Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# Adaptive multiblock kernel principal component analysis for monitoring complex industrial processes*#

Ying-wei ZHANG†, Yong-dong TENG

(*MOE Key Lab of Integrated Automation of Process Industry, Northeastern University, Shenyang 110004, China*)

†E-mail: zhangyingwei@mail.neu.edu.cn

Received Apr. 13, 2010; Revision accepted June 10, 2010; Crosschecked June 29, 2010

**Abstract:** Multiblock kernel principal component analysis (MBKPCA) has been proposed to isolate the faults and avoid the high computation cost. However, MBKPCA is not available for dynamic processes. To solve this problem, recursive MBKPCA is proposed for monitoring large scale processes. In this paper, we present a new recursive MBKPCA (RMBKPCA) algorithm, where the adaptive technique is adopted for dynamic characteristics. The proposed algorithm reduces the high computation cost, and is suitable for online model updating in the feature space. The proposed algorithm was applied to an industrial process for adaptive monitoring and found to efficiently capture the time-varying and nonlinear relationship in the process variables.

**Key words:** Recursive multiblock kernel principal component analysis (RMBPCA), Dynamic process, Nonlinear process
**doi:**10.1631/jzus.C1000148 **Document code:** A **CLC number:** TP27

## 1 Introduction

In many processes, large amounts of data are collected at different sampling intervals. Making better use of these data is a concern of industry. The multivariate statistic process control (MSPC) method has become a common and effective method for fault detection (Zhou *et al*., 2010). Because of the use of samples, principal component analysis (PCA) is the frequently used method. However, the precondition of PCA is that the process variable relationship is linear. This constraint is very tight (Jia *et al*., 2000). Thus, kernel PCA (KPCA) was proposed for nonlinear processes. KPCA (Kruger *et al*., 2007) uses the nonlinear kernel function to collect nonlinear principal components in a high-dimensional feature space; it deals with the nonlinear process better than PCA. Currently, the characteristics of most industrial processes are not only nonlinear, but also time varying, which are also called dynamic processes. The process data show that the dynamic process changes in variance and mean, and the relationship structure among variables. When the traditional KPCA method is used to monitor the dynamics, false alarms usually occur, which dramatically reduces the reliability of the static method. A recursive algorithm for PCA (RPCA), based on adaptive accommodation of the covariance, can better solve this problem (Gallagher *et al*., 1997). An iterative method may be used to enhance the performance of RPCA (Jeng *et al*., 2007). The recursive control limits were used to reduce false alarms and promote the reliability in kernel partial least squares (KPLS) algorithms (Voegtlin, 2005). Wang *et al*. (2003) used an adaptive procedure to accommodate the data feature matrix in a feature space. Recently, equipment for fault detection was tested under different conditions (Liu *et al*., 2009). Because of the static assumption of the process data, KPCA mentioned above may not deal with the process better. To overcome this deficiency, a statistical local method was used for KPCA (Maestri *et al*., 2009). The multivariate exponentially moving

* Project supported by the National Basic Research Program (973) of China (No. 2009CB320600) and the National Natural Science Foundation of China (No. 60974057)
# Presented at the 21st China Process Control Conference, August 6–8, 2010, Hangzhou, China

average (MEMA) was used to describe the changes of dynamic processes, and then to develop an adaptive monitoring statistic (Ge *et al*., 2009) when involved with KPCA components. Two RPCA algorithms were presented to significantly reduce the high cost (Cheng *et al*., 2010). Because the processes become more and more complex and the data amount larger and larger, the computing results can be difficult to understand. There have been many studies on the multiblock model (Qin *et al*., 2001; Elshenawy *et al*., 2010). An overview on multiblock algorithms has been published (Elshenawy *et al*., 2010). Multiblock approaches can reduce the computational load in a decentralized form (Qin *et al*., 2001; Elshenawy *et al*., 2010). In this paper, recursive KPCA (RKPCA) is brought into multiblock KPCA (MBKPCA) to form the recursive MBKPCA algorithm. A new recursive MBKPCA algorithm, RMBKPCA, is presented. The proposed algorithm reduces the high computation cost and is suitable for online model updating in the feature space.

## 2 Recursive kernel principal component analysis algorithm

### 2.1 Recursive kernel principal component analysis method

In this subsection, a new singular value decomposition technique is proposed, based on which a new RKPCA algorithm is proposed.

Set $\boldsymbol{X}=[\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N]$ as the sample matrix, whose singular value decomposition is $\boldsymbol{X}=\boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{D}^{\mathrm{T}}$. Set $\tilde{\boldsymbol{X}}=[\boldsymbol{x}_2,\boldsymbol{x}_3,\cdots,\boldsymbol{x}_N]\in\mathbb{R}^{m\times(N-1)}$ as the intermediate matrix, whose singular value decomposition is $\tilde{\boldsymbol{X}}=\tilde{\boldsymbol{S}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{D}}^{\mathrm{T}}$. $\tilde{\boldsymbol{X}}$ can be approximated by

$$\tilde{\boldsymbol{X}}\approx\tilde{\boldsymbol{X}}_m=\tilde{\boldsymbol{S}}_m\tilde{\boldsymbol{\Lambda}}_m\tilde{\boldsymbol{D}}_m{}^{\mathrm{T}}, \tag{1}$$

where $\tilde{\boldsymbol{S}}_m$ and $\tilde{\boldsymbol{D}}_m$ represent the front $m$ columns of $\tilde{\boldsymbol{S}}$ and $\tilde{\boldsymbol{D}}$, respectively, $\tilde{\boldsymbol{\Lambda}}_m=\mathrm{diag}\{\sqrt{\lambda_1},\sqrt{\lambda_2},\cdots,\sqrt{\lambda_m}\}$, $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_m>0$ represents the front $m$ eigenvalues of $\tilde{\boldsymbol{X}}^{\mathrm{T}}\tilde{\boldsymbol{X}}$. Thus, $\boldsymbol{X}$ can be represented by

$$\boldsymbol{X}=\begin{bmatrix}\boldsymbol{x}_1 & \tilde{\boldsymbol{X}}\end{bmatrix}=\begin{bmatrix}\boldsymbol{x}_1 & \tilde{\boldsymbol{S}}_m\end{bmatrix}\begin{bmatrix}\boldsymbol{1} & \boldsymbol{0}_m^{\mathrm{T}}\\ \boldsymbol{0}_m & \tilde{\boldsymbol{\Lambda}}_m\end{bmatrix}\begin{bmatrix}\boldsymbol{1} & \boldsymbol{0}_m^{\mathrm{T}}\\ \boldsymbol{0}_{N-1} & \tilde{\boldsymbol{D}}_m\end{bmatrix}^{\mathrm{T}}, \tag{2}$$

where $\boldsymbol{0}_m$ is an $m$-dimensional vector of zeros. Set $\boldsymbol{V}'=\begin{bmatrix}\boldsymbol{1} & \boldsymbol{0}_m^{\mathrm{T}}\\ \boldsymbol{0}_m & \tilde{\boldsymbol{\Lambda}}_m\end{bmatrix}$ as the eigenvectors of the sample matrix, which is block symmetric and whose singular value decomposition is computed as

$$\boldsymbol{V}'=\boldsymbol{S}'\boldsymbol{\Lambda}'\boldsymbol{D}'^{\mathrm{T}}. \tag{3}$$

The decomposition matrixes are block symmetric such that the decomposition becomes simple. Substituting Eq. (3) into Eq. (2) gives rise to

$$\begin{aligned}\boldsymbol{X}&=[\boldsymbol{x}_1\ \ \tilde{\boldsymbol{X}}]=[\boldsymbol{x}_1\ \ \tilde{\boldsymbol{S}}_m]\boldsymbol{S}'\boldsymbol{\Lambda}'\boldsymbol{D}'^{\mathrm{T}}\begin{bmatrix}\boldsymbol{1} & \boldsymbol{0}_m^{\mathrm{T}}\\ \boldsymbol{0}_{N-1} & \tilde{\boldsymbol{D}}_m\end{bmatrix}^{\mathrm{T}}\\ &=\boldsymbol{S}\boldsymbol{\Lambda}\boldsymbol{D}^{\mathrm{T}}.\end{aligned} \tag{4}$$

Then $\tilde{\boldsymbol{S}}_m$ can be computed according to Eq. (5):

$$\begin{bmatrix}\boldsymbol{x}_1 & \tilde{\boldsymbol{S}}_m\end{bmatrix}=\boldsymbol{S}(\boldsymbol{S}')^{-1}. \tag{5}$$

Given a new sample $\boldsymbol{x}_{\mathrm{new}}$, the singular value decomposition of the updated sample matrix $[\tilde{\boldsymbol{X}}_m\ \boldsymbol{x}_{\mathrm{new}}]$ can be represented by

$$\begin{bmatrix}\tilde{\boldsymbol{X}}_m & \boldsymbol{x}_{\mathrm{new}}\end{bmatrix}=\begin{bmatrix}\tilde{\boldsymbol{S}}_m & \boldsymbol{x}_{\mathrm{new}}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\Lambda}}_m & \boldsymbol{0}_m\\ \boldsymbol{0}_m^{\mathrm{T}} & \boldsymbol{1}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{D}}_m & \boldsymbol{0}_{N-1}\\ \boldsymbol{0}_m^{\mathrm{T}} & \boldsymbol{1}\end{bmatrix}^{\mathrm{T}}. \tag{6}$$

Set $\boldsymbol{V}''=\begin{bmatrix}\tilde{\boldsymbol{\Lambda}}_m & \boldsymbol{0}_m\\ \boldsymbol{0}_m^{\mathrm{T}} & \boldsymbol{1}\end{bmatrix}$ as the eigenvectors of the updated sample matrix, whose singular value decomposition is computed as

$$\boldsymbol{V}''=\boldsymbol{S}''\boldsymbol{\Lambda}''\boldsymbol{D}''^{\mathrm{T}}. \tag{7}$$

Substituting Eq. (7) into Eq. (6) gives rise to

$$\begin{bmatrix}\tilde{\boldsymbol{X}}_m & \boldsymbol{x}_{\mathrm{new}}\end{bmatrix}=\begin{bmatrix}\tilde{\boldsymbol{S}}_m & \boldsymbol{x}_{\mathrm{new}}\end{bmatrix}\boldsymbol{S}''\boldsymbol{\Lambda}''\boldsymbol{D}''^{\mathrm{T}}\begin{bmatrix}\tilde{\boldsymbol{D}}_m & \boldsymbol{0}_{N-1}\\ \boldsymbol{0}_m^{\mathrm{T}} & \boldsymbol{1}\end{bmatrix}^{\mathrm{T}}=\hat{\boldsymbol{S}}\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{D}}^{\mathrm{T}}, \tag{8}$$

where

$$\hat{\boldsymbol{S}}=\begin{bmatrix}\tilde{\boldsymbol{S}}_m & \boldsymbol{x}_{\mathrm{new}}\end{bmatrix}\boldsymbol{S}'', \tag{9}$$

$$\hat{\boldsymbol{D}}^{\mathrm{T}}=\boldsymbol{D}''^{\mathrm{T}}\begin{bmatrix}\tilde{\boldsymbol{D}}_m & \boldsymbol{0}_{N-1}\\ \boldsymbol{0}_m^{\mathrm{T}} & \boldsymbol{1}\end{bmatrix}^{\mathrm{T}}. \tag{10}$$

Set $\boldsymbol{X}_{\text{new}}=[\tilde{\boldsymbol{X}}_m \quad \boldsymbol{x}_{\text{new}}]$ as the new sample matrix, $\boldsymbol{S}=\hat{\boldsymbol{S}}$, $\boldsymbol{D}=\hat{\boldsymbol{D}}$. Using Eqs. (2), (4)–(6), and (8), the updating of eigenvalues and eigenvectors can be done for the new sample. When the recursive computing process is extended to the feature space $F$, a new recursive KPCA method is proposed here to update the eigenvalue decomposition of the covariance matrix $\boldsymbol{C}^F$. $\lambda_i$ and $\boldsymbol{p}_i$ represent the $i$th eigenvalue and eigenvector of $\boldsymbol{C}^F$, respectively. The radial basis function is applied to build the kernel matrix as follows in this work:

$$K_{i,j}=\exp\left(-\frac{1}{c}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2\right). \tag{11}$$

The centered Gram matrix is

$$\boldsymbol{K}=\bar{\boldsymbol{\Phi}}(\boldsymbol{X})^{\mathrm{T}}\bar{\boldsymbol{\Phi}}(\boldsymbol{X}). \tag{12}$$

$\boldsymbol{v}_i$ represents the $i$th eigenvector of $\boldsymbol{K}$. Thus,

$$\boldsymbol{p}_i=\bar{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{v}_i, \tag{13}$$

$$\begin{aligned}\boldsymbol{P}_m&=[\boldsymbol{p}_1,\boldsymbol{p}_2,\cdots,\boldsymbol{p}_m]\\&=\bar{\boldsymbol{\Phi}}(\boldsymbol{X})[\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_m]\\&=\bar{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}_m=\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{A}_{\boldsymbol{\Phi}},\end{aligned} \tag{14}$$

where $\boldsymbol{A}_{\boldsymbol{\Phi}}=\left(\boldsymbol{I}-\frac{1}{N}\boldsymbol{E}_N\right)\boldsymbol{V}_m$.

Set $\boldsymbol{\Phi}(\tilde{\boldsymbol{X}})=[\boldsymbol{\Phi}(\boldsymbol{x}_2),\boldsymbol{\Phi}(\boldsymbol{x}_3),\cdots,\boldsymbol{\Phi}(\boldsymbol{x}_N)]$. $\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})$ can be approximated by

$$\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\approx\bar{\boldsymbol{\Phi}}_m(\tilde{\boldsymbol{X}})=\tilde{\boldsymbol{S}}_{\boldsymbol{\Phi}_m}\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}^{\mathrm{T}}, \tag{15}$$

where $\tilde{\boldsymbol{S}}_{\boldsymbol{\Phi}_m}=\boldsymbol{\Phi}(\tilde{\boldsymbol{X}})\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}$, and $\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}$ denotes the $m$ dominant eigenvalues of the kernel matrix. According to Eqs. (2) and (15), we have

$$\begin{aligned}\bar{\boldsymbol{\Phi}}(\boldsymbol{X})&=[\boldsymbol{\Phi}(\boldsymbol{x}_1)-\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}\quad\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})]\approx[\boldsymbol{\Phi}(\boldsymbol{x}_1)-\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}\quad\bar{\boldsymbol{\Phi}}_m(\tilde{\boldsymbol{X}})]\\&=\left[\boldsymbol{\Phi}(\boldsymbol{x}_1)-\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}\quad\tilde{\boldsymbol{S}}_{\boldsymbol{\Phi}_m}\right]\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_m&\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}\end{bmatrix}\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_{N-1}&\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}\end{bmatrix}^{\mathrm{T}},\end{aligned} \tag{16}$$

where $\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}=\frac{1}{N-1}\boldsymbol{\Phi}(\tilde{\boldsymbol{X}})\boldsymbol{1}_{N-1}$. Set

$$\boldsymbol{V}'_{\boldsymbol{\Phi}}=\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_m&\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}\end{bmatrix},$$

whose singular value decomposition is computed as

$$\boldsymbol{V}'_{\boldsymbol{\Phi}}=\tilde{\boldsymbol{S}}'_{\boldsymbol{\Phi}}\boldsymbol{\Lambda}'_{\boldsymbol{\Phi}}\boldsymbol{D}'^{\mathrm{T}}_{\boldsymbol{\Phi}}. \tag{17}$$

Substituting Eq. (17) into Eq. (16) gives rise to

$$\begin{aligned}\bar{\boldsymbol{\Phi}}(\boldsymbol{X})&=\left[\boldsymbol{\Phi}(\boldsymbol{x}_1)-\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}\quad\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\right]\\&=\boldsymbol{\Phi}(\boldsymbol{X})\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\-\frac{1}{N-1}\boldsymbol{1}_{N-1}&\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}\end{bmatrix}\boldsymbol{S}'_{\boldsymbol{\Phi}}\boldsymbol{\Lambda}'_{\boldsymbol{\Phi}}\boldsymbol{D}'^{\mathrm{T}}_{\boldsymbol{\Phi}}\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_{N-1}&\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}\end{bmatrix}^{\mathrm{T}}.\end{aligned} \tag{18}$$

According to Eqs. (14), (15), and (18), $\bar{\boldsymbol{\Phi}}(\boldsymbol{X})$ can be represented by

$$\bar{\boldsymbol{\Phi}}(\boldsymbol{X})=\boldsymbol{S}_{\boldsymbol{\Phi}}\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}\boldsymbol{D}_{\boldsymbol{\Phi}}^{\mathrm{T}}=\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{A}_{\boldsymbol{\Phi}}\boldsymbol{\Lambda}'_{\boldsymbol{\Phi}}\boldsymbol{D}'^{\mathrm{T}}_{\boldsymbol{\Phi}}\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_{N-1}&\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}\end{bmatrix}^{\mathrm{T}}, \tag{19}$$

where

$$\boldsymbol{S}_{\boldsymbol{\Phi}}=\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{A}_{\boldsymbol{\Phi}},\ \boldsymbol{D}_{\boldsymbol{\Phi}}=\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\\boldsymbol{0}_{N-1}&\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}\end{bmatrix}\boldsymbol{D}'_{\boldsymbol{\Phi}}.$$

Because Eq. (18) is the same as Eq. (19), we have

$$\boldsymbol{A}_{\boldsymbol{\Phi}}=\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\-\frac{1}{N-1}\boldsymbol{1}_{N-1}&\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}\end{bmatrix}\boldsymbol{S}'_{\boldsymbol{\Phi}}. \tag{20}$$

Then $\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}$ can be computed according to Eq. (20):

$$\begin{bmatrix}1&\boldsymbol{0}_m^{\mathrm{T}}\\-\frac{1}{N-1}\boldsymbol{1}_{N-1}&\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}\end{bmatrix}=\boldsymbol{A}_{\boldsymbol{\Phi}}(\boldsymbol{S}'_{\boldsymbol{\Phi}})^{-1}. \tag{21}$$

Assume $\boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})$ is the new sample transformed into the feature space. $\boldsymbol{\Phi}(\boldsymbol{X}_{\text{new}})=\boldsymbol{\Phi}([\tilde{\boldsymbol{X}}\quad\boldsymbol{x}_{\text{new}}])$ is the updated sample matrix. Then the mean vector and the covariance matrix of $\boldsymbol{\Phi}(\boldsymbol{X}_{\text{new}})$ can be computed:

$$\begin{aligned}\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}&=\frac{1}{N}\boldsymbol{\Phi}([\tilde{\boldsymbol{X}}\quad\boldsymbol{x}_{\text{new}}])\boldsymbol{1}_N\\&=\frac{N-1}{N}\tilde{\boldsymbol{m}}_{\boldsymbol{\Phi}}+\frac{1}{N}\boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}}),\end{aligned} \tag{22}$$

$$\begin{aligned}\boldsymbol{C}_{\text{new}}^{F}&=\frac{1}{N-1}\bar{\boldsymbol{\Phi}}([\,\tilde{\boldsymbol{X}}\quad \boldsymbol{x}_{\text{new}}\,])\bar{\boldsymbol{\Phi}}([\,\tilde{\boldsymbol{X}}\quad \boldsymbol{x}_{\text{new}}\,])^{\mathrm{T}}\\&=\frac{1}{N-1}\left[\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\quad \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\\&\quad\cdot\left[\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\quad \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]^{\mathrm{T}}.\end{aligned}\tag{23}$$

According to Eqs. (6), (15), and (23), we have

$$\begin{aligned}\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})&=\left[\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\ \ \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\\&\approx\left[\bar{\boldsymbol{\Phi}}_{m}(\tilde{\boldsymbol{X}})\ \ \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\\&=\left[\tilde{\boldsymbol{S}}_{\boldsymbol{\Phi}_m}\ \ \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\begin{bmatrix}\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_m\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_{N-1}\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}^{\mathrm{T}}.\end{aligned}\tag{24}$$

Set $\boldsymbol{V}_{\boldsymbol{\Phi}}''=\begin{bmatrix}\tilde{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_m\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}$, whose singular value decomposition is computed as

$$\boldsymbol{V}_{\boldsymbol{\Phi}}''=\boldsymbol{S}_{\boldsymbol{\Phi}}''\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}''\boldsymbol{D}_{\boldsymbol{\Phi}}''^{\mathrm{T}}.\tag{25}$$

Substituting Eq. (25) into Eq. (24) gives rise to

$$\begin{aligned}\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})&=\left[\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}})\quad \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\\&\approx\left[\boldsymbol{\Phi}(\tilde{\boldsymbol{X}})\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}\quad \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\boldsymbol{S}_{\boldsymbol{\Phi}}''\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}''\boldsymbol{D}_{\boldsymbol{\Phi}}''^{\mathrm{T}}\begin{bmatrix}\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_{N-1}\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}^{\mathrm{T}}\\&=\boldsymbol{\Phi}([\,\tilde{\boldsymbol{X}}\quad \boldsymbol{x}_{\text{new}}\,])\begin{bmatrix}\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}&-\frac{\boldsymbol{1}_{N-1}}{N}\\\boldsymbol{0}_m^{\mathrm{T}}&\frac{N-1}{N}\end{bmatrix}\boldsymbol{S}_{\boldsymbol{\Phi}}''\boldsymbol{\Lambda}_{\boldsymbol{\Phi}}''\boldsymbol{D}_{\boldsymbol{\Phi}}''^{\mathrm{T}}\begin{bmatrix}\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_{N-1}\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}^{\mathrm{T}}\\&=\boldsymbol{\Phi}(\boldsymbol{X}_{\text{new}})\hat{\boldsymbol{A}}_{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}}\hat{\boldsymbol{D}}_{\boldsymbol{\Phi}}^{\mathrm{T}}=\hat{\boldsymbol{S}}_{\boldsymbol{\Phi}}\hat{\boldsymbol{\Lambda}}_{\boldsymbol{\Phi}}\hat{\boldsymbol{D}}_{\boldsymbol{\Phi}}^{\mathrm{T}},\end{aligned}\tag{26}$$

where

$$\hat{\boldsymbol{S}}_{\boldsymbol{\Phi}}=\boldsymbol{\Phi}(\boldsymbol{X}_{\text{new}})\hat{\boldsymbol{A}}_{\boldsymbol{\Phi}},\ \hat{\boldsymbol{D}}_{\boldsymbol{\Phi}}=\begin{bmatrix}\tilde{\boldsymbol{D}}_{\boldsymbol{\Phi}_m}&\boldsymbol{0}_{N-1}\\\boldsymbol{0}_m^{\mathrm{T}}&\boldsymbol{1}\end{bmatrix}\boldsymbol{D}_{\boldsymbol{\Phi}}''.$$

Then eigenvectors $\boldsymbol{P}_{\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})}$ of $\boldsymbol{C}_{\text{new}}^{F}$ are represented by

$$\boldsymbol{P}_{\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})}=\left[\boldsymbol{\Phi}(\tilde{\boldsymbol{X}})\tilde{\boldsymbol{A}}_{\boldsymbol{\Phi}_m}\quad \boldsymbol{\Phi}(\boldsymbol{x}_{\text{new}})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}\right]\boldsymbol{S}_{\boldsymbol{\Phi}}''.\tag{27}$$

The KPCA score vector $\boldsymbol{t}$, for a new sample $\boldsymbol{z}$, is given by

$$\boldsymbol{t}=\boldsymbol{P}_{\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})}^{\mathrm{T}}(\boldsymbol{\Phi}(\boldsymbol{z})-\hat{\boldsymbol{m}}_{\boldsymbol{\Phi}}).\tag{28}$$

Thus, fault diagnosis can be performed using the score vectors.

### 2.2 Fault diagnosis

The MBKPCA algorithm is introduced in the Appendix. Although SPE and $T^2$ plots generated by MBKPCA can be used to diagnose which part of the process is most related to the fault, one could not diagnose the fault with dynamic characteristics. In this subsection, the integration of RKPCA into MBKPCA forms an RMBKPCA algorithm:

1. Set $\boldsymbol{X}=[\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N]$ as the sample matrix. Scale the data matrix $\boldsymbol{X}$. Set $\tilde{\boldsymbol{X}}=[\boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_N]$ as the intermediate matrix. Given a new sample $\boldsymbol{x}_{\text{new}}$, the updated sample matrix is $\boldsymbol{X}_{\text{new}}=[\tilde{\boldsymbol{X}}\ \ \boldsymbol{x}_{\text{new}}]$.

2. Derive an MBKPCA model: $\boldsymbol{\Phi}(\boldsymbol{X}_b)\rightarrow\{\boldsymbol{t}_b, \boldsymbol{P}_b\}$, $b$=1, 2, …, $B$.

3. When $\boldsymbol{x}_{b,\text{new}}$ is available, scale it by the same way as in Step 1. Update $\{\boldsymbol{t}_b, \boldsymbol{P}_b\}$ using RKPCA.

4. Compute the decompositions of $\bar{\boldsymbol{\Phi}}(\tilde{\boldsymbol{X}}_b)$, $\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_b)$, and $\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{b,\text{new}})$, respectively.

5. Compute the eigenvectors $\boldsymbol{P}_{\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{\text{new}})}$ of $\bar{\boldsymbol{\Phi}}(\boldsymbol{X}_{b,\text{new}})$.

6. Determine the score vector $\boldsymbol{t}_{b,\text{new}}$.

In each recursive computation, the score vector can be obtained only by computing four parameters (Eqs. (4), (6), (12), and (13)). It is not necessary to compute all of the parameters. Hence, the high computation load is avoided.

The SPE and $T^2$ statistics can be computed according to Zhang and Qin (2010). For the new test sample $\boldsymbol{x}_{\text{new}}\in\mathbb{R}^N$, compute the block kernel vector by

$$\boldsymbol{k}_{b,\text{new}}=\boldsymbol{\Phi}(\boldsymbol{x}_{b,\text{new}})\boldsymbol{\Phi}(\boldsymbol{X}_b)^{\mathrm{T}}.\tag{29}$$

The super $T^2$ statistic can be calculated by

$$T_{\text{new}}^2=\boldsymbol{t}_{T,\text{new}}\boldsymbol{\Lambda}^{-1}\boldsymbol{t}_{T,\text{new}}^{\mathrm{T}},\tag{30}$$

where $\boldsymbol{\Lambda}^{-1}$ is the inverse of the covariance matrix of super scores. Block $T^2$ statistic is computed by

$$T_{b,\text{new}}^2=\boldsymbol{t}_{b,\text{new}}\boldsymbol{\Lambda}_b^{-1}\boldsymbol{t}_{b,\text{new}}^{\mathrm{T}},\tag{31}$$

where $\boldsymbol{\Lambda}_b^{-1}$ is the inverse of the covariance matrix of block scores.

The block SPE statistic can be calculated by

$$\mathrm{SPE}_b = 1 - 2\boldsymbol{k}_{b,\mathrm{new}}\boldsymbol{A}_b\boldsymbol{t}_{b,\mathrm{new}}^{\mathrm{T}} + \boldsymbol{t}_{b,\mathrm{new}}\boldsymbol{A}_b^{\mathrm{T}}\boldsymbol{K}_b\boldsymbol{A}_b\boldsymbol{t}_{b,\mathrm{new}}^{\mathrm{T}}, \quad (32)$$

where

$$\boldsymbol{A}_b=[\boldsymbol{A}_{b,1}, \boldsymbol{A}_{b,2}, \ldots, \boldsymbol{A}_{b,N}],\ \boldsymbol{A}_{b,i} = \boldsymbol{t}_{b,i} \Big/ \sqrt{\boldsymbol{t}_{b,i}(\boldsymbol{K}_b/N)\boldsymbol{t}_{b,i}^{\mathrm{T}}}.$$

Here the super $Q$ statistic is directly cumulated by

$$\mathrm{SPE} = \sum_{b=1}^{B}\mathrm{SPE}_b\ . \quad (33)$$

The $T^2$ statistic follows an $F$-distribution, and the confidence limit, $T_\beta^2$, is given by

$$T_\beta^2 = \frac{r(N^2-1)}{N(N-r)}F_{r,N-r,\beta}, \quad (34)$$

where $\beta$ is the confidence level. The SPE statistic can be approximated by a central $\chi^2$-distribution. The confidence limit for SPE, $\mathrm{SPE}_\beta$, can be approximated by

$$\mathrm{SPE}_\beta = g\chi^2(h), \quad g = \frac{\rho^2}{2\mu}, \quad h = \frac{2\mu^2}{\rho^2}, \quad (35)$$

where $\mu$ and $\rho^2$ are the mean and variance of the SPE statistic, respectively.

## 3 Results and discussion

### 3.1 Simulation example

MBKPCA and the proposed RMBKPCA methods were applied to an artificial example. The RMBKPCA monitoring approach was compared to MBKPCA using a fixed monitoring model. Details of this process are given, followed by a comparison of the application of RMBKPCA with MBKPCA using a fixed monitoring model.

There were five process variables in the simulated process example:

$$x_1 = v,\ x_2 = x_1^2 - 3ax_1,\ x_3 = -x_1^3 + 3ax_1^2,$$
$$x_4 = \mathrm{rand}(1, N),\quad x_5 = \sin x_4^2,$$

where $v$ is a uniformly distributed stochastic sequence within the range of [0.01, 2], and $a$ is a time-varying parameter that gradually increases from an initial value of 1 with a slope of 0.001 per sample, which consequently introduces the time-varying process behaviors for the process variables $x_2$ and $x_3$. The independent and identically distributed Gaussian sequences of zero mean and variance 0.01 were added to five process variables, which represent measurement uncertainty. From the above process, a dataset of 1500 samples was generated.

Based on the structure of this process, the variables were divided into two groups: $\boldsymbol{X}_1=[x_1\ x_2\ x_3]^{\mathrm{T}}$ and $\boldsymbol{X}_2=[x_4\ x_5]^{\mathrm{T}}$. Results using MBKPCA (Figs. 1 and 2) showed that both $T^2$ and SPE plots of the first block generated by KPCA exceeded the 99% confidence limit after 600 samples.

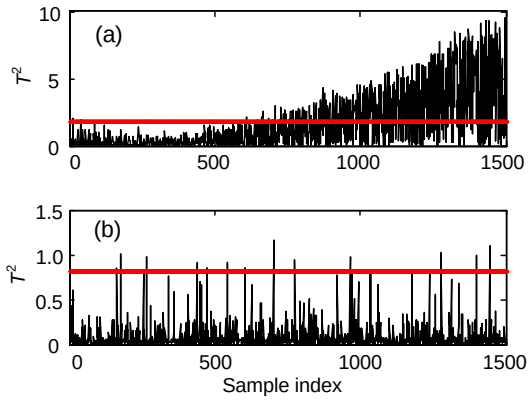


**Fig. 1 Block $T^2$ by multiblock kernel principal component analysis (MBKPCA)**
(a) Block 1; (b) Block 2

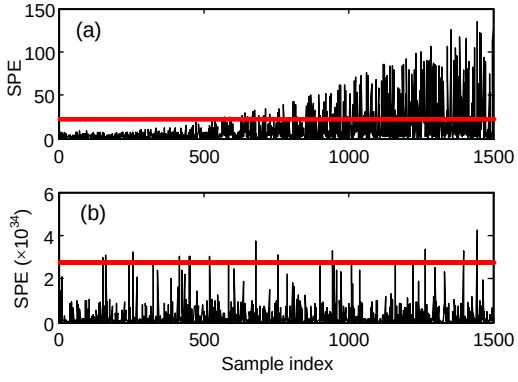


**Fig. 2 Block SPE by multiblock kernel principal component analysis (MBKPCA)**
(a) Block 1; (b) Block 2

Results using RMBKPCA (Figs. 3 and 4) confirmed that the proposed RMBKPCA monitoring approach did adapt to the impact of the time-varying process behavior, and that the process was in control. This simulation example has therefore illustrated that slowly developing time-varying behaviors can be accommodated by the RMBKPCA method.

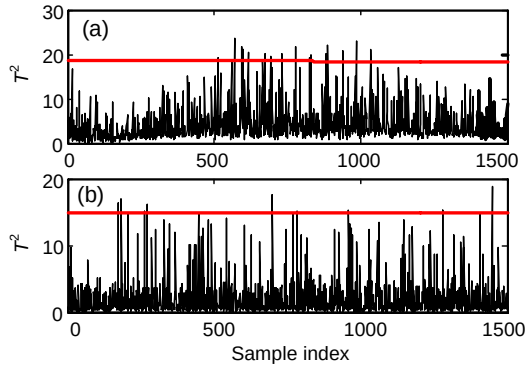


**Fig. 3 Block $T^2$ by recursive multiblock kernel principal component analysis (RMBKPCA)**
(a) Block 1; (b) Block 2

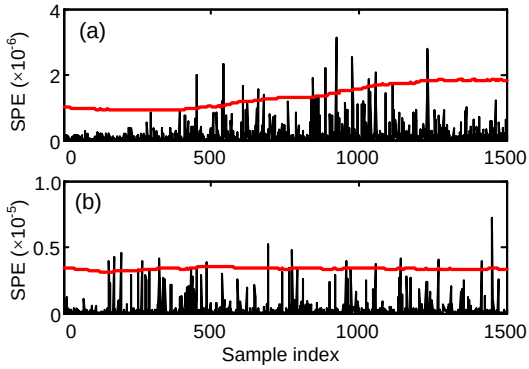


**Fig. 4 Block SPE by recursive multiblock kernel principal component analysis (RMBKPCA)**
(a) Block 1; (b) Block 2

### 3.2 Tennessee Eastman process

MBKPCA and the proposed method were applied to the Tennessee Eastman process, a complex nonlinear process created by Eastman Chemical Company as a realistic industrial process for evaluating the process control and monitoring methods, to verify the need for an adaptive monitoring scheme. The test process was based on a simulation of an actual industrial process, where the components, kinetics, and operating conditions have been modified for proprietary reasons. There are five major unit operations in the process: a reactor, a condenser, a recycle compressor, a separator, and a stripper. This process contains eight components: A, B, C, D, E, F, G, and H. The four reactants A, C, D, E and the inert B are fed to the reactor where the products G and H are formed, and a by-product F is also produced. The process has 22 continuous process measurements, 12 manipulated variables, and 19 composition measurements sampled less frequently. The details on the process description were well explained in Chiang *et al*. (2001). A total of 52 variables were used for monitoring in this study. A sampling interval of 3 min was used to collect the simulated data for the training and testing sets. The data can be downloaded from http://brahms.scs.uiuc.edu. Variables were divided into three groups: continuous process measurements, manipulated variables, and composition measurements.

Fault 6 was selected to test MBKPCA and RMBKPCA, using a dataset of 500 normal samples for training purposes. The test sample set contained 300 samples. The fault occurred at about the 160th sample. Building the model by using MBKPCA and RMBKPCA, the 99% control limits were calculated in each simulation. The MBKPCA model was used to monitor the process and not updated with new test data. The calculated block $T^2$ plots are shown in Fig. 5. In contrast, RMBKPCA used the training data to build the initial MBKPCA model, which was then updated using RKPCA. The monitoring results of RMBKPCA are shown in Fig. 6. Figs. 5 and 6 showed that both MBKPCA and RMBKPCA can detect the fault from about the 160th sample. However, because of the effect of the time-varying process condition, false alarms occurred in the MBKPCA process monitoring (Table 1). In contrast, RMBKPCA could efficiently capture the time-varying and nonlinear relationship in process variables. Thus, the false-alarm rate was reduced.

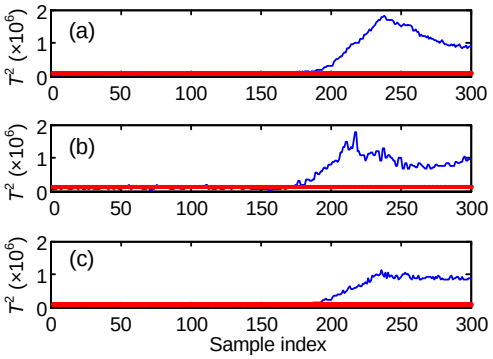


**Fig. 5 Block $T^2$ by multiblock kernel principal component analysis (MBKPCA)**
(a) Block 1; (b) Block 2; (c) Block 3

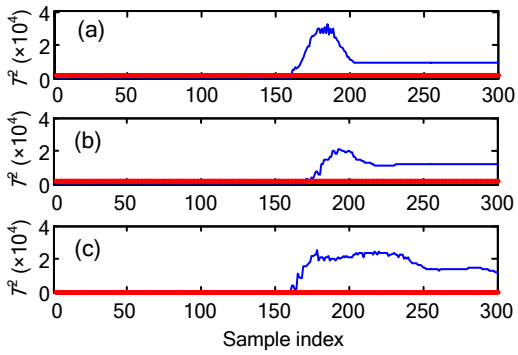


**Fig. 6 Block $T^2$ by recursive multiblock kernel principal component analysis (RMBKPCA)**
(a) Block 1; (b) Block 2; (c) Block 3

**Table 1 False-alarm points of multiblock kernel principal component analysis (MBKPCA) and recursive MBKPCA (RMBKPCA)**

| Scheme | Number of faulty samples | | |
|---|---|---|---|
| | Block 1 | Block 2 | Block 3 |
| MBKPCA | 2 | 16 | 13 |
| RMBKPCA | 0 | 0 | 1 |

## 4 Conclusions

In this paper, the RMBKPCA algorithm is proposed for time-varying processes. Compared to MBKPCA, the model built by RMBKPCA could be updated using the recursive eigenvalues and eigenvectors in the feature space. The proposed method was applied to the fault detection of an industrial process. The examples showed that the proposed method could efficiently capture the time-varying and nonlinear relationship in process variables.

## References

Cheng, C.Y., Hsu, C.C., Chen, M.C., 2010. Adaptive kernel principal component analysis (KPCA) for monitoring small disturbances of nonlinear processes. *Ind. Eng. Chem. Res.*, **49**(5):2254-2262. [doi:10.1021/ie900521b]

Chiang, L.H., Russell, F.L., Braatz, R.D., 2001. Fault Detection and Diagnosis in Industrial Systems. Springer, London.

Elshenawy, L.M., Yin, S., Naik, A.S., Ding, S.X., 2010. Efficient recursive principal component analysis algorithms for process monitoring. *Ind. Eng. Chem. Res.*, **49**(1):252-259. [doi:10.1021/ie900720w]

Gallagher, V.B., Wise, R.M., Butler, S.W., White, D.D., Barna, G.G., 1997. Development and Benchmarking of Multivariate Statistical Process Control Tools for a Semiconductor Etch Process: Improving Robustness Through Model Updating. Proc. ADCHEM, p.78-83.

Ge, Z., Yang, C., Song, Z., 2009. Improved kernel PCA-based monitoring approach for nonlinear processes. *Chem. Eng. Sci.*, **64**(9):2245-2255. [doi:10.1016/j.ces.2009.01.050]

Jeng, J.C., Li, C.C., Huang, H.P., 2007. Fault detection and isolation for dynamic processes using recursive principal component analysis (PCA) based on filtering of signals. *Asia-Pacific J. Chem. Eng.*, **2**(6):501-509. [doi:10.1002/apj.94]

Jia, F., Martin, E.B., Morris, A.J., 2000. Nonlinear principal components analysis with application to process fault detection. *Int. J. Syst. Sci.*, **31**(11):1473-1487. [doi:10.1080/00207720050197848]

Kruger, U., Zhang, J., Xie, L., 2007. Principal Manifolds for Data Visualization and Dimension Reduction. *In*: Gorban, A.N., Kégl, B., Wunsch, D.C., *et al.* (Eds.), Lecture Notes in Computational Science and Engineering, Vol. 58. Springer.

Liu, X., Kruger, U., Littler, T., Xie, L., Wang, S.Q., 2009. Moving window kernel PCA for adaptive monitoring of nonlinear processes. *Chemometr. Intell. Lab. Syst.*, **96**(2):132-143. [doi:10.1016/j.chemolab.2009.01.002]

Maestri, M.L., Cassanello, M.C., Hororwitz, G.I., 2009. Kernel PCA performance in processes with multiple operation modes. *Chem. Prod. Process Model.*, **4**(5), Article 7, p.1-6. [doi:10.2202/1934-2659.1383]

Qin, S.J., Valle, S., Piovoso, M.J., 2001. On unifying multiblock analysis with application to decentralized process monitoring. *J. Chemometr.*, **15**(9):715-742. [doi:10.1002/cem.667]

Voegtlin, T., 2005. Recursive principal components analysis. *Neur. Networks*, **18**(8):1051-1063. [doi:10.1016/j.neunet.2005.07.005]

Wang, X., Kruger, U., Lennox, B., 2003. Recursive partial least squares algorithms for monitoring complex industrial processes. *Control Eng. Pract.*, **11**(6):613-632. [doi:10.1016/S0967-0661(02)00096-5]

Zhang, Y., Qin, S.J., 2010. Decentralized fault diagnosis of large-scale processes using multiblock kernel principal component analysis. *Acta Autom. Sin.*, **36**(4):593-597. [doi:10.3724/SP.J.1004.2010.00593]

Zhou, D.H., Li, G., Qin, S.J., 2010. Total projection to latent structures for process monitoring. *AIChE J.*, **56**(1):168-178.

## Appendix: the multiblock kernel principal component analysis algorithm

MBKPCA is a development of MBPCA, which is an extension of the eigenvectors and eigenvalues solution of the kernel block matrix. Nonlinear iterative partial least squares (NIPALS) is applied for the calculation. Set $\boldsymbol{X}=[\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N]\in\mathbb{R}^{m\times N}$ as the sample matrix. For a new test sample $\boldsymbol{x}_{\text{new}}\in\mathbb{R}^m$, all

variables are blocked in the input space as $\boldsymbol{X}=[\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_B]$, and the kernel block matrix can be denoted as

$$\boldsymbol{K}_b = \boldsymbol{\Phi}(\boldsymbol{X}_b)\boldsymbol{\Phi}(\boldsymbol{X}_b)^{\mathrm{T}}, \tag{A1}$$

where $\boldsymbol{\Phi}(\boldsymbol{X}_b) \in \mathbb{R}^{m\times S_b}$. The $(i, j)$ element of $\boldsymbol{K}_b$ can be calculated by

$$K_{i,j}^{b} = \exp\left(-\frac{1}{c}\left\|\boldsymbol{X}_{b,i} - \boldsymbol{X}_{b,j}\right\|^2\right). \tag{A2}$$

In the original MBKPCA (Zhang and Qin, 2010), all the mixing score matrices obtained from different blocks are equally organized as the super scores of multiple blocks, which reveal the underlying information for process monitoring. Note that these block mixing relationships may be correlated with each other, and may also cover some relevant information more or less. The MBKPCA is introduced in Table A1.

**Table A1 Nonlinear iterative partial least squares (NIPALS) for the weighted multiblock kernel principal component analysis (MBKPCA) algorithm**

| Step | Description |
|---|---|
| 1 | Scale each block data to 0 means |
| 2 | Initialize $\boldsymbol{t}_{T,i}$ |
| 3 | For each block, compute<br>$\boldsymbol{p}_{b,i} = \boldsymbol{\Phi}(\boldsymbol{X}_{b,i})^{\mathrm{T}}\boldsymbol{t}_{T,i}\Big/\sqrt{\boldsymbol{t}_{T,i}^{\mathrm{T}}\boldsymbol{K}_{b,i}\boldsymbol{t}_{T,i}}$<br>$\boldsymbol{t}_{b,i} = \boldsymbol{\Phi}(\boldsymbol{X}_{b,i})\boldsymbol{p}_{b,i} = \boldsymbol{K}_{b,i}\boldsymbol{t}_{T,i}\Big/\sqrt{\boldsymbol{t}_{T,i}^{\mathrm{T}}\boldsymbol{K}_{b,i}\boldsymbol{t}_{T,i}}$<br>$\boldsymbol{T}_i = [\boldsymbol{t}_{1,i}, \boldsymbol{t}_{2,i}, \ldots, \boldsymbol{t}_{B,i}]$<br>$\boldsymbol{p}_{T,i} = \boldsymbol{T}_i^{\mathrm{T}}\boldsymbol{t}_{T,i}/\|\boldsymbol{T}_i^{\mathrm{T}}\boldsymbol{t}_{T,i}\|$<br>$\boldsymbol{t}_{T,i} = \boldsymbol{T}_i\boldsymbol{p}_{T,i}$ |
| 4 | If $\boldsymbol{t}_{T,i}$ is not converging, go to Step 3; else, go to Step 5 |
| 5 | For each block, deflate residual<br>$\boldsymbol{K}_{b,i+1} = [(\boldsymbol{I} - \boldsymbol{t}_{T,i}\boldsymbol{t}_{T,i}^{\mathrm{T}}/(\boldsymbol{t}_{T,i}^{\mathrm{T}}\boldsymbol{t}_{T,i}))]\boldsymbol{K}_{b,i}[\boldsymbol{I} - \boldsymbol{t}_{T,i}\boldsymbol{t}_{T,i}^{\mathrm{T}}/(\boldsymbol{t}_{T,i}^{\mathrm{T}}\boldsymbol{t}_{T,i})]$ |
| 6 | Go to Step 2 to obtain the next principal component |