A Novel Dynamic PCA Algorithm for Dynamic Data Modeling and Process Monitoring

Yining Dong[a], S. Joe Qin[a,b,*]

[a] *Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA*
[b] *Mork Family Department of Chemical Engineering and Material Science, University of Southern California, Los Angeles, CA 90089, USA.*
School of Science and Engineering, The Chinese Unibersity of Hong Kong, Shenzhen, 2001 Longxiang Blvd., Longgang, Shenzhen, Guangdong, China

Abstract

Principal component analysis (PCA) has been widely applied for data modeling and process monitoring. However, it is not appropriate to directly apply PCA to data from a dynamic process, since PCA focuses on variance maximization only and pays no attention to whether the components contain dynamics or not. In this paper, a novel dynamic PCA (DiPCA) algorithm is proposed to extract explicitly a set of dynamic latent variables with which to capture the most dynamic variations in the data. After the dynamic variations are extracted, the residuals are essentially uncorrelated in time and static PCA can be applied. The new models generate a subspace of *principal time series* that are most predictable from their past data. Geometric properties are explored to give insight into the new dynamic model structure. For the purpose of process monitoring, fault detection indices based on DiPCA are developed based on the proposed model. Case studies on simulation data, data from an industrial boiler process, and the Tennessee Eastman process are presented to illustrate the effectiveness of the proposed dynamic models and fault detection methods.

Keywords: dynamic PCA, dynamic data modeling, process monitoring, fault detection

1. Introduction

Industrial process data are becoming massive and increasingly valuable assets for decision making in process operations, process control and monitoring. Since process measurements are often highly correlated, latent variable methods, such as principal component analysis (PCA) and partial least squares (PLS), are effective analytic tools for data modeling and process monitoring [**? ? ?**

*Corresponding author
Email address: sqin@usc.edu (S. Joe Qin)

? **?**]. In PCA, the objective is to extract latent variables from the data such that the variance of the extracted latent variables is maximized. By applying PCA, the measurement space can be decomposed into a principal subspace with maximized variability and a residual subspace. Fault detection statistics are developed for each subspace for process monitoring.

One major shortcoming for PCA is the lack of focus on time dependence, i.e., the structure of autocorrelation in the data is not exploited. However, measurements from industrial processes are often both cross-correlated and autocorrelated. Several problems can arise when applying static PCA to dynamic data directly. Since static PCA is unable to extract dynamic relationships from the data, autocorrelation and cross-correlation are mixed together, which makes it difficult for traditional PCA to reveal what type of relations among the measured variables. Furthermore, autocorrelations invalidate the statistical properties for fault detection methods developed for traditional PCA. Directly applying traditional fault detection methods may lead to misleading results.

Several methods have been developed to deal with measurements from dynamic processes. Negiz et al.[**?**] proposed a time series based approach, where a time series model is identified first. Then, the time series model is used to make predictions, and the prediction errors are used for process monitoring. However, this approach considers univariate time series models for simplicity. Even though it can be extended to multivariate time series, it does not retain the dimension reduction feature that is available in PCA to deal with cross-correlations.

A straightforward dynamic PCA method was proposed by Ku et al., which augments the data matrix with a number of lagged measurements [**?**]. Static PCA is performed on the augmented data matrix to extract latent variables. However, this method has several disadvantages. First, both the dimension of the loading vectors and the number of parameters increase dramatically as the number of lags increases. Second, no explicit representation can be derived between the latent variable and the original measurement variables. Third, the extracted latent variables have no attention on the dynamic content of the data; only variance is focused on, which makes it difficult to interpret the model and explore the underlying correlation structure. Alternative dynamic PCA algorithms were proposed to deal with various cases, such as nonlinearity and batch processes [**?** **?** **?**]. However, augmented data matrices are used in all of these algorithms. Therefore, they share the same limitations as Ku et al. [**?**].

Recently, Li et al. proposed a dynamic latent variable (DLV) modeling algorithm [**?**]. It first uses an auto-regressive PCA algorithm to extract the latent variables so that the maximum auto-covariance is achieved. Then, a vector autoregressive (VAR) model, which is referred as an *inner model*, is built for the latent variables to represent the dynamic relationships. This algorithm provides an explicit representation of the inner dynamic relationships, and a compact model to represent the data structure. However, when extracting the latent variables, only the auto-covariance at one time lag is maximized in this method; the auto-covariance at other time lags are ignored. This makes the extraction of latent variables inconsistent with the dynamic modeling of latent

variables.

A structured dynamic PCA algorithm was proposed by Li et al. as an improvement, where the objective function is to maximize the variance of a weighted sum of lagged latent variables [**?**]. Then a VAR model is built to capture the dynamic relationships of the latent variables. However, the VAR model of the inner dynamic relationships is not consistent with the maximum variance in the objective function. Collinear static relationships can dominate a latent variable that satisfies the maximum-variance, which fails to extract dynamic relationships in the latent variables.

Inspired by dynamic-inner partial least squares (DiPLS) [**?**] developed recently, a dynamic-inner principal component analysis (DiPCA) is proposed in this paper. While DiPLS is suitable for dynamic data modeling between two sets of variables, DiPCA extracts one or more latent variables that are linear combinations of one set of variables and have maximized auto-covariance. In the complement, the residuals after extracting the most predictable latent variables from the data will, in the limiting case, tend to be white noise. In this sense, the DiPCA algorithm extracts a set of principal time series from a multi-dimensional time series. The residual with little or no auto-covariance can then be treated as static data with traditional PCA methods for additional monitoring and analysis.

Although there are many versions of dynamic PCA algorithms in the literature, and some of them are used for process monitoring, most of them follow the approach by extending the data with time-lagged variables. These approaches immediately increase the dimensions of the data matrix, making them hardly a dimension reduction approach. Further, the static PCA objective in these methods does not even guarantee that the extracted principal components are dynamic. We listed other prior work in the following table for comparison, which were developed in our group as precursors to this current work. The interpretability of data analytic methods is always a desirable feature to possess. In this regard none of the existing methods can match the current work.

The remainder of this paper is organized as follows. Section 2 reviews the traditional PCA algorithm and gives the objective function of the DiPCA algorithm. Section 3 presents the proposed DiPCA algorithm. Geometric properties and relations of DiPCA are discussed in Section 4. Section 5 derives fault detection indices and proposes process monitoring schemes based on DiPCA model. Case studies on simulation data, boiler process data, and Tennessee Eastman Process data are presented to show the effectiveness of the proposed algorithm in Section 6. The final section gives conclusions.

2. PCA and DiPCA Objective Functions

Let $\mathbf{x} \in \mathbb{R}^m$ denote a sample vector of m variables. Assuming that there are n samples for each variable, a data matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, with each row representing a sample, can be used to build a PCA model. The objective of PCA is to extract a direction $\mathbf{p}$ of the largest variance in the m dimensional measurement space,

Table 1: Comparisons of DiPCA with other methods

Method	**Focusing on dynamics**	**Explicit dynamic components**	**Interpretability**	**Dimension reduction**
DiPCA (this work)	Yes	Yes	Easy	Yes, up to the dimension of the variable space
PCA with augmented lagged data [**?**]	No	No	Hardly any. There is no guarantee that the extracted principal components are even dynamic	No. The number of principal components can be greater than the dimension of the variable space
DLV [**?**]	No	Yes	Some, but static correlation can dominate a dynamic principal component	Yes, up to the dimension of the variable space
Structured DLV [**?**]	No	Yes	Some, but static correlation can dominate a dynamic principal component	Yes, up to the dimension of the variable space

which can be expressed as

$$\begin{aligned} \max_{\mathbf{p}} \quad & \mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p} \\ \text{s.t.} \quad & \|\mathbf{p}\| = 1 \end{aligned} \tag{1}$$

The solution to this optimization problem is obtained as follows using a Lagrange multiplier,

$$\mathbf{X}^T\mathbf{X}\mathbf{p} = \lambda\mathbf{p} \tag{2}$$

where $\lambda = \mathbf{p}^T\mathbf{X}^T\mathbf{X}\mathbf{p}$ matches the objective in (1). This implies that $\mathbf{p}$ is the eigenvector of $\mathbf{X}^T\mathbf{X}$ corresponding to the largest eigenvalue. After the loading vector $\mathbf{p}$ is obtained, the latent score vector $\mathbf{t}$ can be calculated as $\mathbf{t} = \mathbf{X}\mathbf{p}$. To extract the next principal component, the same optimization problem can be solved on the deflated matrix:

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{3}$$

It is clear from the objective of PCA that only static cross-correlations among the variables are extracted by PCA. If the data is from a dynamic process, traditional PCA will leave the dynamics unmodeled, making the principal components and even residuals having unmodeled dynamics. Having dynamic content in a component means that its future can be predictable from its past to a certain extent. The only uncertainty will be on the prediction error only, not the entire component. Therefore, ignoring dynamic information in the analysis

makes the model not representative of the data when they are used for further analysis and process monitoring. The consequence is that, while the false alarm rate can be controlled with a larger-than-necessary control limit, it leads to an overly large missed alarm rate. This restricts the applicability of PCA and makes it unsuitable for dynamic data modeling.

The objective of this paper is to develop a dynamic PCA objective that is consistent with its inner dynamic VAR relation, which is most predictable by its past. While this seems to be a natural and useful component to derive from the data, no prior work has been found in the literature. This approach is referred to as dynamic inner PCA in this paper. In general, the dynamics in latent variables that predict the current from the past can be expressed as

$$t_k = \beta_1 t_{k-1} + \cdots + \beta_s t_{k-s} + r_k \tag{4}$$

with the latent variable as a linear combination of the original variables

$$t_k = \mathbf{x}_k^T \mathbf{w} \tag{5}$$

where $\|\boldsymbol{\beta}\| = 1, \|\mathbf{w}\| = 1$, $\mathbf{x}_k$ is the sample vector at time k, and $\mathbf{w}$ is the weight vector. If the number of time lags, s, is chosen long enough such that the residual r_k is essentially white noise for each latent variable, the prediction from the dynamic inner model is

$$\begin{aligned} \hat{t}_k &= \mathbf{x}_{k-1}^T \mathbf{w}\beta_1 + \cdots + \mathbf{x}_{k-s}^T \mathbf{w}\beta_s \\ &= [\mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta} \otimes \mathbf{w}) \end{aligned} \tag{6}$$

where $\boldsymbol{\beta} = [\beta_1 \quad \beta_2 \cdots \beta_s]^T$ and $\boldsymbol{\beta} \otimes \mathbf{w}$ is the Kronecker product. Based on the above dynamic inner model, the objective function for extracting the latent variables should maximize the covariance between t_k and $\hat{t}_k$. Let $\mathbf{x}_k$ be the sample vector at time $k, k = 1, 2, ..., N + s$. Therefore, the objective is to maximize

$$\frac{1}{N} \sum_{k=s+1}^{s+N} \mathbf{w}^T \mathbf{x}_k [\mathbf{x}_{k-1}^T \cdots \mathbf{x}_{k-s}^T](\boldsymbol{\beta} \otimes \mathbf{w}) \tag{7}$$

This objective leads to the dynamic inner PCA algorithm to be derived in the next section.

3. Dynamic Inner PCA Algorithm

3.1. Extracting One Dynamic Component

Denote the data matrix as

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_{N+s}]^T$$

and form the following data matrices from $\mathbf{X}$,

$$\begin{aligned} &\mathbf{X}_i = [\mathbf{x}_i \quad \mathbf{x}_{i+1} \quad \cdots \quad \mathbf{x}_{N+i-1}]^T \quad \text{for} \quad i = 1, 2, \cdots, s+1 \\ &\mathbf{Z}_s = [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \cdots \quad \mathbf{X}_s] \end{aligned}$$

The objective in (7) can be rewritten as

$$\begin{aligned} \max_{\mathbf{w},\boldsymbol{\beta}} \quad & \mathbf{w}^T \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1, \|\boldsymbol{\beta}\| = 1 \end{aligned} \tag{8}$$

where s is the dynamic order of the model. The dimension of $\mathbf{w}$ is the same as the number of variables, which does not increase with the dynamic order of the model. After the weighting vector $\mathbf{w}$ is extracted, the latent score t_k is calculated as $t_k = \mathbf{x}_k^T \mathbf{w}$. It is clear that the most co-varying dynamic relationship is extracted between t_k and $t_{k-1}, \cdots, t_{k-s}$ by the objective function. Therefore, an explicit dynamic model is built between the latent variables. Compared to other dynamic PCA algorithms, the objective function of DiPCA leads to the extraction of only dynamic latent relations. The residuals after all dynamic components are extracted contain little dynamic information and, therefore, can be analyzed using static PCA further if so desired. Lagrange multipliers are used to solve the optimization in (8). Define

$$J = \mathbf{w}^T \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{w}) + \frac{1}{2} \lambda_\beta (1 - \boldsymbol{\beta}^T \boldsymbol{\beta}) + \lambda_w (1 - \mathbf{w}^T \mathbf{w}) \tag{9}$$

where

$$(\boldsymbol{\beta} \otimes \mathbf{w}) = (\boldsymbol{\beta} \otimes \mathbf{I})\mathbf{w} = (\mathbf{I} \otimes \mathbf{w})\boldsymbol{\beta}$$

Taking derivatives with respective to $\mathbf{w}$ and $\boldsymbol{\beta}$ and set them to zero, we have

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} &= \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{I})\mathbf{w} + (\boldsymbol{\beta} \otimes \mathbf{I})^T \mathbf{Z}_s^T \mathbf{X}_{s+1} \mathbf{w} \\ &\quad -2\lambda_w \mathbf{w} = 0 \qquad (10) \\ \frac{\partial J}{\partial \boldsymbol{\beta}} &= (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1} \mathbf{w} - \lambda_\beta \boldsymbol{\beta} = 0 \qquad (11) \end{aligned}$$

Define $\mathbf{G}_{\boldsymbol{\beta}} = \mathbf{X}_{s+1}^T \mathbf{Z}_s (\boldsymbol{\beta} \otimes \mathbf{I})$, the above equations can be simplified as

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{w}} &= \mathbf{G}_{\boldsymbol{\beta}} \mathbf{w} + \mathbf{G}_{\boldsymbol{\beta}}^T \mathbf{w} - 2\lambda_w \mathbf{w} = 0 \qquad (12) \\ \frac{\partial J}{\partial \boldsymbol{\beta}} &= (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1} \mathbf{w} - \lambda_\beta \boldsymbol{\beta} = 0 \qquad (13) \end{aligned}$$

Pre-multiplying $\mathbf{w}^T$ to (12) and $\boldsymbol{\beta}^T$ to (13) and using the fact that these vectors are unit norm, we have

$$\begin{aligned} 2\lambda_w &= \mathbf{w}^T (\mathbf{G}_{\boldsymbol{\beta}} + \mathbf{G}_{\boldsymbol{\beta}}^T) \mathbf{w} = 2J \\ \lambda_\beta &= \boldsymbol{\beta}^T (\mathbf{I} \otimes \mathbf{w})^T \mathbf{Z}_s^T \mathbf{X}_{s+1} \mathbf{w} = J \end{aligned} \tag{14}$$

The above results imply that λ_w and λ_β are both equal to the maximum value of J, and $\mathbf{w}$ is the eigenvector of $\mathbf{G}_{\boldsymbol{\beta}} + \mathbf{G}_{\boldsymbol{\beta}}^T$ corresponding to the largest eigenvalue.

However, since $\boldsymbol{\beta}$ and $\mathbf{w}$ are coupled together, there is no analytical solution to the optimization problem (8). Defining the latent scores $\mathbf{t}$ as follows,

$$\mathbf{t} = \mathbf{X}\mathbf{w}$$

and denoting

$$\mathbf{t}_i = \mathbf{X}_i\mathbf{w}, \quad \text{for } i = 1, 2, \cdots, s+1$$

as scores for the sub-matrices, (13) can be rewritten as follows,

$$\lambda_{\boldsymbol{\beta}}\boldsymbol{\beta} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T\mathbf{t}_{s+1} \tag{15}$$

which indicated that $\boldsymbol{\beta}$ depends on $\mathbf{w}$ implicitly through its dependence on $\mathbf{t}_i$, for $i = 1, 2, \cdots, s+1$, entirely. Furthermore, (12) can be re-organized as follows,

$$2\lambda_w\mathbf{w} = \sum_{i=1}^{s} \beta_i(\mathbf{X}_{s+1}^T\mathbf{t}_i + \mathbf{X}_i^T\mathbf{t}_{s+1}) \tag{16}$$

Therefore, the following iterative method is proposed to solve the problem.

(1) Initialize $\mathbf{w}$ with unit vector.
(2) Calculate $\mathbf{w}, \boldsymbol{\beta}$ by iterating the following relations until convergence.

$$\begin{aligned}
&\mathbf{t} = \mathbf{X}\mathbf{w} \ \text{ and form } \mathbf{t}_i \text{ from } \mathbf{t} \text{ for } i = 1, 2, \cdots, s+1\\
&\boldsymbol{\beta} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \cdots \quad \mathbf{t}_s]^T\mathbf{t}_{s+1}\\
&\mathbf{w} = \sum_{i=1}^{s} \beta_i(\mathbf{X}_{s+1}^T\mathbf{t}_i + \mathbf{X}_i^T\mathbf{t}_{s+1})\\
&\mathbf{w} := \mathbf{w}/\left\|\mathbf{w}\right\|\\
&\boldsymbol{\beta} := \boldsymbol{\beta}/\left\|\boldsymbol{\beta}\right\|
\end{aligned}$$

(3) Calculate $J = \sum_{i=1}^{s} \beta_i\mathbf{t}_i^T\mathbf{t}_{s+1}$.

To extract the next latent variable, the same iteration algorithm can be applied to the residual matrices of $\mathbf{X}_{s+1}$ and $\mathbf{Z}_s$. The deflation methods will be discussed in the next subsection.

3.2. Deflation

After the weight vector $\mathbf{w}$ and latent scores $\mathbf{t}$ are obtained from the above algorithm, $\mathbf{X}$ is deflated as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T \tag{17}$$

where the loading vector $\mathbf{p}$ is defined as

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t} \tag{18}$$

Form $\mathbf{X}_i$ from $\mathbf{X}$ for $i = 1, 2, \cdots, s+1$ and repeat the same algorithm to extract the next latent variable.

Remark 1. As in DiPLS[**?**], an alternative approach could be conceived, where the predicted $\hat{\mathbf{t}}_{s+1}$ is used to deflate $\mathbf{X}_{s+1}$ as

$$\begin{aligned}
&\mathbf{X}_{s+1} := \mathbf{X}_{s+1} - \hat{\mathbf{t}}_{s+1}\mathbf{q}^T \\
&\hat{\mathbf{t}}_{s+1} = \alpha_1\mathbf{t}_1 + \alpha_2\mathbf{t}_2 + \cdots + \alpha_s\mathbf{t}_s \\
&\mathbf{q} = \mathbf{X}_{s+1}^T\hat{\mathbf{t}}_{s+1}/\hat{\mathbf{t}}_{s+1}^T\hat{\mathbf{t}}_{s+1}
\end{aligned} \tag{19}$$

While the deflation of other matrices in $\mathbf{Z}_s$ remains the same as (17). However, this approach cannot be applied to DiPCA because we need to deflate both $\mathbf{Z}_s$ and $\mathbf{X}_{s+1}$ the same way to ensure that the next dynamic inner model is still an auto-regressive model. Therefore, it is imperative to deflate the way as shown in (17) to ensure that the subsequent latent variables are derived from consistently deflated residual matrices. The approach depicted in (19) no longer extracts the auto-regressive time series beyond the first latent time series.

3.3. DiPCA Model with l Components

The DiPCA main algorithm extracts the latent time series as the dynamic latent scores one by on in descending order of predicted covariation. These scores are orthogonal as will be shown later in this paper, which are very convenient for interpretation, but there can be some degree of correlation among them at different time lags. There is no reason to believe that these interactions are strong given that they are orthogonal without time lags, but one has the option to predict all latent scores simultaneously to account for these minor interactions. Let $\breve{\mathbf{t}}_j, j = 1, 2, \cdots, l$ denote the j^{th} latent component score vector and let $\mathbf{T} = [\breve{\mathbf{t}}_1 \;\; \breve{\mathbf{t}}_2 \;\; \cdots \;\; \breve{\mathbf{t}}_l]^T$. Form $\mathbf{T}_i$ from $\mathbf{T}$ for $i = 1, 2, \cdots, s+1$ in the same way as forming $\mathbf{X}_i$ from $\mathbf{X}$. A VAR model can be built to represent the dynamic relations between $\mathbf{T}_{s+1}$ and $\mathbf{T}_1, \mathbf{T}_2, \cdots, \mathbf{T}_s$ as follows.

$$\begin{aligned}
\mathbf{T}_{s+1} &= \mathbf{T}_1\boldsymbol{\Theta}_s + \mathbf{T}_2\boldsymbol{\Theta}_{s-1} + \cdots + \mathbf{T}_s\boldsymbol{\Theta}_1 + \mathbf{V} \\
&= \bar{\mathbf{T}}_s\boldsymbol{\Theta} + \mathbf{V}
\end{aligned} \tag{20}$$

where $\bar{\mathbf{T}}_s = [\mathbf{T}_1 \;\; \mathbf{T}_2 \;\; \cdots \;\; \mathbf{T}_s]$ and $\boldsymbol{\Theta} = [\boldsymbol{\Theta}_s \;\; \boldsymbol{\Theta}_{s-1} \;\; \cdots \;\; \boldsymbol{\Theta}_1]$. The least squares estimate for $\boldsymbol{\Theta}$ is

$$\hat{\boldsymbol{\Theta}} = (\bar{\mathbf{T}}_s^T\bar{\mathbf{T}}_s)^{-1}\bar{\mathbf{T}}_s^T\mathbf{T}_{s+1} \tag{21}$$

Once $\hat{\boldsymbol{\Theta}}$ is obtained, $\mathbf{T}_{s+1}$ can be predicted as

$$\hat{\mathbf{T}}_{s+1} = \bar{\mathbf{T}}_s\hat{\boldsymbol{\Theta}} \tag{22}$$

which can further be used to calculate the prediction of $\mathbf{X}_{s+1}$ as

$$\hat{\mathbf{X}}_{s+1} = \hat{\mathbf{T}}_{s+1}\mathbf{P}^T \tag{23}$$

where $\mathbf{P} = [\mathbf{p}_1 \;\; \mathbf{p}_2 \;\; \cdots \;\; \mathbf{p}_l]$ is the loading matrix with each $\mathbf{p}_i$ defined in (18).

With the prediction from past data, the prediction errors (PE) from DiPCA can be calculated as follows.

$$\mathbf{E}_{s+1} = \mathbf{X}_{s+1} - \hat{\mathbf{T}}_{s+1}\mathbf{P}^T \tag{24}$$

After the principal dynamic components are extracted, traditional PCA can be performed on the prediction error as follows,

$$\mathbf{E}_{s+1} = \mathbf{T}_r \mathbf{P}_r^T + \mathbf{E}_r \tag{25}$$

Therefore, $\mathbf{X}_{s+1}$ is decomposed as

$$\mathbf{X}_{s+1} = \hat{\mathbf{T}}_{s+1} \mathbf{P}^T + \mathbf{T}_r \mathbf{P}_r^T + \mathbf{E}_r \tag{26}$$

where the first term on the right hand side of (26) is a prediction using past data, while the other terms are projections of data involving the current data. The procedure of DiPCA modeling can be summarized in Table 2 as the DiPCA Algorithm.

Remark 2. It is possible that the iteration converges to a local maximum. This will lead to a smaller value of J. To avoid local maxima, initialize $\mathbf{w}$ randomly and perform multiple trials. The optimal $\mathbf{w}$ and $\boldsymbol{\beta}$ correspond to the largest value of J.

Remark 3. The inner model dynamics is parameterized with an autoregressive model. It is known in system identification that AR models can approximate more general dynamics such as autoregressive moving average (ARMA) dynamics. If it is desirable to parametrize the inner dynamics as an ARMA model, a subsequent model-reduction can be implemented to obtain an ARMA model or state space model [**?**]. If there exists integrating dynamics, it is likely to be extracted among the first few dynamic components. One has the option to model the integrating dynamics a priori or do it as part of DiPCA.

3.4. Determination of Model Parameters

In DiPCA modeling, three parameters need to be determined: dynamic order s, the number of dynamic latent variables l, and the number of static latent variables. First, assuming s is determined, then l can be chosen such that 95 percent of auto-covariance are captured by the first l dynamic latent variables. Therefore, l is viewed as a function of s, which can be written as $l = l(s)$. To determine the optimal s, a DiPCA model is built based on the training the data first. Then applying the model to the validation dataset allows us to obtain the prediction error matrix $\mathbf{E}_{s+1}^V$ of the validation data matrix. According to the previous analysis, little dynamic relationships are left in $\mathbf{E}_{s+1}^V$. Therefore, the sample crosscorrelation of any two variables in $\mathbf{E}_{s+1}^V$ should be close to 0, except when $lag = 0$. The calculation of the corresponding confidence bounds can be found in [**?**] . When all the pairs of variables are considered, the total violations of the confidence bounds can be obtained for any $(s, l(s))$. The parameter $(s, l(s))$ corresponding to the minimum violations is determined to be optimal. To determine the number of static components, cumulative percentage of variance (CPV) is applied [**?**].

Table 2: DiPCA Algorithm

1. Scale $\mathbf{X}$ to zero-mean and unit-variance. Initialize $\mathbf{w}$ to a random unit vector.
2. Extracting latent variables. Iterate the following relations until convergence.

   $\mathbf{t} = \mathbf{X}\mathbf{w}$ and form $\mathbf{t}_i$ from $\mathbf{t}$ similar to the formation of $\mathbf{X}_i$ for $i = 1, 2, \cdots, s+1$

   $\boldsymbol{\beta} = [\mathbf{t}_1 \;\; \mathbf{t}_2 \;\; \cdots \;\; \mathbf{t}_s]^T \mathbf{t}_{s+1}$

   $\mathbf{w} = \sum_{i=1}^{s} \beta_i (\mathbf{X}_{s+1}^T \mathbf{t}_i + \mathbf{X}_i^T \mathbf{t}_{s+1})$

   $\mathbf{w} := \mathbf{w} / \|\mathbf{w}\|$

   $\boldsymbol{\beta} := \boldsymbol{\beta} / \|\boldsymbol{\beta}\|$

3. Deflation. Deflate $\mathbf{X}$ as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T; \quad \mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}$$

4. Return to Step 2 to extract the next latent variable, until l latent variables are extracted.
5. Dynamic inner modeling. Build a VAR model for latent scores based on (21) and (22). Then, $\mathbf{T}_{s+1}$ is predicted as

$$\hat{\mathbf{T}}_{s+1} = \bar{\mathbf{T}}_s \hat{\boldsymbol{\Theta}}$$

6. Static modeling of prediction errors. Perform traditional PCA on the prediction error matrix $\mathbf{E}_{s+1}$

$$\mathbf{E}_{s+1} = \mathbf{X}_{s+1} - \hat{\mathbf{T}}_{s+1}\mathbf{P}^T = \mathbf{T}_r\mathbf{P}_r^T + \mathbf{E}_r$$

4. DiPCA Geometric Properties and Relations

4.1. DiPCA Geometric Properties

The DiPCA algorithm has a different structure from static PCA and other dynamic PCA algorithms. Therefore, it is important to understand the geometric properties of the DiPCA projections and how the data space is partitioned. To explore the DiPCA geometric properties with j latent variables being extracted, we use a subscript to denote the succession from one latent variable to the next as follows.

$$\mathbf{X}_{j+1} = \mathbf{X}_j - \breve{\mathbf{t}}_j \mathbf{p}_j^T \;\; \text{with} \;\; \mathbf{p}_j = \mathbf{X}_j^T \breve{\mathbf{t}}_j / \breve{\mathbf{t}}_j^T \breve{\mathbf{t}}_j \tag{27}$$

From (27) and the relations in DiPCA algorithm, we can obtain the following lemma.

Lemma 1. *Suppose $i < j$. We have the following relationships among the residual matrices and loading vectors.*

1. $\mathbf{X}_j = \mathbf{H}\mathbf{X}_{i+1}$
2. $\mathbf{X}_j = \mathbf{X}_{i+1}\mathbf{G}$
3. $\mathbf{X}_j\mathbf{w}_i = 0 \qquad i < j$
4. $\mathbf{X}_j^T\breve{\mathbf{t}}_i = 0 \qquad i < j$
5. $\mathbf{w}_i^T\mathbf{p}_i = 1$

where $\mathbf{H}$ *and* $\mathbf{G}$ *are some matrices formed by the DiPCA latent vectors and loadings and are given in the Appendix A.*

With the above Lemma, the following Theorem is given to summarize the orthogonal properties of the DiPCA latent scores and loadings.

Theorem 1. *In the DiPCA algorithm, denoting* $\bar{\mathbf{X}}_s = \sum_{d=1}^{s}\beta_d\mathbf{X}_d$*, then*

$$\mathbf{w} \propto [\mathbf{X}_{s+1}^T\bar{\mathbf{X}}_s + \bar{\mathbf{X}}_s^T\mathbf{X}_{s+1}]\mathbf{w}$$

which means that $\mathbf{w}$ *is the eigenvector of the above symmetric matrix corresponding to the largest eigenvalue. In addition, for Components* i, j *in the model, the following relations hold.*

1. $\mathbf{w}_i^T\mathbf{w}_j = 0 \qquad \forall i \neq j$
2. $\breve{\mathbf{t}}_i^T\breve{\mathbf{t}}_j = 0 \qquad \forall i \neq j$
3. $\mathbf{w}_i^T\mathbf{p}_j = 0 \qquad \forall i < j$

where $\breve{\mathbf{t}}_i$ and $\breve{\mathbf{t}}_j$ denote the score vectors of Components i and j, respectively, to be different from the subscripts that denote time-lagged vectors or matrices. The proof of above theorem can be found in Appendix B.

4.2. DiPCA Model Relations

Assuming the number of latent variables is chosen as l in the DiPCA model and collecting all latent score vectors and model vectors into the following matrices,

$$\begin{aligned}\mathbf{T} &= \left[\begin{array}{cccc}\breve{\mathbf{t}}_1 & \breve{\mathbf{t}}_2 & \cdots & \breve{\mathbf{t}}_l\end{array}\right] \\ \mathbf{W} &= \left[\begin{array}{cccc}\mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_l\end{array}\right] \\ \mathbf{P} &= \left[\begin{array}{cccc}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_l\end{array}\right]\end{aligned}$$

we have the following relation by iterating (27),

$$\mathbf{X}_{l+1} = \mathbf{X}_1 - \sum_{j=1}^{l}\breve{\mathbf{t}}_j\mathbf{p}_j^T = \mathbf{X} - \mathbf{T}\mathbf{P}^T \tag{28}$$

or

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{X}_{l+1} \tag{29}$$

In many cases, it is desirable to express the scores in terms of the original data for the convenience of model calculations. Post-multiplying (29) by $\mathbf{W}$ and using Relation 3. of Lemma 1, we can obtain

$$\begin{aligned} \mathbf{X}\mathbf{W} &= \mathbf{T}\mathbf{P}^T\mathbf{W} + \mathbf{X}_{l+1}\mathbf{W} \\ &= \mathbf{T}\mathbf{P}^T\mathbf{W} \end{aligned}$$

or

$$\mathbf{T} = \mathbf{X}\mathbf{R} \tag{30}$$

where

$$\mathbf{R} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1} \tag{31}$$

Pre-multiplying (31) by $\mathbf{P}^T$, we can obtain

$$\mathbf{P}^T\mathbf{R} = \mathbf{P}^T\mathbf{W}(\mathbf{P}^T\mathbf{W})^{-1} = \mathbf{I} \tag{32}$$

Therefore,

$$(\mathbf{P}\mathbf{R}^T)(\mathbf{P}\mathbf{R}^T) = \mathbf{P}\mathbf{R}^T \tag{33}$$

which is idempotent. Relation (31) and (33) are the same as the relations of $\mathbf{R}$ and $\mathbf{P}$ matrices of PLS given in [**?**], [**?**]. (29) can be rewritten as

$$\mathbf{X} = \mathbf{X}\mathbf{R}\mathbf{P}^T + \mathbf{X}_{l+1}$$

For a given vector $\mathbf{x}_k$ that would appear as a row in $\mathbf{X}$, the score vector is calculated from (30) as follows

$$\mathbf{t}_k = \mathbf{R}^T\mathbf{x}_k \tag{34}$$

and $\mathbf{x}_k$ can be decomposed as

$$\mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \tilde{\mathbf{x}}_k \tag{35}$$

The decomposition as (35) gives the partition of the space formed by current data. To explore how the past data are related to current data, (24) should be used to derive the sample relation for the time k as follows.

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{P}\hat{\mathbf{t}}_k$$

where $\mathbf{e}_k$ is the one step ahead prediction error and $\mathbf{t}_k$ can be predicted using the result in (22) as

$$\hat{\mathbf{t}}_k = \sum_{i=1}^{s} \mathbf{\Theta}_i^T \mathbf{t}_{k-i} = \mathbf{G}(q^{-1})\mathbf{t}_k$$

where $\mathbf{G}(q^{-1}) = \sum_{i=1}^{s} \mathbf{\Theta}_{s+1-i}^T q^{-i}$ is the transfer function of the dynamic model and q^{-1} is the backward shift operator. Therefore,

$$\mathbf{x}_k = \mathbf{P}\mathbf{G}(q^{-1})\mathbf{t}_k + \mathbf{e}_k \tag{36}$$

With the additional static PCA on $\mathbf{e}_k$, for a sample vector $\mathbf{x}_k$, we have

$$\mathbf{x}_k = \mathbf{P}\mathbf{G}(q^{-1})\mathbf{t}_k + \mathbf{P}_r\mathbf{t}_{k,r} + \mathbf{e}_{k,r} \tag{37}$$

We call $\mathbf{t}_k$ dynamic latent variables, and $\mathbf{t}_{k,r}$ static latent variables.

4.3. Dynamic Whitening Filter

DiPCA can also be viewed as a preprocessing filter that applied to $\mathbf{x}_k$ before PCA modeling. Since there are possible dynamic relationships in the data, directly applying traditional PCA to $\mathbf{x}_k$ is not appropriate. After DiPCA is first applied to the data $\mathbf{x}_k$, the prediction errors are essentially white, which makes them suitable to use PCA to model the latent structure in $\mathbf{e}_k$. Therefore, DiPCA can be interpreted as a whitening filter which removes all the dynamic relationships in data. Mathematically, based on (34) and (36), the dynamic whitening filter can be written as

$$\mathbf{e}_k = [\mathbf{I} - \mathbf{P}\mathbf{G}(q^{-1})\mathbf{R}^T]\mathbf{x}_k$$

The block diagram of DiPCA dynamic whitening filter is shown in Figure 1.

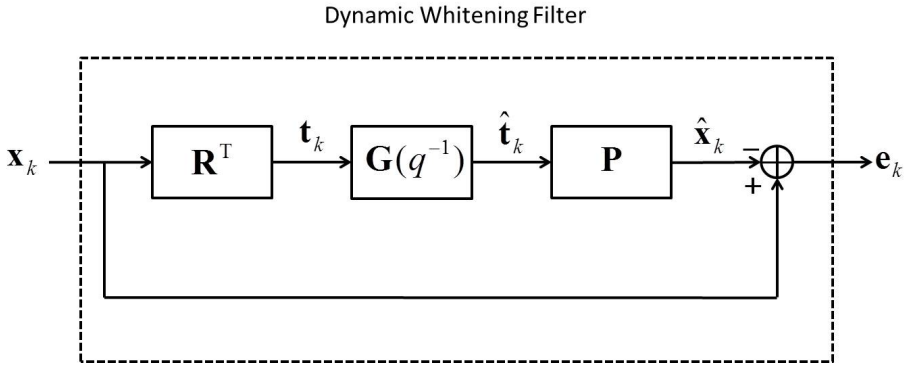

Figure 1: DiPCA dynamic whitening filter structure

5. Fault Detection Based on DiPCA

In process monitoring based on PCA, squared prediction error (SPE) and Hotelling's T^2 are typically used for detecting abnormal situations. SPE monitors the variations in the residual subspace, and Hotelling's T^2 monitors variations in the principal component subspace [**? ?**]. Process monitoring based on DiPCA can be divided into two parts: the monitoring of dynamic relationships and the monitoring of static relationships. Dynamic latent scores, dynamic residuals, static latent scores, and static residuals can be calculated respectively as follows.

$$\begin{aligned}
&\mathbf{t}_k = \mathbf{R}^T\mathbf{x}_k \\
&\mathbf{v}_k = \mathbf{t}_k - \hat{\mathbf{t}}_k; \quad \hat{\mathbf{t}}_k = \sum_{i=1}^{s} \mathbf{\Theta}_i^T \mathbf{t}_{k-i} \\
&\mathbf{e}_k = \mathbf{x}_k - \mathbf{P}\hat{\mathbf{t}}_k \\
&\mathbf{t}_{r,k} = \mathbf{P}_r^T\mathbf{e}_k \\
&\mathbf{e}_{r,k} = (\mathbf{I} - \mathbf{P}_r\mathbf{P}_r^T)\mathbf{e}_k
\end{aligned} \tag{38}$$

It is clear from (37) that process monitoring should be performed on $\hat{\mathbf{t}}_k$, $\mathbf{t}_{r,k}$ and $\mathbf{e}_{r,k}$ respectively. However, since $\hat{\mathbf{t}}_k$ is dynamic and could even be unstationary, monitoring $\hat{\mathbf{t}}_k$ can result in high false alarm rate. Therefore, the monitoring of $\hat{\mathbf{t}}_k$ can be performed through $\mathbf{v}_k$. Since $\mathbf{v}_k$ can be cross-correlated, it is appropriate to build another PCA model on $\mathbf{v}_k$ and construct a combined index to monitor $\mathbf{v}_k$ [**?**]. The combined index for $\mathbf{v}_k$ is defined as

$$\begin{aligned} \varphi_v &= \mathbf{v}_k^T \mathbf{\Phi}_v \mathbf{v}_k = \frac{T_v^2}{\chi_v^2} + \frac{Q_v}{\delta_v^2} \\ \mathbf{\Phi}_v &= \frac{\mathbf{P}_v \mathbf{\Lambda}_v^{-1} \mathbf{P}_v^T}{\chi_v^2} + \frac{\mathbf{I} - \mathbf{P}_v \mathbf{P}_v^T}{\delta_v^2} \end{aligned} \tag{39}$$

where $\mathbf{\Lambda}_v = \frac{1}{N-1}\mathbf{V}^T\mathbf{V}$, T_v^2 and Q_v are the fault detection indices for PCA based fault detection on $\mathbf{v}_k$, and χ_v^2 and δ_v^2 are the corresponding control limits as in [**?**]. In this way, the control region of the prediction $\hat{\mathbf{t}}_k$ can also be indicated by the control region of the combined index φ_v as Figure 2. In this figure, the

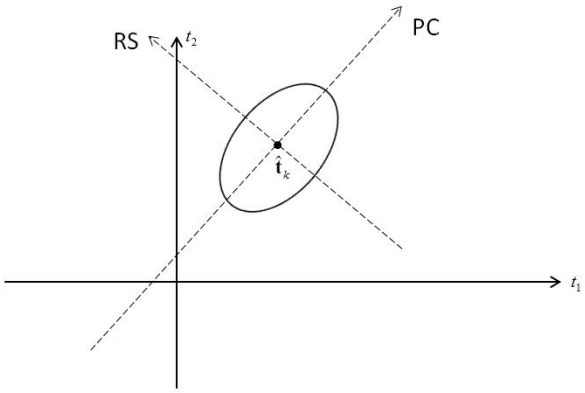

Figure 2: Confidence region of $\hat{\mathbf{t}}_k$ indicated by φ

elliptical area indicates the control region of the combined index φ_v, which is also the control region centered around the prediction $\hat{\mathbf{t}}_k$.

The monitoring of $\mathbf{e}_k$ can be performed as traditional PCA based process monitoring, where T_r^2 and Q_r are defined as follows,

$$\begin{aligned} T_r^2 &= \mathbf{t}_{k,r}^T \mathbf{\Lambda}_r^{-1} \mathbf{t}_{r,k} = \mathbf{e}_k^T \mathbf{P}_r \mathbf{\Lambda}_r^{-1} \mathbf{P}_r^T \mathbf{e}_k \\ Q_r &= \|\mathbf{e}_{r,k}\|^2 = \mathbf{e}_k^T (\mathbf{I} - \mathbf{P}_r \mathbf{P}_r^T) \mathbf{e}_k \end{aligned}$$

where $\mathbf{\Lambda}_r = \frac{1}{N-1}\mathbf{T}_r^T\mathbf{T}_r$. Control limits for T_r^2 and Q_r can be determined based on the results in [**?**], for instance.

6. Case Studies

Two case studies are presented in this section. The first case study is performed on simulation data, and the second case study is performed on Tennessee

Eastman Process (TEP) data. The effectiveness of the proposed DiPCA modeling and process monitoring algorithms are demonstrated in both cases.

6.1. Simulation Data

In this simulation, $\mathbf{t}_k$ is generated from a VAR(1) process, and $\mathbf{x}_k$ is generated from a latent variable model as

$$\begin{cases} \mathbf{t}_k = \mathbf{c} + \mathbf{A}\mathbf{t}_{k-1} + \mathbf{v}_k \\ \mathbf{x}_k = \mathbf{P}\mathbf{t}_k + \mathbf{e}_k \end{cases}$$

$$\mathbf{A} = \begin{pmatrix} 0.5205 & 0.1022 & 0.0599 \\ 0.5367 & -0.0139 & 0.4159 \\ 0.0412 & 0.6054 & 0.3874 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0.5205 \\ 0.5367 \\ 0.0412 \end{pmatrix}$$

$$\mathbf{P} = \begin{pmatrix} 0.4316 & 0.1723 & -0.0574 \\ 0.1202 & -0.1463 & 0.5348 \\ 0.2483 & 0.1982 & 0.4797 \\ 0.1151 & 0.1557 & 0.3739 \\ 0.2258 & 0.5461 & -0.0424 \end{pmatrix}$$

where $\mathbf{e}_k \in \mathbb{R}^5 \sim N([0, 0.1^2])$, and $\mathbf{v}_k \in \mathbb{R} \sim N([0, 1^2])$. 3000 data points are generated. First 1000 data points are used as a training dataset to train the DiPCA model. The next 1000 data points are used as a validation dataset to determine the dynamic order s and the number of dynamic latent variables l. The last 1000 data points are used as a testing dataset to evaluate the proposed fault detection indices.

By using the method described in Section III, $s = 1$ and $l = 3$ are selected. Figure 3 and Figure 4 show the autocorrelation and crosscorrelation of the original variables $\mathbf{x}_k$ and the dynamic latent variables $\mathbf{t}_k$ respectively. It is clear from both figures that strong dynamic relationships exist in $\mathbf{x}_k$ and $\mathbf{t}_k$. After DiPCA modeling, the autocorrelation and crosscorrelation of the prediction error $\mathbf{e}_k$ and the innovation $\mathbf{v}_k$ are plotted in Figure 5 and Figure 6. The results show that all the autocorrelation and crosscorrelation coefficients are close to zero except at lag 0. This indicates that dynamic relationships are removed from $\mathbf{x}_k$ and $\mathbf{t}_k$, and only static relationships remain in $\mathbf{e}_k$ and $\mathbf{v}_k$, which can be further modeled by PCA.

To test the performance of the fault detection indices, the following two types of faults are added to the test dataset. Since there is no residual subspace for $\mathbf{v}_k$, φ reduces to T_v^2.

1. $\mathbf{t}_k := \mathbf{t}_k + [0 \;\; 5 \;\; 0]^T, \quad k > 500.$
   In this case, the fault occurs in the dynamic components and is detected by φ and T_r^2 indices. However, since this fault does not affect the residual part of the static relationships, it is not detected Q_r index. This can be seen from Figure 7.

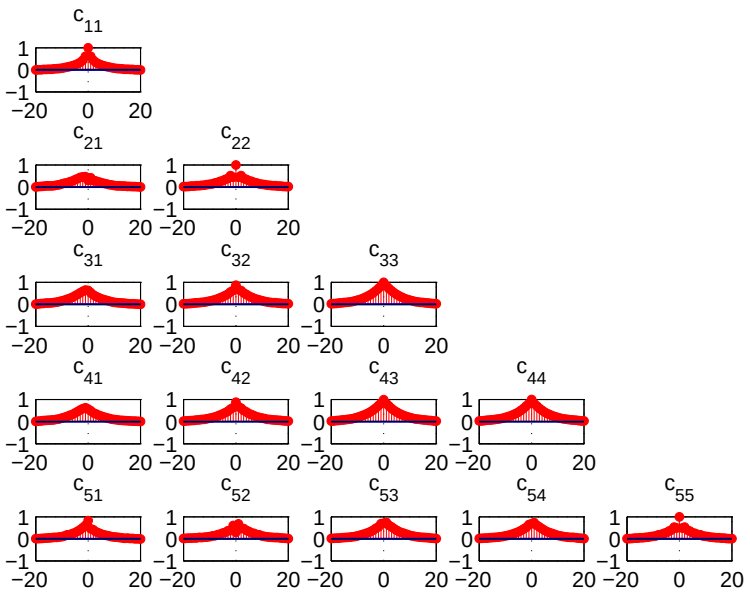

Figure 3: Autocorrelation and crosscorrelation of $\mathbf{x}_k$ of simulation data

2. When $k > 500$,

$$\mathbf{x}_k := \mathbf{x}_k + 5 * [0.0455 \quad 0.3877 \quad -0.8365 \quad 0.2911 \quad 0.2513]^T$$

In this case, the fault occurs in the residual part of the static relationships. Therefore, this fault is only detected by Q index, other indices remain unaffected as shown in Figure 8.

3. $\mathbf{A} := 1.2 * \mathbf{A}$, other matrices in the model remain the same.
In this case, the VAR model is unstable. 500 data points are used to train the DiPCA model, 100 data points are used as test dataset to check the false alarm rate of three fault detection indices. The results are shown in Figure 9. It is clear from the figure that T_d^2 has a high false alarm rate when the process is unstable. Therefore, it should not be used for fault detection purpose.

6.2. Data from an Industrial Boiler

Four sensors with 430 samples were collected from a boiler process under normal operating conditions. 330 samples are used to build the DiPCA model, and the remaining samples are used for testing. $s = 2$ and $l = 2$ are selected according to the method described in Section 3.4. In this example, there is

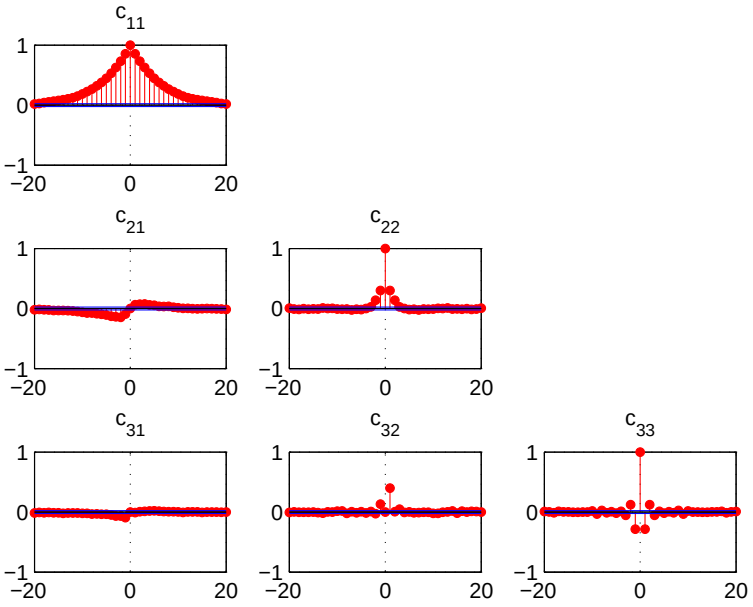

Figure 4: Autocorrelation and crosscorrelation of $\mathbf{t}_k$ of simulation data

no residual subspace for $\mathbf{v}_k$, therefore, the combined index φ_v reduces to T_v^2. Figure 10 shows the crosscorrelations of $\mathbf{e}_k$, from which we can see that negligible dynamics remain in the prediction error $\mathbf{e}_k$ after DiPCA modeling. Figure 11 and Figure 12 show the crosscorrelation of $\mathbf{t}_k$ and $\mathbf{v}_k$ respectively. It is obvious that strong dynamic relationships exist in $\mathbf{t}_k$, while only static relationships remain in $\mathbf{v}_k$ after a VAR model is built for $\mathbf{t}_k$. Figure 13 shows the control region for dynamic components at different time stamps. In Figure 13, the red points indicate $\hat{\mathbf{t}}_k$ and the black points indicate $\mathbf{t}_k$. The blue ellipses indicate the control region. In process monitoring, a sample is considered normal if $\mathbf{t}_k$ is inside the control region centered around $\hat{\mathbf{t}}_k$, and faulty if $\mathbf{t}_k$ is outside the control region. We can see from Figure 13 that all $\mathbf{t}_k$ are within the control region when no fault occurs. In addition, the control region varies with time. Contrast this with static PCA where the control region does not change over time.

6.3. TEP Data

The Tennessee Eastman Process was developed to provide a realistic simulation of an industrial process for the evaluation of monitoring methods [**?**]. The process contains 12 manipulated variables and 41 measured variables. The measured variables contain 22 process variables sampled every 3 minutes, and 19 quality variables sampled with dead time and time delays. In this case study,

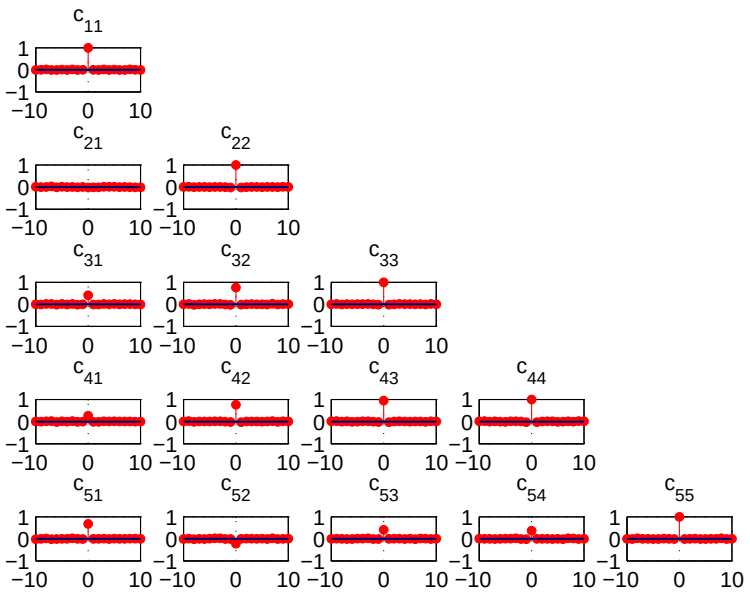

Figure 5: Autocorrelation and crosscorrelation of $\tilde{\mathbf{x}}_k$ of simulation data

22 process variables XMEAS(1-22) and 11 manipulated variables XMV(1-11) are collected as $\mathbf{X}$. 480 data points are used as training dataset. By using the parameter determination method described earlier, $s = 3$ and $l = 13$ are selected. Fig.14 shows the autocorrelation and crosscorrelation of the first four dynamic latent variables $\mathbf{t}_k$, and Figure 15 shows the autocorrelation and crosscorrelation of the innovations $\mathbf{v}_k$. From Figure 14, we can see that there are obvious autocorrelation and crosscorrelations in $\mathbf{t}_k$. After dynamic modeling, little dynamic crosscorrelations are left in $\mathbf{v}_k$, which can be seen from Figure 15.

The false alarm rates of φ_v, T_r^2 and Q_r indices are 5.54%, 6.58%, and 9.82% respectively. Table 3 shows the fault detection rates of three fault detection indices for 15 known faults in TEP. The fault detection results show that most of the faults can be detected successfully by the proposed fault detection indices, except for faults IDV(3), IDV(9) and IDV(15). The types of disturbances of these three faults are listed in Table 4. As discussed in [**?**], these three faults are hard to detect for all the popular process monitoring methods.

7. Conclusions

In this paper, a dynamic inner PCA algorithm is developed for dynamic data modeling. In the proposed method, a dynamic latent variable model is extracted

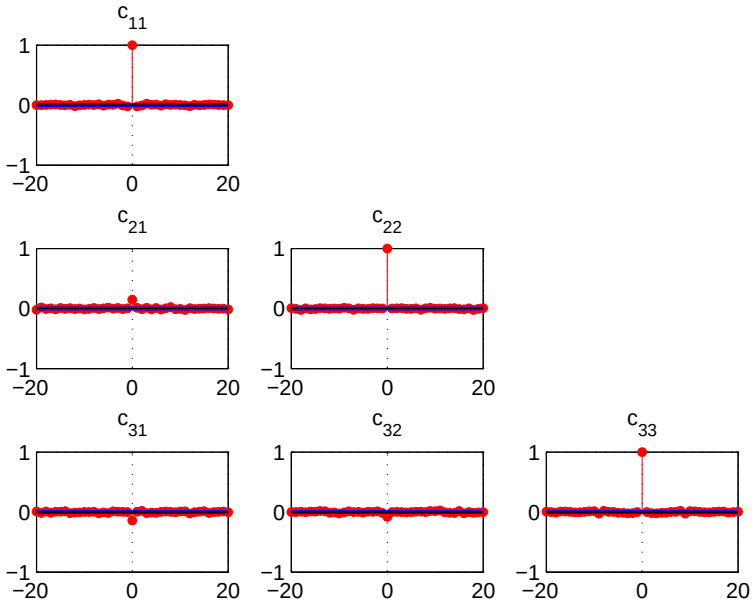

Figure 6: Autocorrelation and crosscorrelation of $\mathbf{v}_k$ of simulation data

first to capture the most auto-covarying dynamics in the data. The captured dynamic components contain the self-predictable variations of the data, while the residuals after extracting the dynamic components are essentially uncorrelated, least predictable and can be treated using static PCA. A cross-validation method is given to determine the order and the number of dynamic latent variables.

Geometric properties of DiPCA method are derived to give insight into the DiPCA model structure. The DiPCA objective guarantees that static variations in the data are left in the residuals after extracting dynamic components. In this case, the DiPCA model can be interpreted as a whitening filter with a dynamic latent structure, where the dimension of the dynamic latent space is generally smaller than the dimension of the original variable space. This feature is not available in other time series modeling or the Kalman filter approach.

For the purpose of process monitoring, three fault detection indices for the dynamic latent model and the subsequent static PCA model are derived. After filtering out the principal dynamics, the residual data are essentially white and therefore appropriate for PCA based monitoring. Since the whitening filter still leaves white noise in the residuals, the overall residual space after DiPCA filtering usually has the same dimension as the original measurement space, but with deflated variance-covariance. The extracted dynamic principal components provide clear dynamic visualization and monitoring. Case studies on simulation

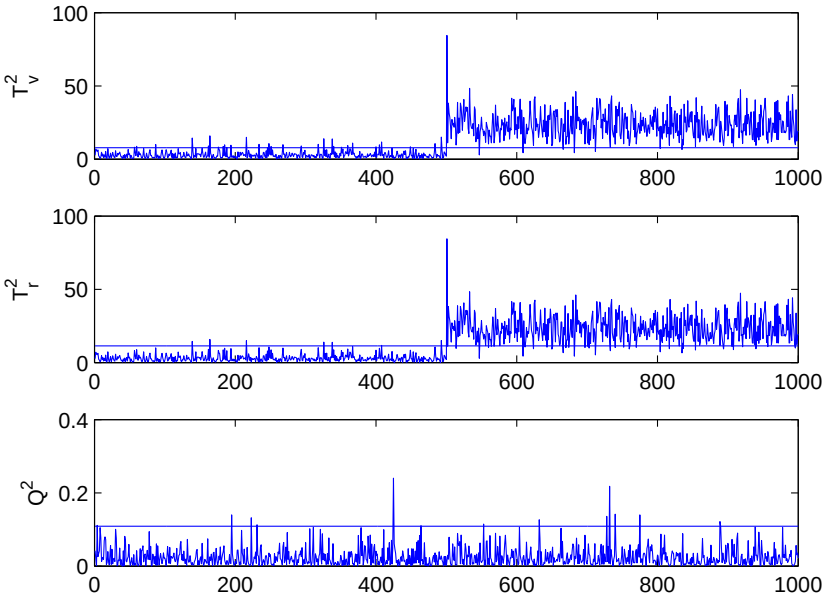

Figure 7: Fault detection results of fault 1 on simulation data

data, boiler process data, and the TEP data demonstrated that the proposed DiPCA modeling method provides a unique way to separate dynamic variations with latent variables from static variations. Further, fault detection indices are made reliable after separating the dynamic variations from static ones.

Acknowledgements

This work is supported by funds from the National Natural Science Foundation of China (61490704), The Fundamental Research Program of the Shenzhen Committee on Science and Innovations, and the Texas-Wisconsin-California Control Consortium.

Appendix A. Proof of Lemma 1

1. From (27) we have

$$\begin{aligned} \mathbf{X}_j &= \mathbf{X}_{j-1} - \breve{\mathbf{t}}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1} - \breve{\mathbf{t}}_{j-1}\breve{\mathbf{t}}_{j-1}^T\mathbf{X}_{j-1}/\breve{\mathbf{t}}_{j-1}^T\breve{\mathbf{t}}_{j-1} \\ &= (\mathbf{I} - \breve{\mathbf{t}}_{j-1}\breve{\mathbf{t}}_{j-1}^T/\breve{\mathbf{t}}_{j-1}^T\breve{\mathbf{t}}_{j-1})\mathbf{X}_{j-1} \end{aligned} \tag{A.1}$$

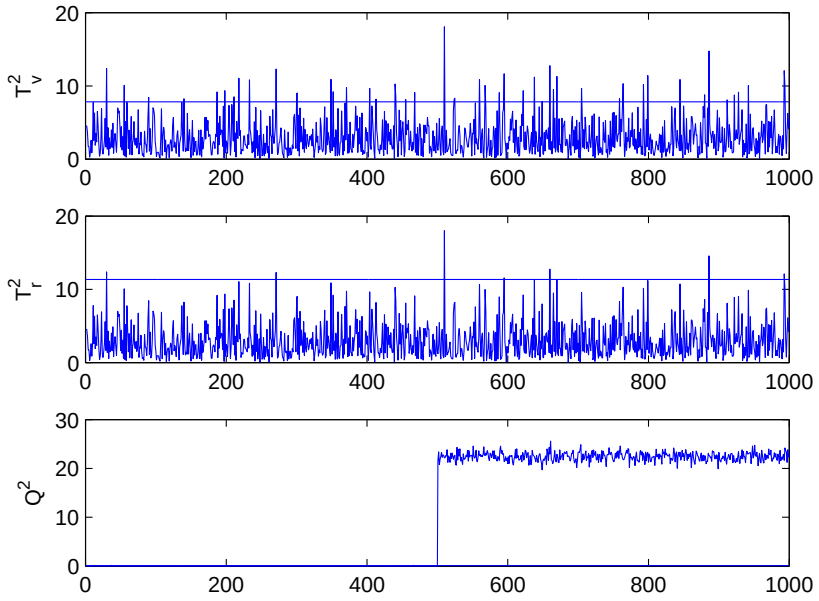

Figure 8: Fault detection results of fault 2 on simulation data

It is clear that this is a recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Relation 1. in Lemma 1 will be resulted with

$$\mathbf{H} = (\mathbf{I} - \breve{\mathbf{t}}_{j-1}\breve{\mathbf{t}}_{j-1}^T/\breve{\mathbf{t}}_{j-1}^T\breve{\mathbf{t}}_{j-1}) \cdots (\mathbf{I} - \breve{\mathbf{t}}_{i+1}\breve{\mathbf{t}}_{i+1}^T/\breve{\mathbf{t}}_{i+1}^T\breve{\mathbf{t}}_{i+1}) \tag{A.2}$$

2. In (A.1) if $\breve{\mathbf{t}}_j$ is replaced by $\breve{\mathbf{t}}_j = \mathbf{X}_j\mathbf{w}_j$, we have

$$\begin{aligned} \mathbf{X}_j &= \mathbf{X}_{j-1} - \breve{\mathbf{t}}_{j-1}\mathbf{p}_{j-1}^T = \mathbf{X}_{j-1} - \mathbf{X}_{j-1}\mathbf{w}_{j-1}\mathbf{p}_{j-1}^T \\ &= \mathbf{X}_{j-1}(\mathbf{I} - \mathbf{w}_{j-1}\mathbf{p}_{j-1}^T) \end{aligned} \tag{A.3}$$

Again, this is another recursive relation for $\mathbf{X}_j$. Iterating this relation until $\mathbf{X}_{i+1}$ appears, Relation 2. of Lemma 1 will be resulted.

3. From Relation 1. in Lemma 1 and (A.1) we have

$$\begin{aligned} \mathbf{X}_j\mathbf{w}_i &= \mathbf{H}\mathbf{X}_{i+1}\mathbf{w}_i = \mathbf{H}(\mathbf{I} - \breve{\mathbf{t}}_i\breve{\mathbf{t}}_i^T/\breve{\mathbf{t}}_i^T\breve{\mathbf{t}}_i)\mathbf{X}_i\mathbf{w}_i \\ &= \mathbf{H}(\mathbf{I} - \breve{\mathbf{t}}_i\breve{\mathbf{t}}^T/\breve{\mathbf{t}}_i^T\breve{\mathbf{t}}_i)\breve{\mathbf{t}}_i = 0 \end{aligned} \tag{A.4}$$

which proves Relation 3. of the Lemma 1.

4. From Relation 2. and (A.1) again, we have

$$\breve{\mathbf{t}}_i^T\mathbf{X}_j = \breve{\mathbf{t}}_i^T\mathbf{X}_{i+1}\mathbf{G} = \breve{\mathbf{t}}_i^T(\mathbf{I} - \breve{\mathbf{t}}_i\breve{\mathbf{t}}_i^T/\breve{\mathbf{t}}_i^T\breve{\mathbf{t}}_i)\mathbf{X}_i\mathbf{G} = 0 \tag{A.5}$$

using the same reasoning.

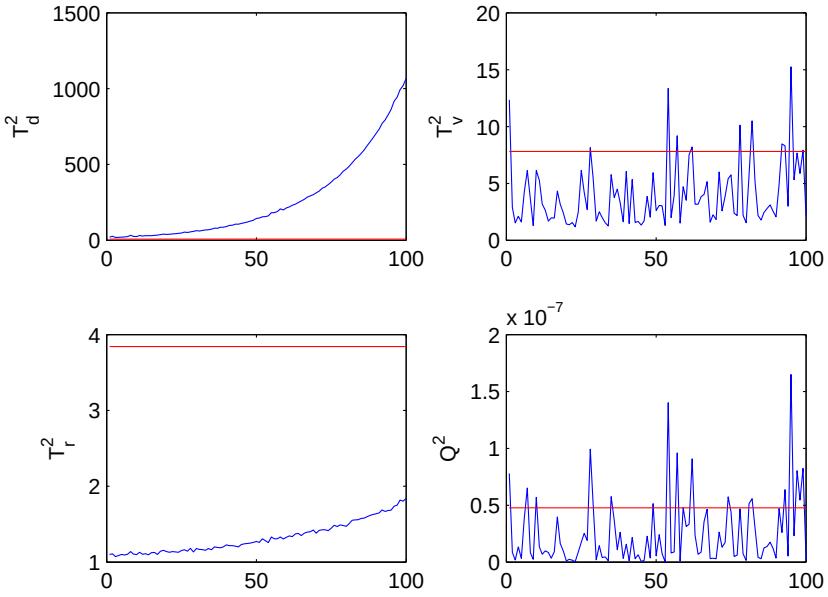

Figure 9: Fault detection results of faulty data with unstable dynamics

5. Using the expression for $\mathbf{p}_i$, Relation 5. of Lemma 1 can be easily shown as follows.

$$\mathbf{w}_i^T \mathbf{p}_i = \mathbf{w}_i^T \mathbf{X}_i^T \breve{\mathbf{t}}_i / \breve{\mathbf{t}}_i^T \breve{\mathbf{t}}_i = \breve{\mathbf{t}}_i^T \breve{\mathbf{t}}_i / \breve{\mathbf{t}}_i^T \breve{\mathbf{t}}_i = 1 \tag{A.6}$$

Appendix B. Proof of Theorem 1

To show relations 1. and 2. for $i \neq j$, it suffices to show them for the case of $i < j$, since the case of $i > j$ can be shown by symmetry. Using the expression for $\mathbf{w}_j$ we have,

$$\mathbf{w}_j = c \sum_{d=1}^{s} \beta_{d,j} (\mathbf{X}_{s+1,j}^T \breve{\mathbf{t}}_{d,j} + \mathbf{X}_{d,j}^T \breve{\mathbf{t}}_{s+1,j}) \tag{B.1}$$

where c is a proportional constant. Since $\mathbf{X}_{d,j}$ and $\mathbf{X}_{s+1,j}$ are sub-matrices of $\mathbf{X}_j$, $\mathbf{w}_i^T \mathbf{X}_j^T = 0$ implies $\mathbf{w}_i^T \mathbf{X}_{d,j}^T = 0$ and $\mathbf{w}_i^T \mathbf{X}_{s+1,j}^T = 0$, which further implies $\mathbf{w}_i^T \mathbf{w}_j = 0$. This is obvious from Relations 3. of Lemma 1. Furthermore, using Relation 4. of Lemma 1,

$$\breve{\mathbf{t}}_i^T \breve{\mathbf{t}}_j = \breve{\mathbf{t}}_i^T \mathbf{X}_j \mathbf{w}_j = 0 \tag{B.2}$$

Using Relation 3. of Lemma 1 again,

$$\mathbf{w}_i^T \mathbf{p}_j = \mathbf{w}_i^T \mathbf{X}_j^T \breve{\mathbf{t}}_j / \breve{\mathbf{t}}_j^T \breve{\mathbf{t}}_j = 0 \tag{B.3}$$

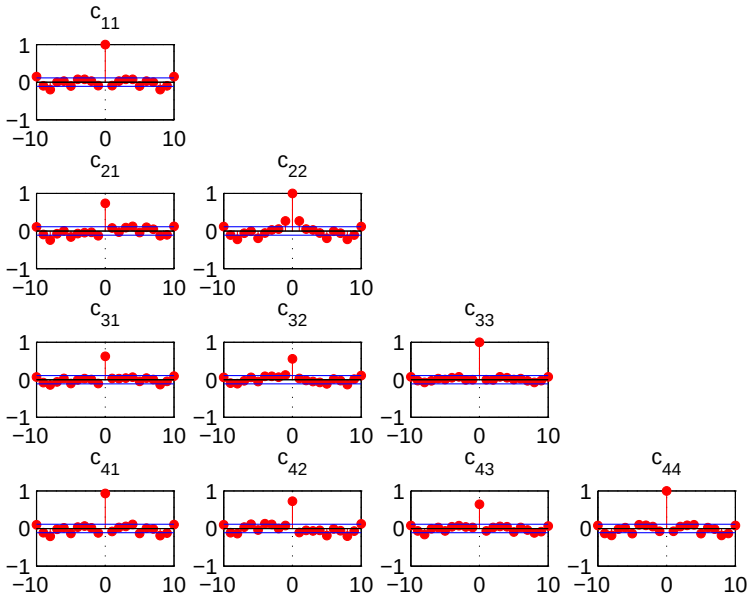

Figure 10: Crosscorrelation of $\mathbf{e}_k$ of data from an industrial boiler

References

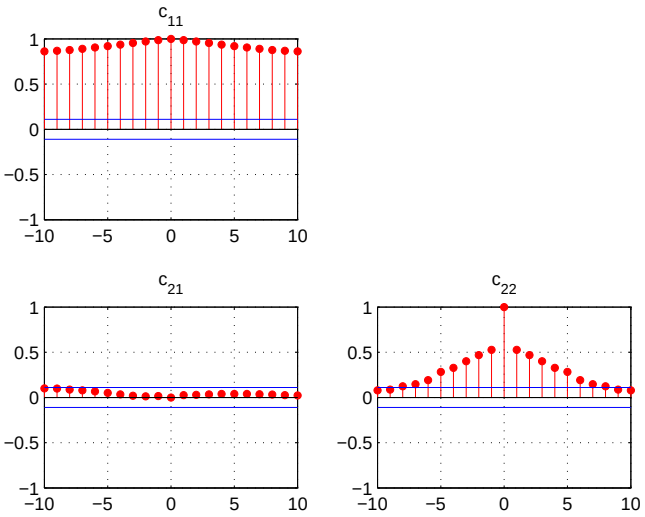

Figure 11: Crosscorrelation of $\mathbf{t}_k$ of data from an industrial boiler

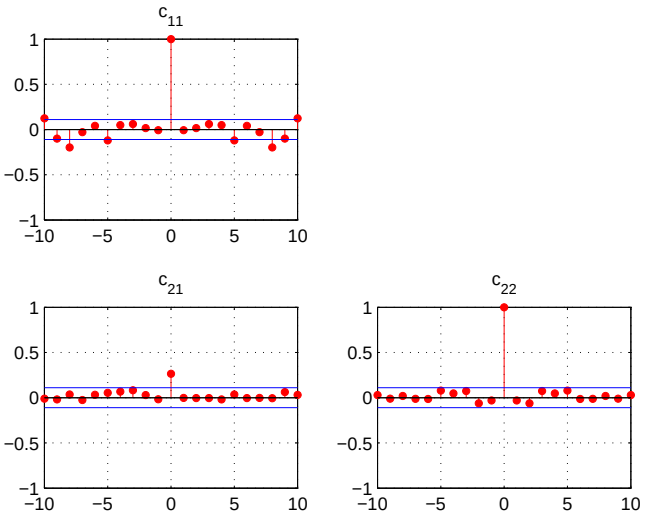

Figure 12: Crosscorrelation of $\mathbf{v}_k$ of data from an industrial boiler

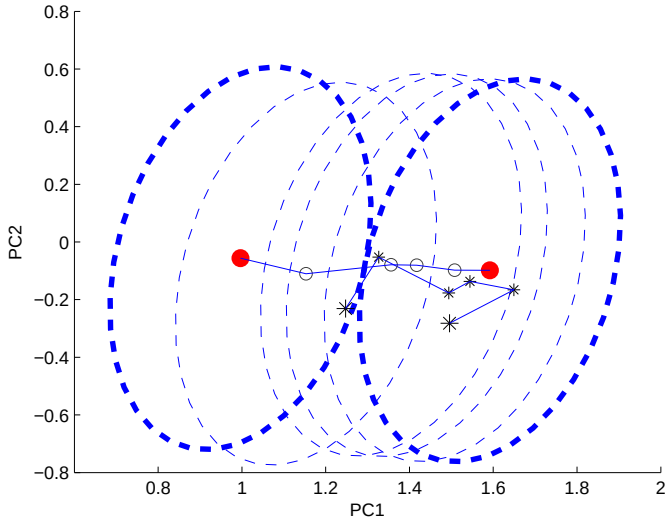

Figure 13: Control regions for dynamic components at different time stamps

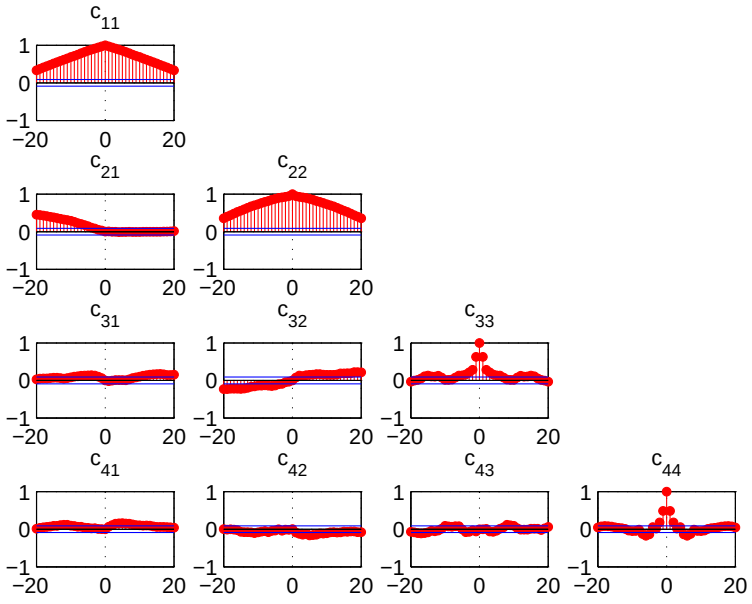

Figure 14: autocorrelation and crosscorrelation of $\mathbf{t}_k$ of TEP data

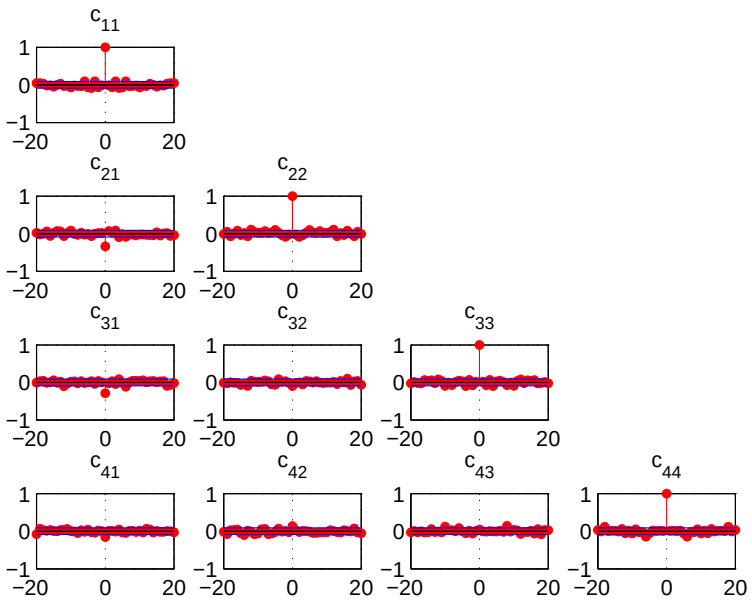

Figure 15: autocorrelation and crosscorrelation of $\mathbf{v}_k$ of TEP data

Table 3: Fault Detection Rate

Fault Type	φ_v	T_r^2	Q_r
$IDV(1)$	100	99.50	100
$IDV(2)$	99.00	98.62	97.87
$IDV(3)$	6.65	7.53	12.55
$IDV(4)$	97.49	100	27.73
$IDV(5)$	22.08	22.33	97.74
$IDV(6)$	100	99.37	100
$IDV(7)$	83.56	100	88.33
$IDV(8)$	95.86	94.10	95.98
$IDV(9)$	7.53	8.41	13.17
$IDV(10)$	15.18	13.93	77.16
$IDV(11)$	76.66	88.83	42.53
$IDV(12)$	95.23	95.61	99.00
$IDV(13)$	94.86	92.35	96.74
$IDV(14)$	100	100	100
$IDV(15)$	7.15	8.28	13.43

Table 4: Fault Description

Fault Type	Description
$IDV(3)$	D feed temperature (step)
$IDV(9)$	D feed temperature (random variation)
$IDV(15)$	Condenser cooling water valve (sticking)