structures [22].

In GPR, the covariance function is a crucial ingredient determining the performance of the prediction model. Therefore, it is important to set the hyper-parameters properly. Comparing to KPCA kernel parameter selection where there is no general rule, the estimation of the hyper-parameters in GPR is more clearly guided, which is based on the maximization of the log-likelihood:

\[
L(\theta) = -\frac{1}{2} \log |C| - \frac{1}{2} y^T C y - \frac{n}{2} \log 2\pi.
\]

To calculate \( \theta \), the partial derivative of the log-likelihood with regard to each single hyper-parameter \( \theta \) is derived as:

\[
\frac{\partial L(\theta)}{\partial \theta} = -\frac{1}{2} \text{tr}(C^{-1} \frac{\partial C}{\partial \theta}) + \frac{1}{2} y^T C^{-1} \frac{\partial C}{\partial \theta} C^{-1} y.
\]

Using (12), any gradient based optimization algorithm can be used to obtain the hyper-parameters. An alternative method is to use Markov Chain Monte Carlo (MCMC) [21]. A Matlab implementation of the GPR model is available from the URL: http://www.gaussianprocess.org/gpml/code/matlab/doc/.
For any predictor vector \( x^* \) that may or may not be included into the training dataset, the predictive distribution of the corresponding response \( y_a^* \) conditional on the hyper-parameters is also Gaussian, whose mean and variance can be calculated as:

\[
E(y_a^*) = k^T(x^*)C^{-1}y,
\]

\[
Var(y_a^*) = C(x^*, x^*) - k^T(x^*)C^{-1}k(x^*),
\]

where \( k(x^*) = \begin{bmatrix} C(x^*, x_1) & C(x^*, x_2) & \cdots & C(x^*, x_n) \end{bmatrix} \). Hence, the GPR model not only provides a prediction of the output \( \hat{y}_a^* = E(y_a^*) \), but also estimates the prediction uncertainty in terms of the variance \( Var(y_a^*) \).

Note that the GPR model discussed in above can only deal with a single response. However, MVU may result in an output space whose dimensionality \( d \) is larger than one, which means that there is more than one response to be predicted. To handle such situation, multiple GPR models may be built for various outputs. Alternatively, this problem can also be addressed by treating Gaussian processes as white noise sources convoluted with smoothing kernels [23]. In the following part of this paper, multiple GPR models are utilized for illustration.

To compare the prediction accuracies of GPR and linear regression in MVU mapping, the 200 samples generated from (5)-(7) and the corresponding MVU outputs are divided into two sets, where 150 samples form a training set and the other 50 samples are included into a test set. Both GPR and linear regression models are built based on the training set. The covariance function used in the GPR model is that described in (10). Using GPR, the root mean squared errors (RMSE) for the training and test sets are 0.0609 and 0.0992, respectively. In comparison, the results from the linear regression model are much worse. The corresponding RMSE values are 0.8913 and 0.5698.

2.3. Output dimensionality determination

As discussed in subsection 2.1, in some situations, the “large” gap between the eigenvalues in the eigenvalue spectrum may not be clear enough to be identified. Under such situation, a more objective criterion, e.g. cross-validation, is desirable. A commonly used quantity in cross-validation is the predictive error sum of squares (PRESS). When applied in component model selection, PRESS for leave-one-out cross-validation (LOOCV) can be defined as:

\[
PRESS_f = \sum_{i=1}^{n} \sum_{j=1}^{D} (x_{ij} - \hat{x}_{ij}^f)^2,
\]

where the subscript and superscript \( f \) denotes the number of components included in the model, \( x_{ij} \) is the \( j \)-th variable in the \( i \)-th sample in the training dataset, \( \hat{x}_{ij}^f \) is the prediction of \( x_{ij} \) based on the training samples excluding \( x_i \). However, such quantity cannot be easily established for EMVU dimensionality determination, since the inverse mapping from the output space to the input space is undefined, and therefore the prediction of \( x_{ij} \) is unachievable. Thus, the PRESS is redefined using the information in the reduced output space as follows:
where \( c \) is the number of nonzero eigenvalues in the kernel matrix \( K \), \( K_{ii} \) is the \((i,i)\)-th element in \( K \), \( \hat{K}_{ii} \) is the estimation of \( K_{ii} \) using the EMVU output \( \hat{y}_i = [\hat{y}_{i1} \hat{y}_{i2} \cdots \hat{y}_{ic}] \) corresponding to the \( i \)-th training sample \( x_i \), \( y_{ij} \) is the \( j \)-th element in \( y_i \), and \( \hat{y}_{ij} \) is the prediction of \( y_{ij} \) with the GPR model based on the training samples excluding \( x_i \) and \( y_i \). Such quantity reflects the cross-validation prediction accuracy of the kernel matrix based on certain output dimensionality.

Based on the PRESS defined in (16), the detailed procedures of the EMVU modeling with cross-validation are as follows.

(a) Select the number of the nearest neighbors \((k)\) used in EVMU modeling, following the guidance introduced in subsection 2.1.

(b) Compute the kernel matrix \( K \) based on the input data \( x_i \in R^D (i = 1, 2, \cdots, n) \), by solving the optimization problem in (2).

(c) Decompose \( K \), as in (3), by performing SVD.

(d) Calculate the MVU outputs \( y_i \in R^c \) for the inputs, where \( c \) is the number of nonzero eigenvalues. The dataset \( \{x_i, y_i\} (i = 1, 2, \cdots, n) \) will be used in GPR model training.

(e) Set \( g = 1 \).

(f) Retain the \( g \)-th pair of training sample, \( \{x_g, y_g\} \), as the validation data for model testing, and use the remaining \((n - 1)\) training samples as the training set for GPR modeling of the nonlinear mapping.

(g) Approximate the nonlinear mapping from inputs to outputs using GPR, based on the training set. The hyper-parameters in the covariance functions are determined based on Bayesian model selection criterion.

(h) Predict the outputs \( \hat{y}_i^c = [\hat{y}_{i1} \hat{y}_{i2} \cdots \hat{y}_{ic}] \) for the retained validation sample \( \{x_g, y_g\} \) using (11).

(i) Set \( g = g + 1 \), and repeat the steps (f) to (h), until \( g = n \).

(j) Set \( f = 1 \).

(k) Calculate \( PRESS_f \) using (16).

(l) Set \( f = f + 1 \), and repeat the step (k), until \( f = c \).

(m) Specify \( d \) as:

\[
d = \arg \min_f(PRESS_f) \tag{17}
\]

(n) Truncate the elements in \( y_i \) according to the determined output dimensionality \( d \).
(o) Re-apply GPR to all $x_i$ and the corresponding truncated $y_i \in R^d (i = 1, 2, \ldots, n)$, to achieve the final nonlinear mapping between the input space and the MVU output space.

(p) Calculate the EMVU outputs for the training samples using (11), based on the models built in the last step.

One may notice that the $PRESS$ value utilized in the above cross-validation algorithm looks at the prediction accuracy of the diagonal elements in the kernel matrix $K$. In fact, (16) can be further revised to take the off-diagonal elements in $K$ into consideration. However, according to our experiences, the $PRESS$ defined in (16) achieves reasonable results in EMVU model building. Therefore, in this paper, such quantity is adopted for illustrating the proposed method.

3. EMVU-based process monitoring

3.1. Monitoring statistics

For process monitoring, three different types of monitoring indices are utilized, including the Hotelling-$T^2$ statistic, the $SPE$ statistic and the predictive variance of the EMVU outputs. The details of these indices are introduced in below.

Similar to PCA and KPCA, the proposed method utilizes the Hotelling-$T^2$ statistic to quantify the variability within the model. Here, $T^2$ is formulated as:

$$T^2 = \left[ \hat{y}_1 \cdots \hat{y}_d \right]^T \Lambda^{-1} \left[ \hat{y}_1 \cdots \hat{y}_d \right],$$

where $\hat{y}_a$ is the $a$th EMVU output variable (prediction of $y_a$), $\Lambda$ is the covariance matrix of the predicted outputs, and $d$ is the dimensionality of the reduced output space. Thus, $T^2$ is defined as the sum of the normalized squared outputs.

As a complement of $T^2$, $SPE$ is also necessary for process monitoring, which measures the model fitness. However, it is problematic to construct $SPE$ in original input space, since the inverse mapping from the outputs to the inputs is unknown. Therefore, the $SPE$ statistic is also calculated in the EMVU output space. The derivation of $SPE$ is similar to that of the $PRESS$ index in (16):

$$SPE = \sum_{j=1}^{c} (\hat{y}_j)^2 - \sum_{j=1}^{d} (\hat{y}_j)^2 = \sum_{j=d+1}^{c} (\hat{y}_j)^2.$$

It may be noticed that such definition is also similar to that of the $SPE$ statistic in KPCA [9].

The third type of monitoring index is the predictive variances of the EMVU outputs, $\text{Var}(y_a)$, derived using (14), where $1 \leq a \leq d$. $\text{Var}(y_a)$ is a measure of the prediction uncertainty of $y_a$, and provides additional information for process monitoring. A too large value of $\text{Var}(y_a)$ implies that the predictor in the GPR model may locate in a region not spanned by the normal operating data, indicating an out-of-control situation.
Instead of making assumptions on the distributions of $T^2$, $SPE$ and predictive variance, kernel density estimation (KDE) [24] is adopted to calculate the control limits (CLs) for all the indices. KDE approximates the data distributions on $(-\infty, +\infty)$, but the values of all the three monitoring indices are nonnegative. Therefore, KDE is utilized to estimate the distributions of the logarithmic values of $T^2$, $SPE$ and the predictive variance instead of the raw values. Then, the confidence bounds on the logarithmic values are transformed to the CLs of the monitoring indices.

3.2. Offline modeling and online fault detection based on EMVU
When applying the proposed EMVU method to process modeling and monitoring, the overall procedures are as follows.

Offline process modeling:
(a) Normalize the historical normal operating data to zero mean and unit variance.
(b) Divide the normalized data $x_i (i = 1, 2, \cdots, n)$ into two sets: the training set and the validation set.
(c) Train the EMVU model with the data in the training set, following the steps introduced in subsection 2.3.
(d) Apply the model to the validation set, and derive the $T^2$ and $SPE$ statistics using (18) and (19). Also record the predictive variances of the EMVU outputs calculated according to (14).
(e) Determine the control limits for all the monitoring indices, using KDE.

Online fault detection:
(a) Record the online measurements, and normalize them with the mean and the standard deviation calculated in the modeling steps.
(b) Obtain the outputs in the reduced space of EMVU by performing GPR prediction as (11).
(c) Calculate the monitoring indices for the EMVU outputs.
(d) Monitor the indices and check whether they exceed the corresponding control limits.
(e) If the indices are well below the control limits, the process is regarded as normal, and the procedure returns to step (a); otherwise, a potential fault is detected, and fault isolation should be conducted.

4. Fault isolation
After a fault is detected, fault isolation is desired to identify the most responsible variables. Hence, the potential causes of the fault can be diagnosed. Contribution plots are widely used in multivariate statistical process monitoring for fault isolation [25, 26]. However, such methods were frequently criticized in recent years, for the existence of the “smearing” effect that often leads to misleading results [27]. To overcome such problem, the idea of fault reconstruction [28] can be adopted. The original idea of such method is to deal with the situation that each time there is only one process variable contribute to the fault. In fault isolation, each variable is treated as if it was missing and the monitoring statistic is re-estimated. Consequently, the variable most significantly reduces the monitoring statistic is considered to be the cause of the alarm, since the
correct reconstruction removes the effect of the fault. Similar idea has been extended to analyze the joint effect of multiple variables. Inspired by the previous research, a fault isolation algorithm is developed for the EMVU based process monitoring.

A straightforward procedure is as following, starting with \( f = 1 \).

(a) Select \( f \) variables to be the potential faulty variables in the sample vector \( \mathbf{x} \).

(b) Treat the \( f \) variables as missing variables, and replace them by estimations.

(c) Reconstruct the monitoring indices based on the adjusted sample from step (b).

(d) Check whether the monitoring indices are below the corresponding control limits.

(e) If yes, the corresponding \( f \) variables are isolated as the cause of the fault, and the algorithm is terminated. Otherwise, select another group of \( f \) variables and return to step (b), until all possible combinations of \( f \) variables have been selected once.

(f) Set \( f = f + 1 \), and return to step (a).

Different types of methods can be utilized to estimate missing data [29]. For illustration, the idea of nearest neighbor imputation is adopted in this paper, for its easy application and good properties such as non-parametric and distribution-free. Suppose a normalized sample \( \mathbf{x}^* \) is divided into two parts, \( \mathbf{x}^*_\text{known} \) and \( \mathbf{x}^*_\text{missing} \), where \( \mathbf{x}^*_\text{known} \) consists of the known elements in \( \mathbf{x}^* \) and \( \mathbf{x}^*_\text{missing} \) contains all the unknown variables. In the first step of the proposed estimation method, \( \mathbf{x}^*_\text{known} \) is compared with the corresponding part of the normalized training samples in the historical dataset, \( \mathbf{x}_{i,\text{known}} \) (\( i = 1, 2, \cdots, n \)). Such comparison can be quantified simply by calculating the Euclidean distance \( \text{dist}_i \) between \( \mathbf{x}^*_\text{known} \) and \( \mathbf{x}_{i,\text{known}} \):

\[
\text{dist}_i = \left\| \mathbf{x}_{i,\text{known}} - \mathbf{x}^*_\text{known} \right\|, \tag{20}
\]

where a small \( \text{dist}_i \) indicates that the \( i \)-th sample in the training dataset is a neighbor near to the current sample, and vice versa. Based on such comparison, \( A \) number of nearest neighbors of the current batch are identified, according to the \( A \) number of smallest \( \text{dist}_i \) values. Then, the missing variables in \( \mathbf{x}^* \) are approximated by the weighted average of the corresponding part in the nearest neighbors:

\[
\hat{\mathbf{x}}^*_\text{missing} = \frac{1}{A_{\text{known}} \text{nearest neighbors}} \sum_{m} w_m \mathbf{x}_{m,\text{missing}}, \tag{21}
\]

where the weights \( w_m \) can be chosen to reflect the degree of similarity, e.g. \( w_m = \frac{1}{\text{dist}_m} / \sum_{i=1}^{A} \frac{1}{\text{dist}_i} \), or just selected to be 1. In the case study in the next section, the weights will be set to be 1 for simplicity.

Note that, besides taking average, there are alternative methods to estimate \( \mathbf{x}^*_\text{missing} \) based on the nearest neighbor information, e.g. the iterative majorization least-squares (IMLS) algorithm [30].
Equation (21) is adopted here for its simplicity. The proper value of the parameter $A$ depends on the process properties and the data characteristics. It may be determined using the procedure of cross-validation or based on process knowledge, and can be either equal or unequal to number of nearest neighbors used in the EMVU algorithm.

A problem of the previous fault isolation procedures is that the computation may be burdensome, especially when there are multiple variables contributing to the fault. In such situation, a branch and bound (BAB) algorithm can be adopted to efficiently find the target variables in a short time. The details about the BAB algorithm are not covered in this paper, to keep the paper within a reasonable length without losing focus. Interested readers are suggested to refer to [31].

5. Case study

In this section, EMVU will be applied to fault detection in both a simple numerical example and the benchmark Tennessee Eastman (TE) process. The effectiveness of the proposed method will be verified through the comparison with Shao and Rong's method [10] that combines MVU with linear regression, and KPCA. Moreover, the missing-data based fault isolation will also be illustrated using TE process.

5.1. A simple numerical example

Consider the following system with three variables.

$$x_1 = 10 \times u^3 + 1.5 \times u^2 - 1.5 \times u + 0.1 + \varepsilon_1,$$

$$x_2 = 30 \times s + \varepsilon_2,$$

$$x_3 = u + \varepsilon_3,$$

where $\varepsilon_1$, $\varepsilon_2$ and $\varepsilon_3$ are independent random noise variables with small magnitudes, $s \in (0,1)$, and $u = s^2 - 1$. Normal data consist of 500 samples generated by the above equations. To test the monitoring efficiency, 100 faulty samples are generated based on equations (25)-(27).

$$x_1 = 0.5 \times u^2 - 2 \times u + 0.5 + \varepsilon_1,$$

$$x_2 = u^2 + u + \sin(\pi \times u) + \varepsilon_2,$$

$$x_3 = u^2 - u - 1 \times \cos(\pi \times u) + \varepsilon_3.$$

All the data are visualized in Fig. 2, where the normal samples are represented as solid circles and the faulty samples are plotted as open circles. The figure shows that a number of faulty samples nearly overlap with the normal data, although the models for different system status are quite different. Therefore, the fault signature in these samples is not significant, while other faulty samples should be recognized easily.

For process monitoring, an EMVU model is built based on the normal data, following the steps introduced in subsection 2.3. For comparison, an SR model, a KPCA model and a linear PCA
model are also constructed using the same training data. In both EMVU and SR models, the number of nearest neighbors, $k$, used in MVU model training is selected to be 5, while the dimensionality of the MVU output space is determined as 2. The eigenvalue spectrum (not provided in this paper) shows that the first two outputs capture more than 99% variation information in the output space. To approximate the nonlinear mapping between MVU input and output spaces, the EMVU model conducts GPR models with a simple second-order polynomial covariance function defined as $k(x_i, x_j) = \left( b + \sum_{k=1}^{D} x_{ik} x_{jk} \right)^2$, while the SR method builds linear regression models. The parameters in the GPR models are determined automatically based on the maximization of the log-likelihood function. The kernel function used in KPCA has the same form as the covariance function adopted by EMVU. Totally 6 principal components are retained in the KPCA model, which lead to the most efficient monitoring. In the linear PCA model, two principal components are retained, which explains more than 90% variance information contained in the original data.

Then, all three models are utilized to monitor the faulty samples. The control charts of EMVU and SR are plotted in Fig. 3 and Fig. 4, respectively. In all control charts, the control limits corresponding to the confidence level at 99% are adopted. The missing alarm rates of EMVU and SR are 24% and 36%, respectively. Obviously, EMVU outperforms the SR method. The monitoring results can be explained by looking at Fig. 5 and Fig. 6 visualizing the output spaces of both models. In both figures, the outputs of the normal samples form a two-dimensional model plane. As seen in Fig. 5, in EMVU's output space, many faulty samples locate outside the model plane, except the points overlapping with the normal data in the original input data space. Therefore, the $SPE$ statistic measuring the model fitness detects the fault at a large number of points. On the other hand, the $T^2$ statistic seems not very sensitive in this particular case, since the variability of the faulty data within the model is not significant. This is the reason why the $SPE$ control chart performs better than the $T^2$ control chart as seen in Fig. 3. The bottom plot shows the monitoring result based on the predictive variance of the first EMVU output, which provides the most efficient detection among all three control charts in Fig. 3. As introduced in previous, both the $SPE$ and $T^2$ statistics of EMVU are calculated in the output space, while the predictive variance is a measure of prediction uncertainty. Therefore, the predictive variance contains the information of similarity between the training data and the test data in the original input space, which is a necessary complement of $SPE$ and $T^2$. Similar pattern can be observed in Fig. 6. Hence, in the SR model, the performance of the $SPE$ control chart is also better than that of the $T^2$ control chart. It should be noticed that, in the SR method, the $SPE$ statistic is calculated in the input space instead of the output space. Therefore, it expresses different information from the $SPE$ statistic in the EMVU model. In this case, KPCA performs better than SR, but worse than EMVU. The missing alarm rate of KPCA is 31%. In addition, the model complexity of KPCA is higher than that of EMVU, since KPCA utilizes 6 PCs to capture the characteristics of the system while EMVU only retains 2 outputs. PCA has the worst fault detection efficiency among all four methods, which has a missing alarm rate of 41%. Such results are not surprised, since the process is nonlinear while PCA is a linear method. In both Fig. 7 and Fig. 8, the $SPE$ charts show better detection ability, confirming the results based on the EMVU and SR methods.

5.2. Tennessee Eastman (TE) process
The benchmark TE process [32] has been widely accepted in performance test of control algorithms and monitoring approaches. A large number of research papers have used such process to test the effectiveness of nonlinear models in process monitoring, e.g. [33-35]. In this section, the EMVU method, the SR method and the popular KPCA method will be further compared based on this process. The TE process has five main units: a reactor, a product condenser, a vapour-liquid separator, a product stripper and a recycle compressor. Streams of the plant consist of eight components among which the components A, B and C are the gaseous reactants fed to the reactor while G and H are the products. There are totally 41 measured variables and 12 manipulated variables recorded. The measured variables can be divided into two groups, where the first 22 variables are measured continuously during the process operation, and the remaining 19 variables are measured by the gas analyzers and can be regarded as the quality variables. 21 process faults may occur to the TE process [36], which are often used to test the fault detection efficiency of different monitoring approaches. The descriptions of process faults are listed in Table 1.

The TE process data are available at [http://web.mit.edu/braatzgroup/links.html](http://web.mit.edu/braatzgroup/links.html), including both normal operation data and all 21 types of faulty data. The entire operation time in each testing dataset is 48 hours, during which the faults are introduced after 8 hours. The sampling time interval is 3 min. Therefore, there are total 960 sampling intervals in each testing set. In this paper, the 22 continuously measured variables and all manipulated variables except the agitation speed are utilized for process modeling. The agitation speed stays constant during the operation. Therefore, the values of this variable are not provided in the downloaded data. The list of the modelled variables is summarized in Table 2.

The monitoring models are trained with the 500 normal operation samples. In EMVU and SR modeling, the number of nearest neighbors, $k$, is set to be 7 based on the guideline introduced in subsection 2.1. For modeling the nonlinear mapping, the structure of the covariance function for GPR is selected to have the form in equation (10) due to its flexibility, and the hyper-parameters are calculated automatically based on the maximization of the log-likelihood function. The dimensionality $d$ of the EMVU output space is determined utilizing the cross-validation procedure proposed in subsection 2.3, which is equal to 6. The dimensionality of the SR output space is also selected to be 6. For comparison, a KPCA model is also built. The popular Gaussian kernel, $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / c)$, is adopted, since it can project data into an infinite dimensional feature space and works well in many cases. The parameter $c$ is selected according to the rule recommended in [9]: $c = 10D\sigma^2$, where $D$ is the input dimension and $\sigma^2$ is the variance of the (normalized) training samples. To achieve efficient monitoring, the number of retained PCs is chosen to be 25, which is significantly higher than the dimensionality of the EMVU model. In EMVU, the statistics of $T^2$, SPE and the predictive variance of the first output are utilized in process monitoring, while SR and KPCA adopt $T^2$ and SPE. In all models, the control limits of the monitoring statistics are calculated using KDE, corresponding to the confidence level at 99%.

The monitoring performances of the three models are compared using two metrics that are missing alarm rate and detection delay. Following the definition in [10], a missing alarm occurs when no monitoring statistic identifies a faulty sample, while the detection delay is defined as the time gap between the occurrence of a fault and three consecutive alarms raised by any monitoring indices for the first time. Since all methods adopt the 99% control limits, they have the same false alarm rate and zero fault alert rate.
rate that is equal to 1%, based on which the comparison is fair. The comparison results are shown in Fig. 9 and Fig. 10. EMVU provides obviously lower missing alarm rates than those of SR or KPCA for most faults. Meanwhile, EMVU outperforms SR and KPCA in terms of detection delay as well. In addition, the computation time of EMVU in online monitoring is about 0.0015 second per sample which is acceptable. All computations are carried out on a desktop computer with Intel® Core™ i7-2600 Quad-Core Processor (3.4 GHz) and 8GB RAM using MATLAB® 2007b.

It is interesting to investigate the monitoring performances of EMVU and KPCA on fault 5. Fault 5 is a step change in the condenser cooling water inlet temperature occurring to the process at the 161st sampling interval. Fig. 11 shows the monitoring results based on KPCA. Although both the $T^2$ and SPE control charts detect the fault efficiently, the number of missing alarms increases significantly after the 370 sampling interval. Because of the existence of the feedback control, the fault becomes insignificant as time goes on and hence difficult to detect, although the process behaviour still deviates from normal operation. In contrast, EMVU performs much better. Fig. 12 shows that, in spite of the relatively high missing alarm rate of $T^2$, the control chart of predictive variance provides consistent detection, while the SPE values also go outside of the control limit frequently. In the detection of fault 5, SR has similar performance to EMVU. Therefore, the monitoring results based on SR are not plotted here.

The effect of the model parameters on process monitoring is also worth to discuss. As is known, the kernel parameter selection in KPCA is still an open problem and usually based on engineering experience, since the procedure of KPCA modeling is unsupervised. Although a practical rule has been provided by [9], there is no guarantee on the monitoring efficiency. Take the detection results of fault 6 as an example. If the kernel parameter $c$ is selected to be 330, KPCA can detect the fault quite well, as shown in Fig. 13. However, when $c$ is specified improperly, e.g. $c = 33$, Fig. 14 shows that the control charts cannot detect the fault. Instead, the values of the monitoring statistics $T^2$ and SPE both decrease. Such phenomena have also been observed in previous research [37] when the value of $c$ is relatively small and the amplitude of disturbance is very large. The parameter $c$ determines the resolution of the kernel function, and hence determines the types of faults the KPCA model can detect. On the contrary, EMVU has no problem of parameter selection, since all hyper-parameters are automatically determined through supervised learning. The monitoring results of EMVU are plotted in Fig. 15.

After a fault is detected, fault isolation is conducted based on the method introduced in section 4. For illustration, the isolation results of fault 5 and fault 6 are display in Fig. 15 and Fig. 16, where the reconstructed values of the monitoring statistics at the first point after a fault is detected are plotted. In the figures, the x-axis corresponds to the index of reconstructed variable(s), where variable 0 indicates no reconstruction. For clear display, these bar-plots only show a part of the reconstruction results. As shown in Fig. 16, for fault 5, if only one variable is treated as missing variable, variable 22 provides the best reconstruction results, indicating that the separator cooling water outlet temperature contributes most to this fault. However, the value of the predictive variance is still higher than the control limit. Then, two variables are treated as missing variables at the same time. The best result is provided by the combination of variables 22 and 11 that makes all the reconstructed indices below the control limits. Such isolation result is reasonable, and helpful for finding out the root cause of the fault, i.e. the change in the condenser cooling water inlet temperature. Since the separator cooling water outlet temperature is directly influenced by the
change in condenser cooling water inlet temperature, it is not strange that variable 22 is the most contributing variable. Meanwhile, such fault is easy to propagate and affect the product separator temperature, i.e. variable 11, in a short time. The fault isolation result of fault 6 (Fig. 17) shows that variables 1 and 25 are the most contributing variables. Fault 6 is caused by a feed loss in component A, which is directly reflected by both identified variables.

5. Conclusions

In this paper, an EMVU method is developed for online process monitoring and fault isolation. Comparing to the popular KPCA method, EMVU has several attractive advantages. First, EMVU considers the manifold structure of data in process modeling, and leads lower-order models. Second, the parameters in the EMVU model can be determined easily. Especially, the best values of the hyper-parameters for the approximation of the nonlinear mapping can be determined automatically through a supervised procedure. On the other hand, there is no objective way to select the kernel parameters in a KPCA model, since such selection is unsupervised. Third, the predictive variance provides additional information for the EMVU-based monitoring, making this method more powerful in fault detection. A fault isolation procedure is also developed for EMVU, based on the idea of missing data analysis. Case studies on a simple numerical example and the benchmark TE process show the effectiveness of the proposed method. It is also worth noting that, for each monitored sample, there is usually more than one predictive variance value which can be estimated from an EMVU model. In fact, for each EMVU output variable, a variance value is calculated and can be utilized in process monitoring. When the dimensionality of outputs increases, it becomes inconvenient to monitor all the predictive variances separately. It may be desirable to find a way to summarize all the variance information in a single joint index. Such issue is out of the scope of this paper and will be studied in the future.

Acknowledgement

This work was supported in part by the National Science Council of R.O.C. under Grant No. NSC 102-2221-E-007-130- and in part by Grant No. 102N2074E1 of the Advanced Manufacturing and Service Management Research Center of National Tsing Hua University.
Figure 1. Test for nonlinearity with (a) a plot of the predicted output values versus the real output values, and (b) a plot of the residuals versus the predicted output values.
Figure 2. A simple numerical example: normal and faulty data in the original input space

Figure 3. Monitoring results based on the EMVU model
Figure 4. Monitoring results based on the SR model

Figure 5. Normal and faulty data in the output space of the EMVU model
Figure 6. Normal and faulty data in the output space of the SR model
Figure 7. Monitoring results based on the KPCA model

Figure 8. Monitoring results based on the linear PCA model

Figure 9. Comparisons between EMVU, SR and KPCA: missing alarm rate
Figure 10. Comparisons between EMVU, SR and KPCA: detection delay

Figure 11. KPCA monitoring results of fault 5
Figure 12. EMVU monitoring results of fault 5
Figure 13. KPCA monitoring results of fault 6, with the kernel parameter $c=330$

Figure 14. KPCA monitoring results of fault 6, with the kernel parameter $c=33$

Figure 15. EMVU monitoring results of fault 6
Figure 16. Isolation results of fault 5

Figure 17. Isolation results of fault 6
Table 1. Descriptions of process faults in the TE process

<table>
<thead>
<tr>
<th>Fault number</th>
<th>Fault cause</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDV(1)</td>
<td>A/C feed ratio, B composition constant</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(2)</td>
<td>B composition, A/C ration constant</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(3)</td>
<td>D feed temperature</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(4)</td>
<td>Reactor cooling water inlet temperature</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(5)</td>
<td>Condenser cooling water inlet temperature</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(6)</td>
<td>A feed loss</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(7)</td>
<td>C header pressure loss-reduced availability</td>
<td>Step</td>
</tr>
<tr>
<td>IDV(8)</td>
<td>A, B and C feed composition</td>
<td>Random variation</td>
</tr>
<tr>
<td>IDV(9)</td>
<td>D feed temperature</td>
<td>Random variation</td>
</tr>
<tr>
<td>IDV(10)</td>
<td>C feed temperature</td>
<td>Random variation</td>
</tr>
<tr>
<td>IDV(11)</td>
<td>Reactor cooling water inlet temperature</td>
<td>Random variation</td>
</tr>
<tr>
<td>IDV(12)</td>
<td>Condenser cooling water inlet temperature</td>
<td>Random variation</td>
</tr>
<tr>
<td>IDV(13)</td>
<td>Reaction kinetics</td>
<td>Slow drift</td>
</tr>
<tr>
<td>IDV(14)</td>
<td>Reactor cooling water valve</td>
<td>Sticking</td>
</tr>
<tr>
<td>IDV(15)</td>
<td>Condenser cooling water valve</td>
<td>Sticking</td>
</tr>
<tr>
<td>IDV(16)</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>IDV(17)</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>IDV(18)</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>IDV(19)</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>IDV(20)</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>IDV(21)</td>
<td>The valve fixed at steady state position</td>
<td>Constant position</td>
</tr>
</tbody>
</table>

Table 2. Process variables included in the monitoring model

<table>
<thead>
<tr>
<th>Variable number</th>
<th>Variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A feed</td>
</tr>
<tr>
<td>2</td>
<td>D feed</td>
</tr>
<tr>
<td>3</td>
<td>E feed</td>
</tr>
<tr>
<td>4</td>
<td>A and C feed</td>
</tr>
<tr>
<td>5</td>
<td>Recycle flow</td>
</tr>
<tr>
<td>6</td>
<td>Reactor feed rate</td>
</tr>
<tr>
<td>7</td>
<td>Reactor pressure</td>
</tr>
<tr>
<td>No.</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>8</td>
<td>Reactor level</td>
</tr>
<tr>
<td>9</td>
<td>Reactor temperature</td>
</tr>
<tr>
<td>10</td>
<td>Purge rate</td>
</tr>
<tr>
<td>11</td>
<td>Product separator temperature</td>
</tr>
<tr>
<td>12</td>
<td>Product separator level</td>
</tr>
<tr>
<td>13</td>
<td>Product separator pressure</td>
</tr>
<tr>
<td>14</td>
<td>Product separator underflow</td>
</tr>
<tr>
<td>15</td>
<td>Stripper level</td>
</tr>
<tr>
<td>16</td>
<td>Stripper pressure</td>
</tr>
<tr>
<td>17</td>
<td>Stripper underflow</td>
</tr>
<tr>
<td>18</td>
<td>Stripper temperature</td>
</tr>
<tr>
<td>19</td>
<td>Stripper steam flow</td>
</tr>
<tr>
<td>20</td>
<td>Compressor work</td>
</tr>
<tr>
<td>21</td>
<td>Reactor cooling water outlet temp.</td>
</tr>
<tr>
<td>22</td>
<td>Separator cooling water outlet temp.</td>
</tr>
<tr>
<td>23</td>
<td>D feed flow</td>
</tr>
<tr>
<td>24</td>
<td>E feed flow</td>
</tr>
<tr>
<td>25</td>
<td>A feed flow</td>
</tr>
<tr>
<td>26</td>
<td>A and C feed flow</td>
</tr>
<tr>
<td>27</td>
<td>Compressor recycle valve</td>
</tr>
<tr>
<td>28</td>
<td>Purge valve</td>
</tr>
<tr>
<td>29</td>
<td>Separator pot liquid flow</td>
</tr>
<tr>
<td>30</td>
<td>Stripper liquid product flow</td>
</tr>
<tr>
<td>31</td>
<td>Stripper steam valve</td>
</tr>
<tr>
<td>32</td>
<td>Reactor cooling water flow</td>
</tr>
<tr>
<td>33</td>
<td>Condenser cooling water flow</td>
</tr>
</tbody>
</table>
Reference list


