12th IFAC International Symposium on
Advanced Control of Chemical Processes
July 14-17, 2024. Toronto, Canada

Recursive Dynamic inner Principal Component Analysis for Adaptive Process Modeling ⋆

Qilin Qu [*,**] **Yining Dong** [**] **Ying Zheng** [*]

[*] *School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: zyhidy@mail.hust.edu.cn)*
[**] *School of Data Science, City University of Hong Kong, Hong Kong China (e-mail: yining.dong@cityu.edu.hk)*

Abstract: Dynamic latent variable (DLV) methods, represented by dynamic-inner principal component analysis (DiPCA), take into account the high dimensionality and auto-correlation of industrial process data to successfully extract and model the dynamic components. Meanwhile, the time-varying dynamics involved in industrial processes motivate us to explore adaptive DLV methods. In this paper, we propose a recursive DiPCA (RDiPCA) for time-varying dynamic process modeling. Specifically, a recursive autocovariance matrices updating method and the corresponding deflation method are given to achieve low computational costs. The computational efficiency is further improved by a recursive parameter initialization approach in the iterative optimization algorithm solving procedure. Finally, the effectiveness of the proposed algorithm is demonstrated with experiments on a numerical dataset and a wastewater treatment plant dataset.

Keywords: Dynamic latent variable methods; Recursive dynamic-inner principal component analysis; Adaptive dynamic process modeling.

1. INTRODUCTION

Owing to the development of computer science and measurement technology, accessible industrial process data has been of great help in improving the safety and quality of production. In addition, monitoring and control based on big data analytics such as Qin et al. (2020) and Basanta-Val (2018) are becoming an important part of the industrial production process. At the same time, the large-scale, high-dimensional, dynamic, and time-varying industrial process data poses significant challenges to data-driven modeling and monitoring.

Principal component analysis (PCA) is one of the most widely used techniques in industrial process monitoring. It detects process anomalies by mapping high-dimensional cross-correlated data into principal component subspace and residual subspace and establishing separate or combined monitoring indices. To address the time-varying characteristic of processes, Li et al. (2000) proposed a recursive PCA, which transforms the eigenvalue computation into a one-rank modification problem by recursively updating the data covariance matrix. On top of that, Qin (1998) proposed a recursive PLS. These methods greatly reduce the computational cost and makes it possible to update the model online to monitor time-varying processes.

⋆ This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFA1003504; in part by the National Natural Science Foundation of China under Grant 22322816 and 61873102; in part by the CityU New Research Initiatives under Grant 9610640.

However, PCA assumes that process data is static and not auto-correlated. Practically, widespread control loops make this assumption difficult to justify. Therefore, some methods consider the dynamics of processes, such as DPCA by Ku et al. (1995). Further, attempting to address dynamic and time-varying characteristics simultaneously, Hu et al. (2012); Hajarian et al. (2020); Feng et al. (2022) extended the DPCA method into a recursive approach and also established the corresponding adaptive monitoring indices. Despite discussing on dynamics, these methods are generally based on DPCA, which does not have an explicit dynamic objective thus lacks the ability to separate dynamic components. Moreover, the consistency between the mean and variance of the lagged sample vectors is ignored, which increases the computational cost and makes the covariance matrix of the augmented sample vector not block Toeplitz.

To construct temporal monitoring statistics and provide more meaningful information, slow feature analysis (SFA) based modeling and monitoring was proposed by Guo et al. (2016). The idea is to isolate temporal dynamics from steady conditions of processes. Shang et al. (2018) developed a tailored recursive SFA (RSFA) algorithm and an adaptive monitoring method by considering the time-varying dynamic behavior of process variables. However, SFA focuses only on first order dynamics, which makes the dynamic information inadequate.

Dynamic latent variable (DLV) methods were proposed to extract the dynamic components. Li et al. (2011,

2014) proposed a DLV modeling algorithm and an improved structured dynamic PCA algorithm. The former method extracts latent variables by maximizing the auto-covariance between samples and their delayed augmentation, while the latter one maximizes the variance of the delayed samples. Then both methods establish a vector auto-regressive (VAR) model for latent variables to represent dynamic relationships. Further, Dong and Qin (2018); Dong et al. (2020) proposed DiPCA and DiCCA methods and designed a model structure that ensures the objective of the outer dynamic latent variable extraction model is consistent with the inner latent variable prediction model, making the extracted dynamic components interpretable.

Currently, to the best of our knowledge, no existing DLV methods can address the time-varying nature of processes. In this paper, we take the advantages of DiPCA to investigate adaptive modeling methods for dynamic time-varying processes. Specifically, we propose a recursive DiPCA that greatly reduces the computational costs of model updating by consistently updating the mean and variance of the sample and augmented sample vectors. Therefore, the auto-covariance matrices are updated in an efficient way that only calculation on new samples is required. The corresponding efficient deflation method is also given to extract multiple DLVs. In addition, a recursive initialization method is proposed to further accelerate the computation by reducing the iteration times in solving the optimization problem.

The subsequent parts of this paper are organized as follows. Section 2 reviews the dynamic-inner principal component analysis method. Section 3 introduces the proposed the recursive dynamic-inner principal component analysis algorithm. Section 4 gives experiments with simulation data and industrial data. In Section 5 we discuss the conclusions.

2. DYNAMIC-INNER PRINCIPAL COMPONENT ANALYSIS

2.1 Objective Function

The optimization problem of DiPCA can be formally described as follows. Let $\boldsymbol{x}_h$ be a $m \times 1$ time series at time h where the dynamics are concentrated in a subspace spanned by $\mathbf{P}$, then it can be represented by

$$\boldsymbol{x}_h = \mathbf{P}\boldsymbol{t}_h + \boldsymbol{\epsilon}_h. \tag{1}$$

The l-dimensional ($l < m$) dynamic latent vector $\boldsymbol{t}_h$ contains all the dynamics in $\boldsymbol{x}_h$.

By focusing on the first dynamic latent variable, we have a scalar AR inner model as

$$t_h = \beta_1 t_{h-1} + \cdots + \beta_s t_{h-s} + r_h, \tag{2}$$

with the latent variable extracted from the original variables by a linear outer model as

$$t_h = \boldsymbol{x}_h^T \mathbf{w}. \tag{3}$$

Then the prediction of $\hat{t}_h$ formed according to (2) can be written as

$$\begin{aligned} \hat{t}_h &= \boldsymbol{x}_{h-1}^T \mathbf{w}\beta_1 + \boldsymbol{x}_{h-2}^T \mathbf{w}\beta_2 + \cdots + \boldsymbol{x}_{h-s}^T \mathbf{w}\beta_s \\ &= \left[\boldsymbol{x}_{h-1}^T \ \boldsymbol{x}_{h-2}^T \ \cdots \ \boldsymbol{x}_{h-s}^T\right](\boldsymbol{\beta} \otimes \mathbf{w}), \end{aligned} \tag{4}$$

where $\boldsymbol{\beta} = [\beta_1 \ \beta_2 \ \cdots \ \beta_s]$.

If raw samples are collected as a data block

$$\begin{aligned} \mathbf{X}^0 &= \left[\boldsymbol{x}_1^0 \ \boldsymbol{x}_2^0 \ \cdots \ \boldsymbol{x}_{s+N}^0\right]^T, \\ \mathbf{X}_i^0 &= \left[\boldsymbol{x}_i^0 \ \boldsymbol{x}_{i+1}^0 \ \cdots \ \boldsymbol{x}_{i+N-1}^0\right]^T, \text{ for } i = 1, \ldots, s+1. \end{aligned} \tag{5}$$

For easy representation, we refer to the current model as the kth model. Then the data should be standardized before model building using the current mean vector $\mathbf{b}_k$ and a diagonal matrix $\boldsymbol{\Sigma}_k$ containing the standard deviations as

$$\begin{aligned} \mathbf{X} &= (\mathbf{X}^0 - \mathbf{1}_{s+N}\mathbf{b}_k^T)\boldsymbol{\Sigma}_k^{-1}, \\ \mathbf{X}_i &= (\mathbf{X}_i^0 - \mathbf{1}_N\mathbf{b}_k^T)\boldsymbol{\Sigma}_k^{-1}, \end{aligned}$$

where

$$\begin{aligned} &\mathbf{b}_k = \frac{1}{s+N}(\mathbf{X}^0)^T\mathbf{1}_{s+N}, \\ &\mathbf{1}_j = [1 \quad 1 \quad \cdots \quad 1]^T \in \mathcal{R}^j, \quad \text{for} \quad j = N, s+N, \\ &\boldsymbol{\Sigma}_k = diag(\sigma_{k,1}, \sigma_{k,2}, \cdots, \sigma_{k,m}), \end{aligned}$$

The element $\sigma_{k,i}$ in $\boldsymbol{\Sigma}_k$ denotes the standard deviation of the ith variable.

Then define

$$\begin{aligned} &\mathbb{X}_k = \mathbf{X}_{s+1}, \\ &\mathbb{Z}_k = [\mathbf{X}_s \quad \mathbf{X}_{s-1} \quad \cdots \quad \mathbf{X}_1], \\ &\mathbf{R}_k^{zx} = \mathbb{Z}_k^T\mathbb{X}_k/(N-1), \end{aligned}$$

where $\mathbf{R}_k^{zx}$ essentially contains a series of auto-covariance matrices of the standardized $\boldsymbol{x}_h$ from lag 1 to lag s.

Aiming to extract explicit dynamic latent relations, DiPCA objective function maximizes the covariance between t_h and $\hat{t}_h$, which can further be derived as

$$\begin{aligned} \max_{\mathbf{w},\boldsymbol{\beta}} . \quad & \mathbf{w}^T(\mathbf{R}_k^{zx})^T(\boldsymbol{\beta} \otimes \mathbf{w}) \\ \text{s.t.} \quad & \|\mathbf{w}\| = 1, \quad \|\boldsymbol{\beta}\| = 1. \end{aligned} \tag{6}$$

2.2 Algorithm

According to Dong and Qin (2018), the optimization problem (6) can be solved by inducting Lagrangian multipliers. The key parameters $\mathbf{w}$ and $\boldsymbol{\beta}$ for one DLV extraction can be obtained using the following algorithm.

Algorithm 1 Iterative algorithm to solve problem (6).

Input: $\mathbf{R}_k^{zx}$ and order s
1: Initialize a random unit vector $\mathbf{w}$
2: **repeat**
3: $\quad \boldsymbol{\beta} = (\mathbf{I}_s \otimes \mathbf{w})^T\mathbf{R}_k^{zx}\mathbf{w}$
4: $\quad \boldsymbol{\beta} := \boldsymbol{\beta}/\|\boldsymbol{\beta}\|$
5: $\quad$ Find $\mathbf{w}$ as the eigenvector corresponding to the largest eigenvalue of $(\mathbf{R}_k^{zx})^T(\boldsymbol{\beta}\otimes\mathbf{I}_m)+(\boldsymbol{\beta}\otimes\mathbf{I}_m)^T\mathbf{R}_k^{zx}$
6: **until** convergence
Output: Parameters $\mathbf{w}$ and $\boldsymbol{\beta}$

After solving $\mathbf{w}$ and $\boldsymbol{\beta}$, the predicted DLV can be obtained by a inner model

$$\hat{\mathbf{t}}_{s+1} = \alpha_1\mathbf{t}_s + \alpha_2\mathbf{t}_{s-1} + \cdots + \alpha_s\mathbf{t}_1, \tag{7}$$

where

$$\mathbf{t}_i = \mathbf{X}_i\mathbf{w}. \tag{8}$$

The coefficients $\boldsymbol{\alpha} = [\alpha_1 \ \alpha_2 \ \cdots \ \alpha_s]^T$ is the least square solution by

$$\boldsymbol{\alpha} = (\mathbf{T}_s^T\mathbf{T}_s)^{-1}\mathbf{T}_s^T\mathbf{t}_{s+1}, \tag{9}$$

where

$$\mathbf{T}_s = [\mathbf{t}_s \ \mathbf{t}_{s-1} \ \cdots \ \mathbf{t}_1] = \mathbb{Z}_k(\mathbf{I}_s \otimes \mathbf{w}). \tag{10}$$

To extract the next DLV, the data matrix should be deflated to remove the impact from the current DLV, $\mathbf{t} = \mathbf{X}\mathbf{w}$, as

$$\mathbf{X} := \mathbf{X} - \mathbf{t}\mathbf{p}^T, \tag{11}$$

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

$$\mathbf{p} = \mathbf{X}^T\mathbf{t}/\mathbf{t}^T\mathbf{t}. \tag{12}$$

Then the deflated $\mathbf{X}$ can form $\mathbb{X}_k$ and $\mathbb{Z}_k$. The next DLV can be extracted by repeating the same algorithm above.

3. RECURSIVE DYNAMIC-INNER PRINCIPAL COMPONENT ANALYSIS

If a new DiPCA model is required when new raw samples are collected after the initial samples as

$$\mathbf{X}^0_{new} = \begin{bmatrix} \boldsymbol{x}^0_{s+N+1} & \boldsymbol{x}^0_{s+N+2} & \cdots & \boldsymbol{x}^0_{s+N+L} \end{bmatrix}^T, \tag{13}$$

a direct way is to repeat the processes in Section 2.1 by adding the new raw samples in (5) to re-organize the whole data matrix as

$$\mathbf{X}^0_{k+1} = \begin{bmatrix} \mathbf{X}^0 \\ \mathbf{X}^0_{new} \end{bmatrix}. \tag{14}$$

Considering the time complexity, this approach has to calculate all the required relations of $\mathbf{X}^0_{k+1}$. For space complexity, the algorithm has to memorize the old data matrix. Undoubtedly, its computational costs increase rapidly when the model updates over and over.

By noticing that the calculation procedure of Algorithm 1 is only related to $\mathbf{R}^{zx}_k$, a recursive algorithm is proposed to efficiently update the model by keeping and updating the autocovariance matrices as follow.

3.1 Recursive Updating of Autocovariance Matrices

In order to update the autocovariance matrices, the mean and variance update strategy is first explored. The updated mean vector can be represented with relation to the initial mean vector as

$$\mathbf{b}_{k+1} = \frac{1}{s+N+L}\left[(s+N)\,\mathbf{b}_k + \left(\mathbf{X}^0_{new}\right)^T \mathbf{1}_L\right]. \tag{15}$$

The recursive calculation of the standard deviation of the jth original variable is

$$\begin{aligned}(s+N+L-1)\sigma^2_{k+1,j} =& (s+N-1)\sigma^2_{k,j} + (s+N)\Delta b^2_{k+1}(j) \\ &+ \|\mathbf{X}^0_{new}(:,j) - \mathbf{1}_L b_{k+1}(j)\|^2,\end{aligned} \tag{16}$$

where $\Delta\mathbf{b}_{k+1} = \mathbf{b}_{k+1} - \mathbf{b}_k$; $b_{k+1}(j)$ and $\Delta b_{k+1}(j)$ are the j-th elements of the corresponding vectors; $\mathbf{X}^0_{new}(:,j)$ is the j-th column of $\mathbf{X}^0_{new}$. The detailed derivation can be found in Li et al. (2000).

Then the standardized $\mathbf{X}^0_{k+1}$ is calculated by

$$\mathbf{X}_{k+1} = (\mathbf{X}^0_{k+1} - \mathbf{1}_{s+N+L} b^T_{k+1})\boldsymbol{\Sigma}^{-1}_{k+1}, \tag{17}$$

where $\boldsymbol{\Sigma}_{k+1} = diag(\sigma_{k+1,1}, \sigma_{k+1,2}, \cdots, \sigma_{k+1,m})$.

To update the autocovariance matrices, which is essentially equivalent to the update of $\mathbf{R}^{zx}_k$, $\mathbb{X}_k$ and $\mathbb{Z}_k$ should be updated in advance. The derivation of the recursive update of $\mathbb{X}_{k+1}$ is

$$\begin{aligned}\mathbb{X}_{k+1} &= \left[\begin{bmatrix} \mathbf{X}^0_{s+1} \\ \mathbf{X}^0_{new} \end{bmatrix} - \mathbf{1}_{N+L}\mathbf{b}^T_{k+1}\right]\boldsymbol{\Sigma}^{-1}_{k+1} \\ &= \left[\begin{bmatrix} \mathbb{X}_k\boldsymbol{\Sigma}_k + \mathbf{1}_N\mathbf{b}^T_k \\ \mathbf{X}^0_{new} \end{bmatrix} - \mathbf{1}_{N+L}\mathbf{b}^T_{k+1}\right]\boldsymbol{\Sigma}^{-1}_{k+1} \\ &= \begin{bmatrix} \mathbb{X}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}^{-1}_{k+1} - \mathbf{1}_N\Delta\mathbf{b}^T_{k+1}\boldsymbol{\Sigma}^{-1}_{k+1} \\ \mathbf{X}_{new} \end{bmatrix}.\end{aligned} \tag{18}$$

where $\mathbf{X}_{new} = (\mathbf{X}^0_{new} - \mathbf{1}_L\mathbf{b}^T_{k+1})\boldsymbol{\Sigma}^{-1}_{k+1}$.

$\mathbf{Z}^0_{k+1}$ is formed with $\mathbf{X}^0_{k+1}$ as

$$\mathbf{Z}^0_{k+1} = \begin{bmatrix} \mathbf{Z}^0_k \\ \mathbf{Z}^0_{new} \end{bmatrix}, \tag{19}$$

where

$$\begin{aligned}\mathbf{Z}^0_k &= \begin{bmatrix} \mathbf{X}^0_s & \mathbf{X}^0_{s-1} & \cdots & \mathbf{X}^0_1 \end{bmatrix}, \\ \mathbf{Z}^0_{new} &= \begin{bmatrix} \boldsymbol{x}^0_{s+N} & \boldsymbol{x}^0_{s+N-1} & \cdots & \boldsymbol{x}^0_{N+1} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{x}^0_{s+N+L-1} & \boldsymbol{x}^0_{s+N+L-2} & \cdots & \boldsymbol{x}^0_{N+L} \end{bmatrix}.\end{aligned}$$

The blocks in $\mathbf{Z}^0_{k+1}$ should have the same $\mathbf{b}_{k+1}$ and $\boldsymbol{\Sigma}_{k+1}$ as $\mathbf{X}^0_{k+1}$, since they are submatrices of $\mathbf{X}^0_{k+1}$. Therefore, the following intermediate vector and matrix can be introduced

$$\begin{aligned}\mathbf{a}_{k+1} &= \mathbf{1}_s \otimes \mathbf{b}_{k+1}, \\ \boldsymbol{\Xi}_{k+1} &= \mathbf{I}_s \otimes \boldsymbol{\Sigma}_{k+1}.\end{aligned}$$

Then, the recursive update of $\mathbb{Z}_{k+1}$ is calculated as

$$\mathbb{Z}_{k+1} = \begin{bmatrix} \mathbb{Z}_k\boldsymbol{\Xi}_k\boldsymbol{\Xi}^{-1}_{k+1} - \mathbf{1}_N\Delta\mathbf{a}^T_{k+1}\boldsymbol{\Xi}^{-1}_{k+1} \\ \mathbf{Z}_{new} \end{bmatrix}, \tag{20}$$

where

$$\begin{aligned}\Delta\mathbf{a}_{k+1} &= \mathbf{a}_{k+1} - \mathbf{a}_k = \mathbf{1}_s \otimes \Delta\mathbf{b}_{k+1}, \\ \mathbf{Z}_{new} &= (\mathbf{Z}^0_{new} - \mathbf{1}_L\mathbf{a}^T_{k+1})\boldsymbol{\Xi}^{-1}_{k+1}.\end{aligned}$$

Following by the update of $\mathbb{Z}_{k+1}$ and $\mathbb{Z}_{k+1}$, the update of $\mathbf{R}^{zx}_{k+1}$ that consists of autocovariance matrices from lag 1 to lag s can be expressed as

$$\begin{aligned}\mathbf{R}^{zx}_{k+1} =& \frac{1}{N+L-1}\mathbb{Z}^T_{k+1}\mathbb{X}_{k+1} \\ =& \frac{N-1}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\boldsymbol{\Xi}_k\mathbf{R}^{zx}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}^{-1}_{k+1} \\ &- \frac{1}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\boldsymbol{\Xi}_k\mathbb{Z}^T_k\mathbf{1}_N\Delta\mathbf{b}^T_{k+1}\boldsymbol{\Sigma}^{-1}_{k+1} \\ &- \frac{1}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\Delta\mathbf{a}_{k+1}\mathbf{1}^T_N\mathbb{X}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}^{-1}_{k+1} \\ &+ \frac{N}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\Delta\mathbf{a}_{k+1}\Delta\mathbf{b}^T_{k+1}\boldsymbol{\Sigma}^{-1}_{k+1} \\ &+ \frac{1}{N+L-1}\mathbf{Z}^T_{new}\mathbf{X}_{new}.\end{aligned} \tag{21}$$

Regarding the above relation, we can have the following assumption in most cases of industrial process data:

(1) The number of samples for training is large such that $N >> s > 1$. Then we have $\frac{1}{N+L-1}\mathbb{Z}^T_k\mathbf{1}_N \approx \mathbf{0}$, $\frac{1}{N+L-1}\mathbf{1}^T_N\mathbb{X}_k \approx \mathbf{0}$, and therefore

$$\begin{aligned}&\frac{1}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\boldsymbol{\Xi}_k\mathbb{Z}^T_k\mathbf{1}_N\Delta\mathbf{b}^T_{k+1}\boldsymbol{\Sigma}^{-1}_{k+1} \approx 0, \\ &\frac{1}{N+L-1}\boldsymbol{\Xi}^{-1}_{k+1}\Delta\mathbf{a}_{k+1}\mathbf{1}^T_N\mathbb{X}_k\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}^{-1}_{k+1} \approx 0.\end{aligned} \tag{22}$$

(2) The amount of new samples is much less than that of the initial samples $N >> L$, and perturbation on the variance is not significant, then

$$\begin{aligned}\boldsymbol{\Xi}_k\boldsymbol{\Xi}_{k+1}^{-1} &\approx \mathbf{I}_{sm},\\ \boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_{k+1}^{-1} &\approx \mathbf{I}_m.\end{aligned} \tag{23}$$

(3) Old samples are exponentially ignored as they do not represent the current process when process dynamics changes. The recursive calculations for (15) and (16) with a forgetting factor μ are:

$$\begin{aligned}\mathbf{b}_{k+1} =&\mu\mathbf{b}_k + (1-\mu)\frac{1}{L}(\mathbf{X}_{new}^0)^T\mathbf{1}_L,\\ \sigma_{k+1,j} =&\mu(\sigma_{k,j}^2 + (s+N)\Delta b_{k+1}^2(j))\\ &+ (1-\mu)\frac{1}{L}\|\mathbf{X}_{new}^0(:,j) - \mathbf{1}_L b_{k+1}(j)\|^2.\end{aligned} \tag{24}$$

Therefore, the recursive calculation of $\mathbf{R}_{k+1}^{zx}$ in (21) reduces to

$$\begin{aligned}\mathbf{R}_{k+1}^{zx} =&\mu\left(\mathbf{R}_k^{zx} + \mathbf{1}_s\otimes(\boldsymbol{\Sigma}_{k+1}^{-1}\Delta\mathbf{b}_{k+1}\Delta\mathbf{b}_{k+1}^T\boldsymbol{\Sigma}_{k+1}^{-1})\right)\\ &+ (1-\mu)\frac{1}{L}\mathbf{Z}_{new}^T\mathbf{X}_{new}.\end{aligned} \tag{25}$$

By now, a new DLV that describes the current dynamics can be obtained by Algorithm 1.

3.2 Recursive Parameter Initialization

Recall that in Algorithm 1, the initialization of the parameter $\mathbf{w}$ exerts a great influence on the convergence speed. To reduce the computational costs further, a recursive parameter initialization strategy is proposed here.

When the projection matrix $\mathbf{W}_k = \left[\mathbf{w}_k^1, \mathbf{w}_k^2, \cdots, \mathbf{w}_k^l\right]$ of the k-th model is known, where l is the number of DLVs, we initialize the first $\mathbf{w}$ of the $(k+1)$-th model as

$$\mathbf{w}_{k+1}^1 = \mathbf{w}_k^1. \tag{26}$$

After the i-th vector $\mathbf{w}_{k+1}^i$ of the $(k+1)$-th model is obtained, where $i = 1, 2, \cdots, l-1$, the $\mathbf{w}_{k+1}^{i+1}$ of the next DLV is initialized as

$$\begin{aligned}&\mathbf{W}_{k+1}^i = \left[\mathbf{w}_{k+1}^1, \mathbf{w}_{k+1}^2, \cdots, \mathbf{w}_{k+1}^i\right]\\ &\mathbf{w}_{k+1}^{i+1} = \left(\mathbf{I}_m - \mathbf{W}_{k+1}^i(\mathbf{W}_{k+1}^i)^T\right)\mathbf{w}_k^{i+1},\\ &\mathbf{w}_{k+1}^{i+1} := \mathbf{w}_{k+1}^{i+1}/\|\mathbf{w}_{k+1}^{i+1}\|.\end{aligned} \tag{27}$$

Obviously, the orthogonality of $\mathbf{W}_{k+1}$ are maintained from (27).

3.3 Deflation

After one DLV is extracted with the updated $\mathbf{R}_{k+1}^{zx}$, this component should be removed from the data when another DLV is required. According to (12), the loading vector $\mathbf{p}$ can be calculated with the current $\mathbf{X}_{k+1}$ as

$$\mathbf{p} = \frac{\mathbf{X}_{k+1}^T\mathbf{X}_{k+1}\mathbf{w}}{\mathbf{w}^T\mathbf{X}_{k+1}^T\mathbf{X}_{k+1}\mathbf{w}} = \frac{\mathbf{R}_{k+1}^{xx}\mathbf{w}}{\mathbf{w}^T\mathbf{R}_{k+1}^{xx}\mathbf{w}}. \tag{28}$$

where $\mathbf{R}_{k+1}^{xx} = \mathbf{X}_{k+1}^T\mathbf{X}_{k+1}/(N+s+L-1)$.

Similar to (25), the covariance matrix $\mathbf{R}_{k+1}^{xx}$ can be updated recursively as

$$\begin{aligned}\mathbf{R}_{k+1}^{xx} =&\mu\left(\mathbf{R}_k^{xx} + \boldsymbol{\Sigma}_{k+1}^{-1}\Delta\mathbf{b}_{k+1}\Delta\mathbf{b}_{k+1}^T\boldsymbol{\Sigma}_{k+1}^{-1}\right)\\ &+ (1-\mu)\frac{1}{L}\mathbf{X}_{new}^T\mathbf{X}_{new},\end{aligned} \tag{29}$$

Therefore, the calculation of $\mathbf{p}$ can be achieved after updating $\mathbf{R}_{k+1}^{xx}$.

Once $\mathbf{p}$ is calculated, $\mathbb{Z}_{k+1}$ and $\mathbb{X}_{k+1}$ can be deflated using $\mathbf{p}$ as in (11), which in turn leads to a deflation in $\mathbf{R}_{k+1}^{zx}$ as

$$\begin{aligned}\mathbf{R}_{k+1}^{zx} :=&\left[\mathbb{Z}_{k+1} - \mathbb{Z}_{k+1}(\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)\right]^T\left[\mathbb{X}_{k+1} - \mathbb{X}_{k+1}\mathbf{w}\mathbf{p}^T\right]\\ :=&\mathbf{R}_{k+1}^{zx} - (\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)^T\mathbf{R}_{k+1}^{zx} - \mathbf{R}_{k+1}^{zx}\mathbf{w}\mathbf{p}^T\\ &+ (\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)^T\mathbf{R}_{k+1}^{zx}\mathbf{w}\mathbf{p}^T.\end{aligned} \tag{30}$$

In addition, it can be inferred from (28) that the covariance matrix $\mathbf{R}_{k+1}^{xx}$ should also be deflated such that the next $\mathbf{p}$ can be calculated. The deflation of $\mathbf{R}_{k+1}^{xx}$ is

$$\begin{aligned}\mathbf{R}_{k+1}^{xx} :=&\left[\mathbf{X}_{k+1} - \mathbf{X}_{k+1}\mathbf{w}\mathbf{p}^T\right]^T\left[\mathbf{X}_{k+1} - \mathbf{X}_{k+1}\mathbf{w}\mathbf{p}^T\right]\\ :=&\mathbf{R}_{k+1}^{xx} - 2\mathbf{R}_{k+1}^{xx}\mathbf{w}\mathbf{p}^T + (\mathbf{w}\mathbf{p}^T)^T\mathbf{R}_{k+1}^{xx}\mathbf{w}\mathbf{p}^T.\end{aligned} \tag{31}$$

3.4 Inner model

The least square solution of the inner model (9) can be rewritten as

$$\boldsymbol{\alpha} = ((\mathbf{I}_s\otimes\mathbf{w})^T\mathbf{R}_k^{zz}(\mathbf{I}_s\otimes\mathbf{w}))^{-1}(\mathbf{I}_s\otimes\mathbf{w})^T\mathbf{R}_k^{zx}\mathbf{w}, \tag{32}$$

where $\mathbf{R}_k^{zz} = \mathbb{Z}_k^T\mathbb{Z}_k/(N-1)$.

Therefore, it can be calculated by the recursive update of $\mathbf{R}_{k+1}^{zz}$ as

$$\begin{aligned}\mathbf{R}_{k+1}^{zz} =&\mu\left(\mathbf{R}_k^{zz} + (\mathbf{1}_s\mathbf{1}_s^T)\otimes(\boldsymbol{\Sigma}_{k+1}^{-1}\Delta\mathbf{b}_{k+1}\Delta\mathbf{b}_{k+1}^T\boldsymbol{\Sigma}_{k+1}^{-1})\right)\\ &+ (1-\mu)\frac{1}{L}\mathbf{Z}_{new}^T\mathbf{Z}_{new}.\end{aligned} \tag{33}$$

When the inner model of the next DLV is required, $\mathbf{R}_{k+1}^{zz}$ should be deflated as

$$\begin{aligned}\mathbf{R}_{k+1}^{zz} :=&\bar{\mathbb{Z}}_{k+1}^T\bar{\mathbb{Z}}_{k+1}\\ =&\mathbf{R}_{k+1}^{zz} - 2\mathbf{R}_{k+1}^{zz}(\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)\\ &+ (\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)^T\mathbf{R}_{k+1}^{zz}(\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T),\end{aligned} \tag{34}$$

where $\bar{\mathbb{Z}}_{k+1} = \left[\mathbb{Z}_{k+1} - \mathbb{Z}_{k+1}(\mathbf{I}_s\otimes\mathbf{w}\mathbf{p}^T)\right]$.

So far, the RDiPCA algorithm can be organized as Algorithm 2. We can conclude that multiple DLVs and the corresponding inner model can be updated with the recursive update of $\mathbf{R}_{k+1}^{zx}$, $\mathbf{R}_{k+1}^{xx}$, and $\mathbf{R}_{k+1}^{zz}$. Additionally, for space complexity, RDiPCA only needs to memorize the three matrices $\mathbf{R}_k^{zx}$, $\mathbf{R}_k^{xx}$, $\mathbf{R}_k^{zz}$, and the corresponding mean vector and standard deviation. More importantly, the computational costs of updating these matrices will not increase since the scales of these data matrices do not accumulate.

4. CASE STUDY

We conduct experiments on a numerical case and a wastewater treatment plant case to find out the effectiveness of the proposed method. By comparing with DiPCA without update (DiPCA-N) and direct update (DiPCA-D), the RDiPCA achieves significant improvement in time-varying DLV extraction, computational costs, and prediction accuracy.

Algorithm 2 RDiPCA algorithm to extract multiple DLVs.

Input: Previous model $\mathbf{R}_k^{zx}, \mathbf{R}_k^{xx}, \mathbf{R}_k^{zz}, \mathbf{b}_k, \mathbf{\Sigma}_k, \mathbf{W}_k$; new raw samples $\mathbf{X}_{new}^0$; order s; number of DLVs l

Update $\mathbf{b}_{k+1}, \mathbf{\Sigma}_{k+1}$ using (15), (16)
Update $\mathbf{R}_{k+1}^{zx}, \mathbf{R}_{k+1}^{xx}, \mathbf{R}_{k+1}^{zz}$ using (25), (29), (33)
for $i = 1, 2, \cdots, l$ **do**
  if $i = 1$ **then**
    Initialize $\mathbf{w}_{k+1}^i$ using (26)
  else
    Initialize $\mathbf{w}_{k+1}^i$ using (27)
  end if
  repeat
    $\boldsymbol{\beta}^i = (\mathbf{I}_s \otimes \mathbf{w}^i)^T \mathbf{R}_{k+1}^{zx} \mathbf{w}^i$
    $\boldsymbol{\beta}^i := \boldsymbol{\beta}^i / \|\boldsymbol{\beta}^i\|$
    $\mathbf{w}^i$ is the eigenvector corresponding to the largest eigenvalue of $(\mathbf{R}_{k+1}^{zx})^T(\boldsymbol{\beta}^i \otimes \mathbf{I}_s) + (\boldsymbol{\beta}^i \otimes \mathbf{I}_s)^T \mathbf{R}_{k+1}^{zx}$
    $\mathbf{w}^i := \mathbf{w}^i / \|\mathbf{w}^i\|$
  until convergence
  Inner model $\boldsymbol{\alpha}^i = ((\mathbf{I}_s \otimes \mathbf{w}^i)^T \mathbf{R}_{k+1}^{zz} (\mathbf{I}_s \otimes \mathbf{w}^i))^{-1} (\mathbf{I}_s \otimes \mathbf{w}^i)^T \mathbf{R}_{k+1}^{zx} \mathbf{w}^i$
  Deflation using (30), (31), (34)
end for

Output: Updated model $\mathbf{R}_{k+1}^{zx}$, $\mathbf{R}_{k+1}^{xx}$, $\mathbf{R}_{k+1}^{zz}$, $\mathbf{b}_{k+1}$, $\mathbf{\Sigma}_{k+1}$, $\mathbf{W}_{k+1}$

4.1 Numerical case

In this numerical case, $\boldsymbol{t}_k$ is generated from a time-varying vector auto-regression process, and $\mathbf{x}_k$ is generated from a latent variable model as

$$\boldsymbol{t}_k = \mathbf{A}\boldsymbol{t}_{k-1} + \boldsymbol{v}_k$$
$$\boldsymbol{x}_k = \mathbf{P}\boldsymbol{t}_k + \boldsymbol{e}_k,$$

where $\boldsymbol{v}_k \in \mathbb{R}^3 \sim N([0, 1^2])$, $\boldsymbol{e}_k \in \mathbb{R}^{10\times 1} \sim N([0, 1^2])$, and $\mathbf{P} \in \mathbb{R}^{10\times 3}$ sampling from $\mathbb{R}^{10\times 3} \sim N([0, 1^2])$. Then samples are continuously generated from a initial coefficient matrix $\mathbf{A}_1$ to a updated coefficient matrix $\mathbf{A}_2$, where

$$\mathbf{A}_1 = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 0.3 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0.9 & 0 & 0 \\ 0 & 0.4 & 0 \\ 0 & 0 & 0.3 \end{pmatrix}.$$

22000 samples are generated in total and the coefficient matrix changes at the 20000-th point. We choose $s = 1$ and $l = 3$ to fit the models.

Fig. 1 shows the DLVs extraction results of several methods. By comparing the auto-correlation of the DLVs and the corresponding dynamic error (DR), it can be concluded that when dynamic changes, DiPCA-N and DiPCA-D can not accurately extract and model the auto-correlation of the first DLV, resulting in time-dependent DR. Meanwhile, RDiPCA extracts all the dynamics in DLVs. Fig. 2 illustrates the prediction results on the first variable (X1). RDiPCA shows its best prediction accuracy among these methods.

Table 1 summarizes several indices of these methods, where RDiPCA has the best performance in terms of smaller mean square error (MSE), more accurate eigenvalues and lower computational costs. Notably, despite the number of initial samples is quite large, the Computational Time (CT) of RDiPCA is not affected. This demonstrates

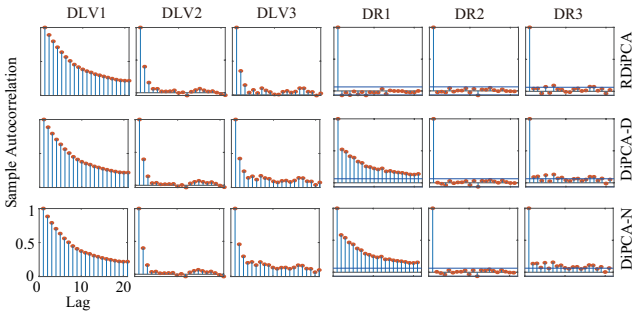

Fig. 1. DLVs extraction results when dynamic changes. DLV(k) means the k-th dynamic latent variable. DR(k) means the k-th dynamic error.

RDiPCA ensures the CT on covariance matrices not increase as model updates.

Table 1. Performance comparison between different methods in simulation data.

	CT	MSE	eigenvalues		
DiPCA-N	0.010403s	0.6224	0.4919	0.4004	0.3207
DiPCA-D	0.010499s	0.5908	0.5226	0.4002	0.3176
RDiPCA	0.001272s	0.5050	0.8936	0.4024	0.3045
Real			0.9	0.4	0.3

4.2 Wastewater treatment plant-BSM1

This is a Benchmark Simulation Model 1 (BSM1) developed by the International Water Association (IWA), which aims to realize a long-term simulation study in the whole wastewater treatment plant (WWTP). A number of researches related to chemical engineering have been taken on this dataset to verify their effectiveness in process modeling, control, and monitoring. In this case, 17 process variables (PV1-17) are used, and 4000 continuous samples with time-varying dynamics are collected. The first 2700 samples are used to train an initial model, followed by 500 samples to update it, and the last 800 to test.

After normalizing with the same scaling, we compare the prediction performance of the three methods. Fig. 3 shows the prediction results on PV1. It can seen that RDiPCA prediction is the closest to the real value, especially on the peak points. Table 2 shows the corresponding MSEs of all the process variables and CTs. It illustrates RDiPCA is able to capture the dynamic changes more appropriately while costs less time to update the model.

Table 2. Performance comparison between different methods in steam process data.

	CT	MSE
DiPCA-N	0.011282s	0.7193
DiPCA-D	0.013447s	0.3717
RDiPCA	0.005570s	0.0698

5. CONCLUSION

In this paper, an RDiPCA is proposed to model high-dimensional, dynamic, and time-varying processes. Model performances are maintained via a recursive update strategy to follow process dynamic changes. Meanwhile, the low computational costs of RDiPCA make online updates

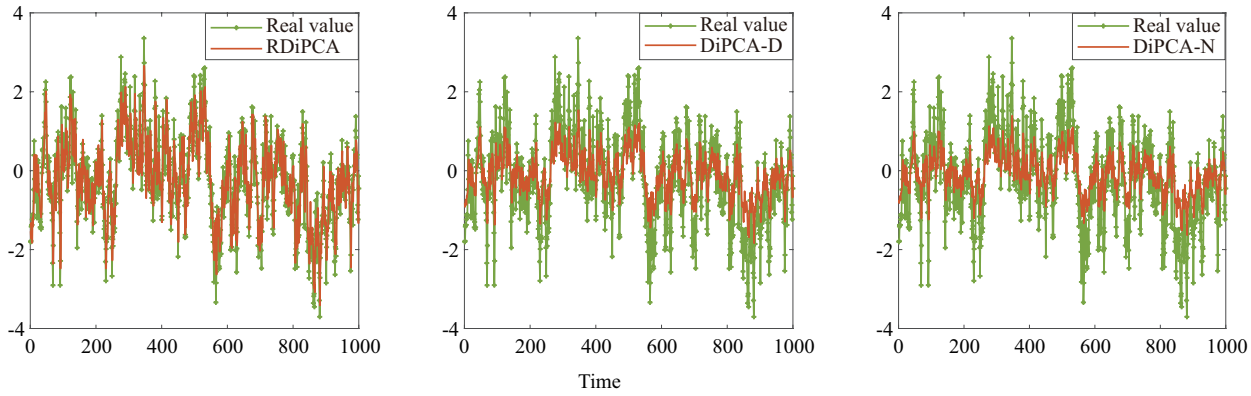

Fig. 2. Prediction with lower dimensional dynamics on the first variable of simulation data.

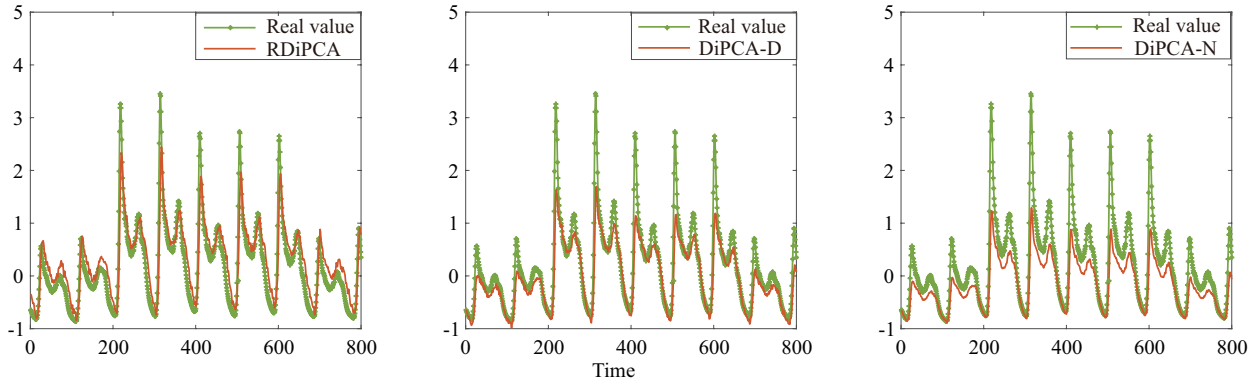

Fig. 3. Prediction with lower dimensional dynamics on the MV8 of steam process data.

of the dynamic model feasible. RDiPCA, therefore, can be directly used to model and monitor nonstationary processes. Case studies on a numerical and a wastewater treatment plant demonstrate the effectiveness of the proposed RDiPCA algorithm.

REFERENCES

Basanta-Val, P. (2018). An efficient industrial big-data engine. *IEEE Transactions on Industrial Informatics*, 14(4), 1361–1369.

Dong, Y., Liu, Y., and Joe Qin, S. (2020). Efficient dynamic latent variable analysis for high-dimensional time series data. *IEEE Transactions on Industrial Informatics*, 16(6), 4068–4076.

Dong, Y. and Qin, S. (2018). A novel dynamic pca algorithm for dynamic data modeling and process monitoring. *Journal of Process Control*, 67, 1–11.

Feng, X., Kong, X., He, C., and Luo, J. (2022). High-dimensional, slow-time-varying process monitoring technique based on adaptive eigen subspace extraction method. *Journal of Process Control*, 117, 122–131.

Guo, F., Shang, C., Huang, B., Wang, K., Yang, F., and Huang, D. (2016). Monitoring of operating point and process dynamics via probabilistic slow feature analysis. *Chemometrics and Intelligent Laboratory Systems*, 151, 115–125.

Hajarian, N., Sobhani, F., and Sadjadi, S. (2020). An improved approach for fault detection by simultaneous overcoming of highdimensionality, autocorrelation, and timevariability. *PLoS ONE*, 15(12 December).

Hu, Z., Chen, Z., Hua, C., Gui, W., Yang, C., and Ding, S. (2012). A simplified recursive dynamic pca based monitoring scheme for imperial smelting process. *International Journal of Innovative Computing, Information and Control*, 8(4), 2551–2561.

Ku, W., Storer, R.H., and Georgakis, C. (1995). Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 30(1), 179 – 196. Cited by: 1284.

Li, G., Qin, S., and Zhou, D. (2014). A new method of dynamic latent-variable modeling for process monitoring. *IEEE Transactions on Industrial Electronics*, 61(11), 6438–6445.

Li, G., Liu, B., Qin, S.J., and Zhou, D. (2011). Dynamic latent variable modeling for statistical process monitoring. volume 44, 12886 – 12891.

Li, W., Yue, H., Valle-Cervantes, S., and Qin, S. (2000). Recursive pca for adaptive process monitoring. *Journal of Process Control*, 10(5), 471–486.

Qin, S. (1998). Recursive pls algorithms for adaptive data modeling. *Computers and Chemical Engineering*, 22(4-5), 503–514.

Qin, S., Dong, Y., Zhu, Q., Wang, J., and Liu, Q. (2020). Bridging systems theory and data science: A unifying review of dynamic latent variable analytics and process monitoring. *Annual Reviews in Control*, 50, 29–48.

Shang, C., Yang, F., Huang, B., and Huang, D. (2018). Recursive slow feature analysis for adaptive monitoring of industrial processes. *IEEE Transactions on Industrial Electronics*, 65(11), 8895–8905.