IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 18, 2020, accepted June 10, 2020, date of publication June 15, 2020, date of current version June 26, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3002257*

# Segmentation of Multivariate Industrial Time Series Data Based on Dynamic Latent Variable Predictability

**SHAOWEN LU, (Member, IEEE), AND SHUYU HUANG**
State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China

Corresponding author: Shaowen Lu (lusw@mail.neu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61833004, and in part by the Youth Science and Technology Innovation Leader of Shenyang under Grant RC190443.

**ABSTRACT** Time series segmentation is an important vehicle of data mining and extensively applied in the areas of machine learning and anomaly detection. In real world tasks, dynamics widely exist in time series but have been little concerned. This paper proposes an algorithm which can partition a multivariate time series into subsequences at points where the dynamics change. In process industries, the change of dynamics often relates to operation regime change, working condition shifting or faults. Therefore, to segment time series according to change of dynamics can be useful in data preprocessing and getting deep insight of the process in various industrial process monitoring tasks. The proposed algorithm recursively merges neighborhood subsequences through a heuristic bottom-up procedure. The cost of merging is defined as the mutual predictability of the subsequence models which are constructed using the dynamic-inner principal component analysis algorithm. Then, the goal becomes finding the segmentation scheme which minimizes the averaged dynamic prediction errors of each subsequence model. The proposed algorithm is evaluated on both simulated data and the time series data collected from an industrial processing plant. The results show that it outperforms the static principal component analysis based methods.

**INDEX TERMS** Time series analysis, time series prediction, time series segmentation, change point detection, dynamic inner model, dynamic principal component analysis.

## I. INTRODUCTION

There has been a growing recognition of the value of time series data because many applications, such as anomaly detection [1], [2], working condition perception [3], [4] and soft measurement [5]–[7], are being enabled by the rapid development of data driven algorithms. Time series segmentation is to partition a given time series into subsequences which are internally homogeneous [8]. By segmentation, each subsequence is expected to be interpreted by a relatively simple model so as to reduce the representational complexity of the data. The problem of time series segmentation can be considered as an unsupervised clustering problem with the constraints that the data in each cluster are successive and has immutable order [9]. Most of the basic segmentation approaches are rooted on the idea of piecewise linear approximation (PLA) [10]. Then, the problem of finding the best segmentation can be formulized as a problem of finding the segmentation scheme which minimizes the overall or the averaged approximation error. The implementation of a PLA model is straightforward, e.g. by using linear regression and interpolation for univariate time series data [10].

In many real world tasks, time series segmentation serves as a data pre-treatment tool for representation reduction [11]–[15], change point detection [4], [15]–[18], feature extraction [17], [19] and more. Especially, accurate segmentation of the time series is often the key for the success of unsupervised or semi-supervised time series classification tasks [20].

In process industry, a processing plant constantly generates a huge volume of time series data by sensors, assay reports, operators, control systems, surveillance systems and more. Time series segmentation can be applied on time-resolved measurements to identify major changes in ongoing process. The changes in the dynamics are usually associated

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

with major events contributing to the behavior change of the system, such as operation regime change, working condition shifting or faults [21]. Detecting those changes is an important task for process control, optimization and safety assurance. Industrial time series data commonly include more variables, e.g. flow rates, temperatures, pressures and compositions, and they can be highly correlated. For these situations, multivariate time series segmentation can offer significant advantages over the univariate methods. Various approaches have be developed for the multivariate industrial time series segmentation problem. A nature extension of the PLA approach is to use principal component analysis (PCA) to linearly represent the subsequence data. Then, the Hotelling $T^2$ statistic and reconstruction error are employed as measures of the approximation errors in finding the optimal segmentation [22], [23]. A further approach extends the PCA based approach to a probabilistic framework [9], [24].

However, industrial time series data typically show both cross-correlation and autocorrelation, and the structure of those are likely to changes due to uncontrollable disturbances or switching of the operation regime. The PCA based approaches are often applicable to the situation where there is little cross-correlation among variables [25]. But they are not suitable for detecting the temporal correlation changes in the data. For the problem dynamic time series data segmentation, Dobos and Abonyi [3] proposed an algorithm based on the dynamic PCA (DPCA) [26] to support the detection of operation regime changes. DPCA uses an augmented data matrix including the time lagged process variables to extract the time-dependent relationships from data. Nonetheless, the DPCA model still relies on PCA and the structure extracted from the augmented data matrices is only determined by variance rather than the underlining temporal relationship. Therefore it is difficult for a DPCA model to differentiate the type of relations among the measured variables because autocorrelation and cross-correlation is mixed in the covariance matrix.

In view of the limitation of DPCA, Li *et al.* [27] proposed a dynamic latent variable (DLV) algorithm where a vector autoregressive (VAR) model is constructed for the latent variables extracted by the auto-regressive PCA to represent the dynamic relationships. Recently, Dong and Qin [28] proposed a dynamic inner PCA (DiPCA) algorithm which is able to extract explicitly the latent variables capturing the most dynamic variations in the data. This makes it possible to extract an embedded model of the dynamic latent variable which are best predicted from the past values.

Inspired by the idea of DiPCA, this paper proposes a new multivariate time series segmentation algorithm. The goal is to partition time series data into piece-wise stationary subsequences using dynamic predictability as a criterion. Assuming that the time series data are white noised and autocorrelated, the key idea of this work is to represent the time series data using a dynamic latent variable (DLV) model as inner model and a PCA mapping model as outer model.

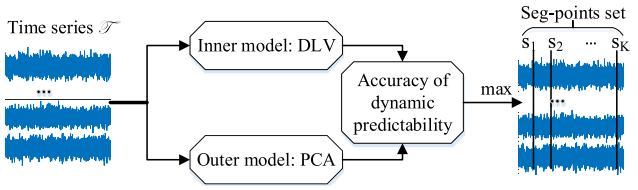



**FIGURE 1.** The main idea of the proposed method.

A global objective function of the dynamic prediction accuracy is constructed by choosing the latent variables which are mostly relevant to the dynamic auto-correlation. Then the best segmentation scheme can be obtained by maximizing the global objective, which corresponds to a strong dynamic autocorrelation for data in a subsequence. The main idea is shown in Figure 1.

The remainder of this paper is structured as follows. Section 2 formally defines the segmentation problem and introduces the DiPCA algorithm and the proposed cost function. In section 3, the parameters of the proposed algorithm are discussed using a synthetic dataset. Section 4 presents the test results on both simulated and real-world industrial datasets. And section 5 concludes this work.

## II. THE MUTIVARIATE TIME SERIES SEGMENTATION ALGORITHM

### A. DESCRIPTION OF PROBLEM

Considering a multivariate time series data $\mathcal{T} = \{x_k \in \mathbb{R}^m | 1 \leq k \leq N\}$, the goal of segmentation is to find a set of points to divide $\mathcal{T}$ into $K$ subsequences which are piecewise homogeneous. Let $S_i(a_i, b_i) = \{x_{a_i}, x_{a_i+1}, \ldots, x_{b_i}\}$ be a non-overlapping subsequence in $\mathcal{T}$, where $a_i$ and $b_i$ are partitioning points and $1 = a_1 \leq a_i < b_i \leq b_K = N$, $b_i + 1 = a_{i+1}$ and $K$ is the number of subsequences. A segmentation scheme is defined as $S_T^K = \{S_i(a_i, b_i) | 1 \leq i \leq K\}$, which represents a partition of $\mathcal{T}$ into $K$ non-overlapping subsequences.

In order to find the best segmentation scheme, first it needs a measure of the quality of a segmentation scheme. Denote $cost(S_i(a_i, b_i))$ as the cost function for a given subsequence $S_i(a_i, b_i)$ for this purpose. The formalization of the cost function depends on the characteristics of the time series data. Most often it is defined based on the distance between the original subsequence and the data fitted model. For example, the PCA similarity factor was used in [29]–[31], and the dynamic time warping (DTW) is used in [32]. If the probability distribution of the data is known, then the data in a subsequence can be treated as samples and the likelihood is used as the cost function [18]. Based on the cost function, the global objective function of a given segmentation scheme is

$$G = \frac{1}{K} \sum_{i=1}^{K} cost(S_i(a_i, b_i)) \tag{1}$$

where $K$ is the number of subsequences. Then the segmentation problem can be considered as an optimization

problem searching partitioning points $a_i$ and $b_i$ to optimize $G$, subjected to constraint $b_i + 1 = a_{i+1}$.

### B. DYNAMIC LATENT VARIABLE MODEL

Let $\mathcal{T}$ represent a multivariate time series composed of the sensory data collected from a dynamic industrial process. For an idea segmentation scheme, each subsequence of segment can be assumed stationary or weakly stationary. Under this assumption, a subsequence in $\mathcal{T}$ can be described using a linear generative model based on vector auto-regressive (VAR) model. Since there inevitably exists noise in industrial process data, the model can be expressed using dynamic latent variables [27]:

$$t_k = A_0 + A_1 t_{k-1} + \cdots + A_s t_{k-s} + v_k \tag{2}$$

where $t_k \in \mathbb{R}^l$ is the DLV at time $k$, $A_0$ is a constant vector, $A_1, A_2, \ldots, A_s \in \mathbb{R}^{l\times l}$ are parameter matrices, $v_k \in \mathbb{R}^l$ is new information and $s$ is the number of time-lags. Then, $x_k$ in $\mathcal{T}$ can be generated by

$$x_k = Ct_k + e_k \tag{3}$$

where $C \in \mathbb{R}^{m\times l}$ is the output coefficient matrix and $e_k \in \mathbb{R}^m$ is white noise.

The key idea of this work is to synthesize a cost function based on the prediction accuracy of the dynamic model trained using the subsequence data. Assuming that the new information $v_k$ is independent and identically distributed (i.i.d) for each $k$, the change of dynamic structure can be detected by the error of predicting $t_k$ using its past values. This requires extracting the DLVs according to their auto-correlation. However, the DPCA algorithms cannot extract DLVs when static relationship dominates the variance. In this paper, a recently developed tool, DiPCA, introduced in [28] is employed. With DiPCA, the most predictable DLV, $t^j$, can be expressed as the weighted form:

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

where the superscript represents the $j$th principal DLV, and $t_k^j$ is a projection of primitive variables on a weight vector:

$$t_k^j = x_k^T w^j \tag{5}$$

$w^j$ is the weight vector, $\|w^j\| = 1$. $x_k$ represents a data sample at time $k$ in time series $\mathcal{T}$. $s$ is the number of time lags. If $s$ is large enough, the residual $v_k^j$ can be considered as white noise. Thus, the estimate of the latent variable $t_k^j$ using the dynamic inner model is

$$\begin{aligned}\hat{t}_k^j &= x_{k-1}^T w^j \beta_1^j + \cdots + x_{k-s}^T w^j \beta_s^j \\ &= [x_{k-1}^T, \ldots, x_{k-s}^T](\beta^j \otimes w^j)\end{aligned} \tag{6}$$

where $\beta^j = [\beta_1^j, \ldots, \beta_s^j]^T$ and $\|\beta^j\| = 1$, and $\otimes$ is Kronecker product. It can be shown that minimizing the mean square error and maximizing the covariance between$t_k^j$ and $\hat{t}_k^j$ is equivalent [28]. Hence the objective function of extracting the DLV is to maximize

$$\frac{1}{N-s}\sum_{k=s+1}^{N}(x_{k-1}^T w^j)^T[x_{k-1}^T, \ldots, x_{k-s}^T](\beta^j \otimes w^j) \tag{7}$$

where $N$ is the number of samples in $\mathcal{T}$.

Assume that the time lag, s, is large enough for fitting the DLV model. Normally, it is reasonable to believe that not all those past data make well contribution to the modeling. Therefore, we add a sparsity regularization term to the cost function (7) to prevent over-fitting.

$$\frac{1}{N-s}\sum_{k=s+1}^{N}(x_{k-1}^T w^j)^T[x_{k-1}^T, \ldots, x_{k-s}^T](\beta^j \otimes w^j) - \lambda\|\beta^j\| \tag{8}$$

where $\lambda$ is the penalty factor and $\|\cdot\|_1$ is L1 norm. Recall that $S_i(a_i, b_i) = \{x_{a_i}, x_{a_i+1}, \ldots, x_{b_i}\}$, we denote $X = [x_1, x_2, \ldots, x_{n_i}]^T$ as the data matrix of $S_i$, $n_i = b_i - a_i + 1$, and form the following sub-matrix from $X$:

$$X_\iota = [x_\iota, x_{\iota+1}, \ldots, x_{\iota+n_i-s-1}]^T$$

where $\iota = 1, 2, \ldots, s+1$. Collecting $X_\iota$ into

$$Z_s = [X_1, X_2, \ldots, X_s]$$

then the objective function of extracting DLV in subsequence $S_i$ can be shown as follow:

$$\begin{aligned}&\max_{w^j,\beta^j} J = (X_{s+1}w^j)^T Z_s(\beta^j \otimes w^j) - \lambda\|\beta^j\| \\ &s.t.\ \|\beta^j\| = 1, \quad \|w^j\| = 1\end{aligned} \tag{9}$$

The above optimization problem can be solved iteratively and the penalty factor $\lambda$ can be found by cross-validation. Finally, the proposed L1 regularized DiPCA algorithm is listed in Algorithm 1. The algorithm is a modification of the DiPCA algorithm of [28]. For brevity only the main steps are given.

### C. DYNAMIC PREDICTION ACCURACY

The static PCA based segmentation algorithms using objective function built on the Hotelling $T^2$ and Q statistics of PCA work well in the tasks of finding the best piece linear representation of time series data [22]. However, those methods are not suitable for detecting change in auto-correlation structure because the indices are insensitive to these changes. In this section, the cost function and the global objective function are defined based on the reconstruction accuracy of $t_k$, and we will also show that the reconstruction accuracy can be seen as the prediction accuracy.

Collecting $t^j$, $p^j$ and $w^j$ into $T$, $P$ and $W$ respectively, we have

$$\begin{aligned}T &= [t^1, t^2, \ldots, t^l] \\ P &= [p^1, p^2, \ldots, p^l] \\ W &= [w^1, w^2, \ldots, w^l]\end{aligned}$$

**Algorithm 1** DiPCA Algorithm

**function** dipca($X$, $s$, $l$)
  $X^1 \leftarrow$ Standardize $X$
  **for** $j = 1 \rightarrow l$ **do**
    **for** $\lambda$ in $\{0.005, 0.01, 0.02, \ldots\}$ **do**
      $w^j \leftarrow$ a random unit vector
      **while** $w^j$ *is not convergence* **do**
        $t^j \leftarrow X^j w^j$
        $t^j_\iota \leftarrow$ *form* $t^j$ *for* $\iota = 1, \ldots, s+1$
        $\beta^j \leftarrow [t^j_1, \ldots, t^j_s]^T t^j_{s+1}$
        $w^j \leftarrow \sum_{i=1}^{s} \beta^j_i (X^T_{s+1} t^j_i + X^T_\iota t^j_{s+1})$
        Normalize $\beta^j$ and $w^j$
      **end while**
    **end for**
    Choose $\lambda$ and $w^j$ with maximal $J$
    $p^j \leftarrow (X^j)^T t^j / (t^j)^T t^j$
    $X^{j+1} \leftarrow X^j - t^j (p^j)^T$
  **end for**
**end function**

Forming $T_\iota$ from $T$ just as forming $X_\iota$ from $X$, $\iota = 1, 2, \ldots, s+1$, and then the VAR model of DLV in Eq. (2) can be built as

$$\begin{aligned} T_{s+1} &= T_1\Theta_s + \cdots + T_s\Theta_1 + V \\ &= \bar{T}\Theta + V \end{aligned} \tag{10}$$

where $\bar{T} = [T_1, T_2, \ldots, T_s]$ and $\Theta = [\Theta_s^T, \Theta_{s-1}^T, \ldots, \Theta_1^T]$. The parameter $\Theta$ can be estimated using the least squares method:

$$\hat{\Theta} = (\bar{T}^T\bar{T})^{-1}\bar{T}^T T_{s+1} \tag{11}$$

Once $\hat{\Theta}$ is obtained, the estimate of $T_{s+1}$ is

$$\hat{T}_{s+1} = \bar{T}\hat{\Theta} \tag{12}$$

The above equation shows that the reconstruction is based on the auto-regressive equation which calculates the current data by its past-s-step data. Therefore the reconstruction can be seen as a prediction. And the prediction accuracy of $T_{s+1}$ can be measured by cosine similarity:

$$\mathcal{A} = \cos < T_{s+1}, \hat{T}_{s+1} > = \frac{T_{s+1} \cdot \hat{T}_s + 1}{\|T_{s+1}\|_F \|\hat{T}_s + 1\|_F} \tag{13}$$

where $\cdot$ is the inner product and $\|\cdot\|_F$ is the Frobenius norm.

Given a model trained using data $X$ and a subsequence $Y$, the latent score can be calculated as

$$T^Y = YW(P^T W)^{-1} \tag{14}$$

where $X$, $P$ are trained using $X$. Form $T^Y$ using $T^Y_1, T^Y_2, \ldots, T^Y_{s+1}$ in the same way as $X_\iota$. Then the estimate of $T^Y_{s+1}$ is

$$\hat{T}^Y_{s+1} = \sum_{\iota=1}^{s} T^Y_\iota \Theta_{s+1-\iota} \tag{15}$$

The prediction accuracy is

$$\mathcal{A} = \frac{T^Y_{s+1} \cdot \hat{T}^Y_{s+1}}{\|T^Y_{s+1}\|_F \|\hat{T}^Y_{s+1}\|_F} \tag{16}$$

Based on the prediction accuracy, the global objective function can be constructed by summarizing the dynamic predictability measure $\mathcal{A}$ presented in Eq. (12). The goal of the segmentation algorithm is to find those partitioning points which make each subsequence best reconstructed by the trained VAR model. Recall that $\mathcal{T} = \{S_1, S_2, \ldots, S_K\}$, for each subsequence $S_i$. $\Theta^i = [\Theta^i_1, \ldots, \Theta^i_s]$ are parameters of the trained model and then $\hat{T}^i_{s+1}$, the prediction of $T^i_{s+1}$ can be calculated. The prediction accuracy is

$$\mathcal{A}^i = \frac{T^i_{s+1} \cdot \hat{T}^i_{s+1}}{\|T^i_{s+1}\|_F \|\hat{T}^i_{s+1}\|_F} \tag{17}$$

Review Eq. (1) in section II A, the global objective function is

$$\max G = \frac{1}{K}\sum_{i=1}^{K} \mathcal{A}^i \tag{18}$$

### D. A GREEDY SEGMENTATION ALGORITHM

The problem presented in Eq. (18) is a combinatorial problem where the decision variables are the partitioning points. The globally optimal solution of this problem can be solved by dynamic programming [18]. However, it is often too computationally complex for an analytical approach [18], [31].

In this section, a greedy algorithm is presented for fitting a peace-wise stationary representation to the given industrial process data. The algorithm realizes the "bottom-up" approach [33] and can work in a very scalable way. The algorithm starts from the initial state where the time series are partitioned into the maximum allowed number of subsequences. Then, in each iteration, it merges a pair of adjacent subsequences so that by merging them the objective function is maximized. And the iteration will repeat until the termination condition is satisfied. For this goal, a merging cost must be designed. Corresponding to the objective function, for a pair of adjacent subsequences $S_i$ and $S_{i+1}$, the concatenation $S^{i\cup i+1}$ is used to train the parameter $\Theta^{i\cup i+1}$ and the prediction accuracy $\mathcal{A}^{i\cup i+1}$ is obtained.

$$\mathcal{A}^{i\cup i+1} = \frac{T^{i\cup i+1}_{s+1} \cdot \hat{T}^{i\cup i+1}_{s+1}}{\|T^{i\cup i+1}_{s+1}\|_F \|\hat{T}^{i\cup i+1}_{s+1}\|_F} \tag{19}$$

This is the cost of merging subsequences $i$ and $i+1$. If $\mathcal{A}^{i\cup i+1}$ is the largest among $i = 1, 2, \ldots, K$, then $S_i$ and $S_{i+1}$ should be merged, where $K$ is the number of subsequences. In case that multiple pairs have the same largest merging cost, one pair is chosen arbitrarily, e.g. the pair with the smallest indices. In this case, other policy may be adopted, i.e. to choose the pair having the shortest merged length. The segmentation algorithm procedure is listed in Algorithm 2.

*Remark 1:* The algorithm listed in Algorithm 2 terminates when the minimal number of subsequences has been reached.

**Algorithm 2** Segmentation Algorithm

**function** segment($\mathcal{T}$,$N_{\text{target}}$,*len*)
  *//*$N_{\text{target}}$ : *Final number of subsequences*
  *//len* : *Minimum allowable length*
  Partition $\mathcal{T}$ into initial subsequences TS
  Determine the time-lag $s$
  $n \leftarrow$ number of initial subsequences
  **while** $n > N_{\text{target}}$ **do** *//termination flag*
    **for** $i = 1$ to $n - 1$ **do**
      Calculate the merging cost $\mathcal{A}$(TS($i$),TS($i+1$))
    **end for**
    idx $\leftarrow$ The index with the maximum merging cost
    Merge TS(idx) and TS(idx+1)
    $n \leftarrow n - 1$
  **end while**
  *//Filter out invalid subsequences*
  **for** Each subsequence whose length $<$ *len* **do**
    Merge the subsequence with one of its neighbors having the larger merging cost
  **end for**
  **return** TS
**end function**

This is indeed a very usual choice of the terminating condition in practice, though it is not the only one. Other terminating condition such as the PETE [8] method can be applied when the computational speed is not a concern.

*Remark 2:* The merging operation in the proposed algorithm is subjected to the problem of data unbalance. This is because the merging prefers long subsequences so that the shorter subsequence of the neighbor of a long subsequence may be deprived the chance of merging at all. This results of an undesirably large number of small subsequences. To overcome this problem, when calculating the merging cost in Eq. (19) the longer one between the pair of adjacent subsequences is always chosen for training the parameters $W^{long}$, $P^{long}$ and $\Theta^{long}$ to predict the shorter one. Therefore, the prediction accuracy for merging cost is

$$\mathcal{A} = \frac{T_{s+1}^{short} \cdot \hat{T}_{s+1}^{short}}{\|T_{s+1}^{short}\|_F \|\hat{T}_{s+1}^{short}\|_F} \tag{20}$$

*Remark 3:* The algorithm is greedy as it always chooses to combine the two subsequences with maximum merging cost. Therefore, the algorithm only finds the local optimal segmentation position, which is not necessary to be global optimal.

*Remark 4:* The post-processing, which is optional, is used to filter out all the subsequences whose length is smaller than the given minimal allowed length which should be determined depending on the application.

*Remark 5:* Assuming a fix number of iterations in extracting each latent variable denoting as $\iota$, the complexity of evaluating the DiPCA based cost function is $O(lm\iota s)$. Introducing L1 regularization affects the algorithm only on the lower bound of complexity as we shall see in the following section that L1 regularization can deactivate redundant latent variables. The complexity of the bottom-up segmentation algorithm is $O(\bar{L}n)$ where $\bar{L}$ is the average segment length of the target subsequence [33]. The overall complexity is $O(nlm\iota s/N_{\text{target}})$.

## III. SIMULATION AND MODEL PARAMETERS

The proposed segmentation algorithm includes several parameters which must be given. For the convenience of discussion, the algorithm is first tested using the simulated data.

### A. SIMULATION SETUP

The data are generated from the following simulation model

$$t_k^{(i)} = \sum_{j=1}^{3} A_j^{(i)} t_{k-j} + v_k^{(i)}$$

where the superscript $i = 1, 2, 3, 4$ represents the $i$th model and $j = 1, 2, 3$ represents the coefficients of past-j-step. $v_k^{(i)} \sim N(0, 1^2)$ is a 3-dimensional column vector. These coefficients are set as follow.

$$A_1^{(1)} = \begin{bmatrix} 0.1395 & 0.0958 & 0.0256 \\ 0.0248 & 0.1307 & 0.0466 \\ 0.0327 & 0.1005 & 0.0431 \end{bmatrix}$$

$$A_2^{(1)} = \begin{bmatrix} 0.1488 & 0.0194 & 0.0364 \\ 0.0002 & 0.0067 & 0.0440 \\ 0.0622 & 0.0410 & 0.1628 \end{bmatrix}$$

$$A_3^{(1)} = \begin{bmatrix} 0.0674 & 0.0126 & 0.1594 \\ 0.1015 & 0.0967 & 0.1291 \\ 0.0638 & 0.1064 & 0.1193 \end{bmatrix}$$

$$A_j^{(2)} = -A_j^{(1)}, \quad for\ j = 1, 2, 3$$

$$A_1^{(3)} = \begin{bmatrix} 0.1132 & 0.0005 & 0.1198 \\ 0.0531 & 0.1595 & 0.0756 \\ 0.1170 & 0.0114 & 0.0388 \end{bmatrix}$$

$$A_2^{(3)} = \begin{bmatrix} 0.0205 & 0.0239 & 0.0843 \\ 0.0580 & 0.1317 & 0.1511 \\ 0.1158 & 0.1260 & 0.0573 \end{bmatrix}$$

$$A_3^{(3)} = \begin{bmatrix} 0.1273 & 0.0933 & 0.1589 \\ 0.1024 & 0.0429 & 0.0385 \\ 0.1128 & 0.1599 & 0.0853 \end{bmatrix}$$

$$A_j^{(4)} = -A_j^{(3)}, \quad for\ j = 1, 2, 3$$

For each model, 1000 samples are generated for segmentation respectively, in other words $t^{(i)} = [t_1^{(i)}, t_2^{(i)}, \ldots, t_{1000}^{(i)}]$. Concatenate $t^{(1)}$, $t^{(2)}$, $t^{(3)}$ and $t^{(4)}$, and express as $t = [t^{(1)}, t^{(2)}, t^{(3)}, t^{(4)}]$, then a data set including 4 known subsequences is generated by the following formula

$$x_k = Ct_k + e_k$$

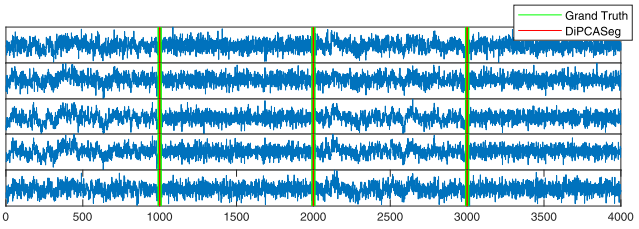



**FIGURE 2.** The segmentation result on the simulated data. The green lines are the true segmentation locations and the red lines are found by the proposed algorithm using the merging cost defined in Eq. (20).

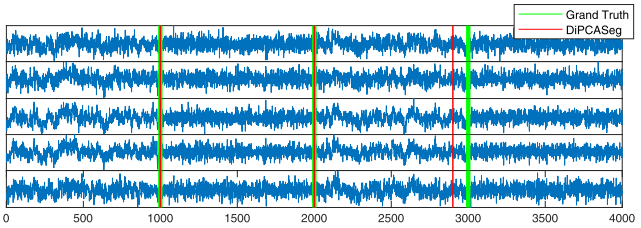



**FIGURE 3.** The data unbalance problem causes poor segmentation performance. The green lines are the true partitioning locations and the red lines are partitioning locations found by the algorithm using the merging cost defined in Eq. (19).

where $e_k \sim N(0, 0.1^2)$ and

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

Review that $a_{i+1} = b_i + 1$ are two adjacent partitioning points, so in order to express simply, $a_{i+1}$ and $b_i$ are regarded as one segmentation point. With the above configuration, the ground truth of the segmentation points are at {1000,2000,3000}. In the following simulation experiments, the parameter *len* is fixed to 50.

### B. RESULTS ON SIMULATED DATA

Figure 2 shows the segmentation results on the simulated data. The green lines represent true segmentation points and the red lines are algorithm effect. Since the true number of subsequences is known, i.e. $K = 4$, the termination condition of the algorithm can set to $K = 4$. The time-lag $s$ and the number of DLVs $l$ are set to 3 and 3 respectively. The L1 regularization coefficient $\lambda$ is 0.03. The determination of those parameters will be discussed later.

To illustrate the problem of data unbalance mentioned in Remark 2, the segmentation algorithm is performed according to the merging cost defined in Eq. (19). The segmentation result is shown in Figure 3, where the partitioning locations found by the algorithm are at {1000, 2000, 2900}. Obviously, the last location is incorrect. After changing the merging cost function to Eq.(20), the segmentation results are shown in Figure 2, where all the ground truth segmentation positions, {1000, 2000, 3000}, are correctly located.

The proposed method is compared with the other relevant segmentation methods including the PCA based method [23], the DPCA based method [3] and the fuzzy probabilistic PCA based method [9]. There are plenty of other developments in this field. One reason to choose those methods for comparison is that those and the proposed method are all in the same category of generalized piecewise linear approximation approach. Another reason is that the main contribution of this work lies on the way of measuring the approximation error. The methods for comparison all use the same bottom-up backend algorithm and the other settings can be made the same. This helps one to analyze the cause of performance difference and more clearly demonstrate the contribution of this work.

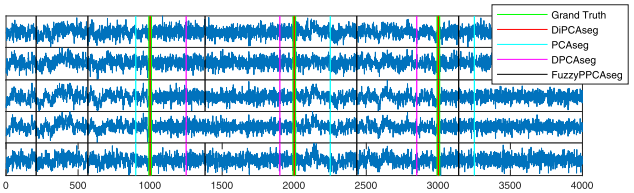



**FIGURE 4.** Comparison studies on the synthetic dataset.

The segmentation results obtained on the synthetic dataset are shown in Figure 4, where the methods for comparison are abbreviated as DiPCAseg, PCAseg, DPCAseg and FuzzyPPCAseg respectively. The green lines represent the ground truth. Note that for the FuzzyPPCAseg algorithm [34], the final number of segments is determined by the threshold for merging similar clusters. And the subsequences whose lengths are smaller than the predefined minimal allowed length are also merged. As we can see, the DiPCAseg outperforms all the rest methods in this case. This is because the dataset is specifically designed to demonstrate the strength of the proposed algorithm which is more sensitive in detecting the change of dynamics.

### C. DISCUSSIONS ON MODEL PARAMETERS

The proposed segmentation algorithm includes the following parameters which must be given:

- The number of DLVs, $l$: Similar to PCA, the number of DLVs indicates the amount of information to keep in the model. A large number of DLVs will lead to relatively less information loss and weak dimension reduction, and vice versa.
- The order of time lag, $s$: Recall in Eq. (2), the time-lag s refers to how long the past data will be considered having correlation with the current data. A moderate selection of s can better capture the dynamic correlation.
- The regularization coefficient, $\lambda$. The L1 regularization can suppress the non-zero weights, reduce the complexity of the model and increase the robustness of the algorithm. A larger $\lambda$ corresponds to a simpler model.
- The number of final subsequences, $N_{\text{target}}$. This parameter corresponds to the termination condition of the

segmentation algorithm. It is often unknown in practical tasks.
- The length of the initial subsequence, *len*. This parameter defines the initial subsequence length, which is dependent on the nature of data and application requirements.

In general, the first 3 parameters are DiPCA related and those can be determined by cross-validation [28]. The rest parameters are application dependent and those can be determined by generic approaches. In the following the effects of parameterization on the algorithm will be discussed.

#### 1) ON THE SELECTION OF *l*

In PCA algorithm, the number of principal components determines the information loss after reconstruction using the latent variables. This principle is still applicable in DiPCA algorithm [28]. The difference is that the number of DLVs is no longer determined by covariance, but by auto-covariance, which refers to the covariance between the actual sample and its prediction by the VAR model. Assuming the other parameters, i.e. $s$ and $\lambda$, are determined, then $l$ can be determined by a constructional approach which stops when 90% of auto-covariance are captured by the first $l$ dynamic latent variables. In the case that it is not possible to reach the desired percentage the construction procedure should shop.

#### 2) ON THE SELECTION OF *s*

The authors of [28] introduce a cross-validation procedure to simultaneously determine the optimal value of the dynamic order $s$ and the number of DLVs $l$. This is achieved by using a validation dataset to evaluate the prediction error of the model built on the training data. In segmentation problem, the data to be partitioned may not be homogeneous. Therefore it is not possible to split the training data and the validation data. For this problem, a generic procedure is used. According to Eq. (9), the fitting accuracy of the DLV model is expected to increase as s using the same training data. This is associated with increasing objective $J$ against $s$. However, for real world industrial data, it is reasonable to assume that the auto-correlation dependence decays with the time interval. Further increasing $s$ will contribute less to $J$. Hence the optimal value of $s$ is assigned to the point where the rate of increasing $J$ starts to slow down, as it best balances the models expressing power and complexity. It is also worth pointing out that in general the dynamic order $s$ is different for different DLVs. For the task of segmentation, it is sufficient to consider only the most dominant DLVs.

To illustrate this point, a subsequence is randomly chosen from the simulated data, e.g. $\{x_{551}, \ldots, x_{600}\}$. Then, by increasing $s$ from 1 to 10, the value of objective, $J$, denoted in Eq. (9) against $s$ for all the available DLVs is plotted in Figure 5. It can be seen that the first DLV dominates the contribution to $J$. And an obvious elbow point can be observed at $s = 3$, which is associated with the point of decay rate slowing down. Though the elbow point is not obvious for the other DLVs. Therefore, only the first DLV is considered and $s = 3$ is chosen as the optimal choice for the order of model, which is identical to the true order of the simulation model. Next, some subsequences are randomly sampled from all the available subsequences and find out their optimal $s$ respectively using the above method. Then, the largest one is chosen as the final optimal value for $s$.

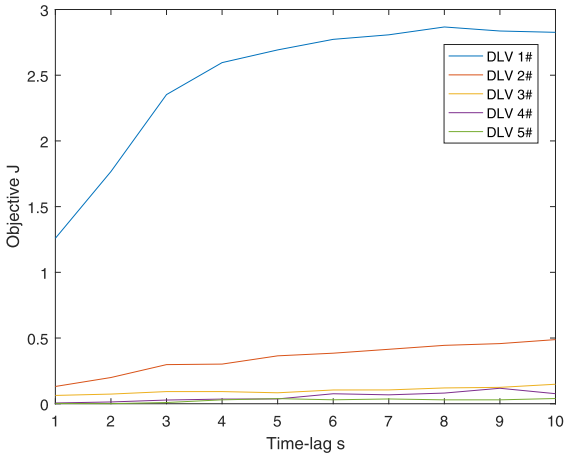



**FIGURE 5.** The time lag *s* versus the objective *J* of different DLVs.

**TABLE 1.** Reducing nonzero weights by L1 regularization.

| | |
|---|---|
| $\lambda = 0.000$ | $\beta^1 = (0.569 \quad -0.938 \quad 0.337 \quad 0.904 \quad 0.874)^T$ |
| $\lambda = 0.005$ | $\beta^1 = (0.569 \quad -0.938 \quad 0.337 \quad 0.904 \quad 0.000)^T$ |
| $\lambda = 0.010$ | $\beta^1 = (0.569 \quad -0.938 \quad 0.337 \quad 0.000 \quad 0.000)^T$ |
| $\lambda = 0.020$ | $\beta^1 = (0.569 \quad -0.938 \quad 0.000 \quad 0.000 \quad 0.000)^T$ |

#### 3) THE EFFECTS OF L1 REGULARIZATION

First, to illustrate the effect of L1 regularization on DiPCA algorithm, the algorithm parameters are set to $s = 3$, $l = 5$ and let $\lambda$ having values 0, 0.005, 0.01, 0.02 respectively. Table 1 shows the weights of the first dynamic latent vector, $\beta^1$. It can be can observed that the number of nonzero terms decreases as the value of $\lambda$. In fact, even when the number of DLVs, $l$, is unnecessarily large, i.e. $l = 5$, introducing L1 regularization can successfully deactivate those redundant latent variables by zeroing their weights.

Secondly, the overall effect of L1 regularization on the segmentation algorithm is reducing the variance of segmentation locations found by the algorithm. The proposed segmentation algorithm starts from an initial state where each subsequence has the minimum allowed length. The output variance arises if there is under fitting caused by possible insufficient initial number of samples in each subsequence. To see this, the segmentation experiment is repeated 100 times on the same simulated data with $\lambda = 0$ and $\lambda = 0.02$ respectively. Figure 6 shows the three partitioning locations found by each run of the algorithm. The red lines represent the results obtained with and the green lines represent the results obtained with $\lambda$. In the first case, the standard deviation of the

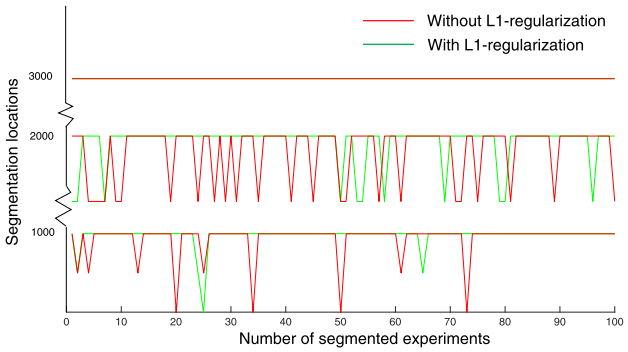



**FIGURE 6.** The effects of variance reduction with L1 regularization.

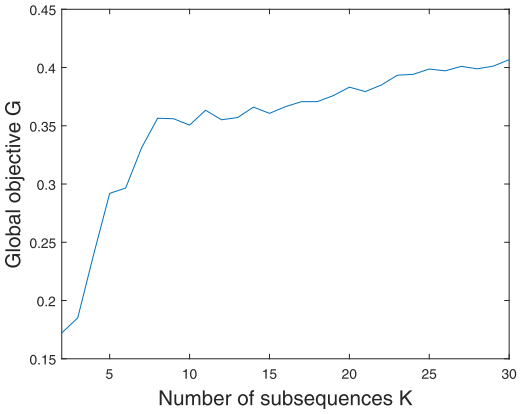



**FIGURE 7.** The number of subsequences versus global objective.

three locations found by the algorithm are 11.04, 10.73 and 0 respectively. And in the second case, the standard deviation of the locations are 6.53, 7.86 and 0 respectively. Obviously, the L1 regularization indeed reduces the variance of the algorithm outputs. Note that when $\lambda = 0.02$ is relatively large, we set $T^Y = YW(P^T W + \epsilon I_l)^{-1}$ in Eq. (14) to prevent problems with singular matrices, where $\epsilon$ is a small constant, e.g. 0.1.

#### 4) ON THE SELECTION OF $N_{\text{target}}$

In practical tasks, the true number of final segments is rarely known in advance. It is often provided considering the amount of the available data, the goal of time series data analysis, and the temporal granularity of the data, etc. There are also more sophisticated methods such as PETE [8] which can be applied in expecting the number subsequences.

In this work, a generic approach is used to the determination of an appropriate number of subsequences. The global objective $G$ versus the number of subsequences $K$ is plotted to find the elbow point. Figure 7 plots the results using the simulated data as an example. An obvious elbow point is located at $K = 8$. Since the true number of subsequences and their locations are known, there are 4 incorrect subsequences under this selection of $K$. Figure 8 shows the segmentation results. It can be seen that the three true partitioning locations are all accurately located. There are also four falsely found partitioning locations, which are labeled using the purple lines. However those false locations are all near the beginning of time series between $x_1 \ldots x_{200}$, and the length of those subsequence are equal to the initial subsequences. Therefore, those false locations can be easily filtered out by setting a minimal allowed length of subsequence.

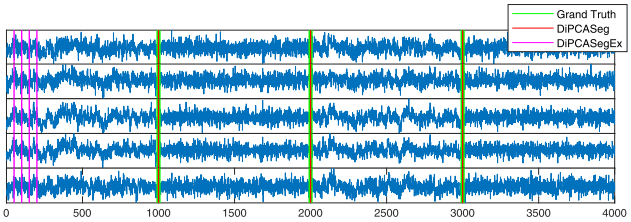



**FIGURE 8.** The purple lines are the false partitioning locations (marked as DiPCASegEx). The green lines are the true partitioning locations. The red lines are partitioning locations found by the algorithm.

## IV. EXPERIMENTS ON INDUSTRIAL PROCESS DATA

The results shown in Figure 4 is obtained on the synthetic dataset to demonstrate the strength of the proposed algorithm. To evaluate the performance more thoroughly, the proposed method is compared with PCAseg, DPCAseg and FuzzyPPCAseg on various datasets with more diversified characteristics.

### A. TENNESSEE EASTMAN PROCESS

The proposed segmentation method is tested using the Tennessee Eastman Process (TEP) simulated data [35]. The TEP has been widely used by the process monitoring community for comparing different approaches [36], [37]. The simulated TEP includes five unit operations: a reactor, condenser, compressor, separator and stripper. There are 12 manipulated variables (XMV1..XMV11), 22 process variables (XMEV1..XMEV22), and 19 composition measurements (XMEVS23..XMEVS41). Each simulated time series data represents a 48 hour-run with sampling interval of 3 minutes. The time series dataset is constructed using the 22 process variables and the 12 manipulated variables.

The TEP simulation contains 21 preprogrammed faults, and those faults are introduced after the 8th hour. The test datasets corresponding to faults IDV5 and IDV10 are used. The proposed DiPCAseg algorithm was compared with the PCAseg, DPCAseg and FuzzyPPCAseg on the datasets. The initial length of each subsequence is set to 30, the final number of segments to 4, the number of DLVs to 13 and the order of time lag to 3. For FuzzyPPCAseg, the final number of segments is adaptively determined by the algorithm, and the final subsequences whose length is smaller or equal to 30 are merged. The final results are shown in Figure 9 and Figure 10. Since the preprogrammed faults are introduced after the 8th hour, we can only confidently set one ground truth segmentation position to 160-165, indicated by green lines in the figures.

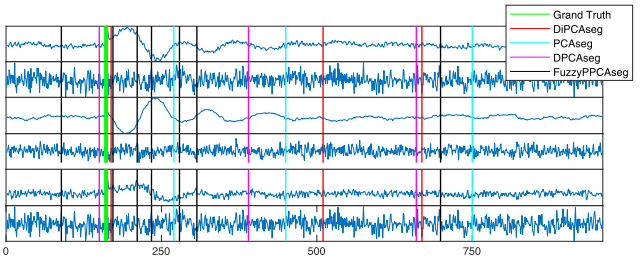



**FIGURE 9.** Comparison on the TEP preprogrammed fault IDV(5) data. The displayed variables from the top to the bottom are product separator temperature (XMEAS11), product separator level (XMEAS12), product separator pressure (XMEAS13), product separator underflow (XMEAS14), separator cooling water outlet temperature (XMEAS22) and separator pot liquid flow (XMV7).

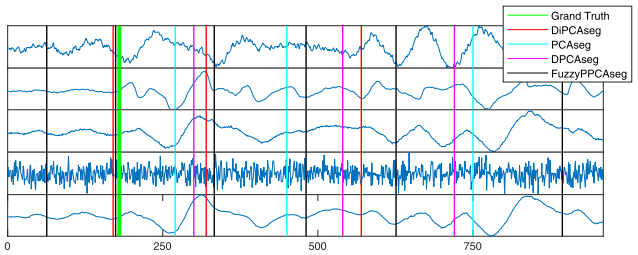



**FIGURE 10.** Comparison on the TEP preprogrammed fault IDV(10) data. The displayed variables from the top to the bottom are stripper pressure (XMEAS16), stripper temperature (XMEAS18), stripper stream flow (XMEAS19), stripper liquid product flow (XMV8), stripper steam valve (XMV9).

The fault IDV5 denotes a step change in the condenser cooling water inlet temperature. The segmentation results are shown in Figure 9. Since there is no direct measurement of the cooling water temperature, the process variables related to the separator which is the downstream unit operation of the condenser are chosen to visualize the segmentation results. In the figure, those variables are product separator temperature (XMEAS11), product separator level (XMEAS12), product separator pressure (XMEAS13), product separator underflow (XMEAS14), separator cooling water outlet temperature (XMEAS22) and separator pot liquid flow (XMV7). The step change in the condenser cooling water temperature can be indirectly observed in its effects on the product separator temperature. Among the 4 methods for comparison, the DiPCAseg and the FuzzyPPCAseg accurately locate the first segmentation position. While the DPCAseg method shows little discrepancy and the PCAseg based on the $T^2$ statistics fail to locate it. The rest segmentation locations found by the different algorithms are diverse, and there is no ground truth for evaluation. There is one expectance around 660-670 where both DiPCAseg and DPCAseg suggest a segmentation, which may be explained as the convergence point of the effects introduced to the system by the step change in the condenser cooling water inlet temperature.

For the fault IDV10, random variation is introduce to the feed temperature of component C. Since there is no direct measurement of the temperature of C feed stream, some measurements and manipulated variables of the stripper are chosen to indicate the effects of the fault. Those include stripper pressure (XMEAS16), stripper temperature (XMEAS18), stripper stream flow (XMEAS19), stripper liquid product flow (XMV8), stripper steam valve (XMV9). Figure 10 shows the results. It can be observed that both DiPCAseg and FuzzyPPCAseg accurately detect the change near 180, while the DPCAseg and PCAseg fail to detect it.

### B. SHARI EVAPORATION PROCESS

We applied the proposed segmentation algorithm on an industrial process data set collected from the evaporation plant of Shanxi Huaxing Alumina Refinery Industry (SHARI). The evaporation plant at SHARI uses the 6 effects counter current evaporation process which including 6 tube falling film evaporators, 4 self-evaporators and 6 condensation tanks. A brief diagram of the process is shown in Figure 11. The evaporators of alumina evaporation process are heated by steam. The slurry stream is fed into evaporators I, II and III respectively. In order to make full use of the heat source, the secondary steam of the evaporator I is fed into the II evaporator, and the secondary steam of the evaporator II is fed into the evaporator III, i.e., the secondary steam of each effect of evaporator enters the next-effect evaporator as heating. At the same time, full using of the secondary steam heat source of the self-evaporator, the secondary steam from the 1st $\sim$ 4th self-evaporator is used as the heating source for the evaporators III $\sim$ VI, and the secondary steam of the last evaporator VI enters the condenser. In addition, the condensate water of evaporator I is fed into the 1# condenser, then the secondary steam regenerated from the 1# condenser is re-used as the heat source to the evaporator I. Similarly, the condensate water of evaporators II $\sim$ VI inflows the 2# $\sim$ 6# condensers and the secondary steam is fed into the evaporators II $\sim$ VI respectively.

From the description of the evaporation process, it can be seen that the process variables, such as the pressure, the temperature and the level are highly correlated. The slurry levels of the five condensers are chosen as an example to demonstrate the performance of the proposed algorithm. The condenser level is the key process variable of the evaporation process and it is also less volatile than other variables such as pressure. The data set includes 4162 samples collected from the plant. For this case, the true segmentation and the number of subsequences are not known. To evaluate the performance of the proposed segmentation method, we invite experienced operators and process engineers to mark the segmentation points by hand. For the current data, the confirmed segmentation points are at {1220, 2535, 2815, 3020, 3120, 3638, 3880}.

For the SHARI data set, the segmentation algorithm parameters are: $s = 3$, $l = 3$, $\lambda = 0.005$, $N_{\text{target}} = 10$, and $len = 30$. The segmentation results are shown in Figure 12-(a) where the red and green lines represent the

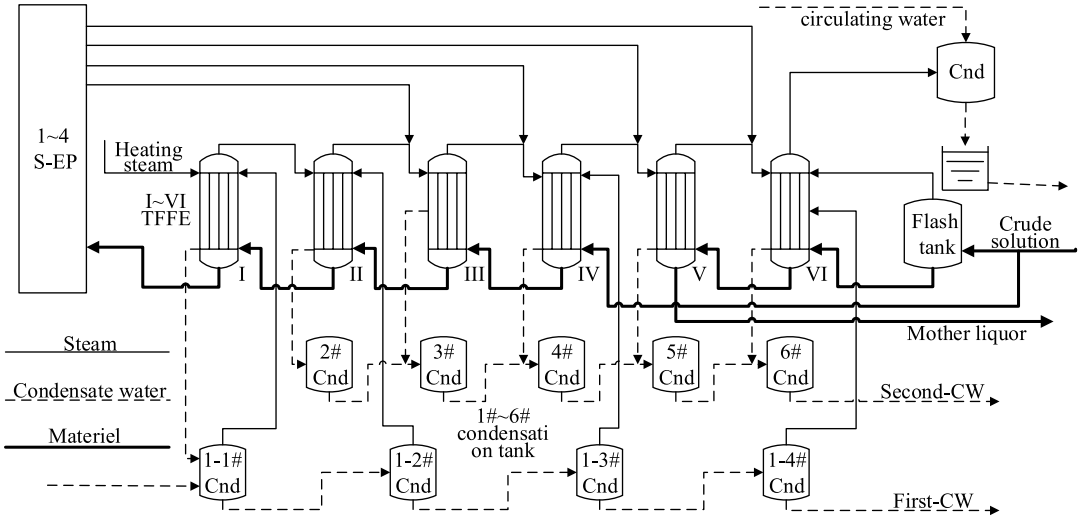
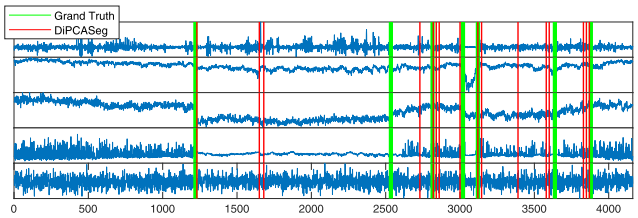
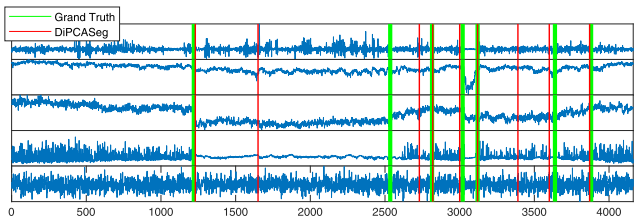



**FIGURE 11.** Brief process diagram of the six effects evaporation process of SHARI.




(a) Before merging invalid subsequences




(b) After merging invalid subsequences

**FIGURE 12.** Segmentation results of the proposed method on the SHARI data. The green lines are the partitioning locations confirmed by experts. The red lines are partitioning locations found by the algorithm.

segmentation locations found by the proposed algorithm and experts respectively. It can be seen that the algorithm found 17 partitioning locations in total, i.e. 18 subsequences. Note that some of those subsequences have very small length. Those data are related to the very short period of transient period of the process or simply caused by noise. Since the preset the minimal allowed subsequence length is $len = 30$, those subsequences of length shorter than n are filtered out as invalid subsequences. Those are forced to merge with one of their neighbour subsequences. For this goal, the merging cost in Eq. (20) can be used to choose the pair with smaller merging cost to form a new subsequence. After merging these short subsequences, the final set of segmentation points are: {1230, 1650, 2730, 2820, 3000, 3120, 3390, 3600, 3870}, and those are shown in Figure 12-(b).

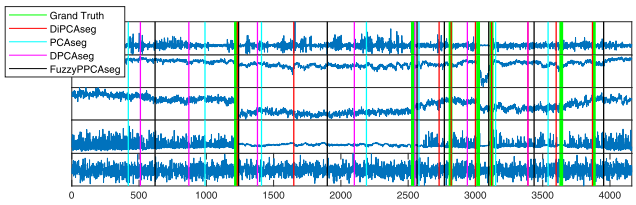



**FIGURE 13.** Comparison on the SHARI data.

**TABLE 2.** Comparison of the four methods.

| | DiPCAseg | PCAseg | DPCAseg | FuzzyPPCAseg |
|---|---|---|---|---|
| Precision | **66.66%** | 33.33% | 33.33% | 55.56% |

To evaluate the results of the proposed algorithm, Precision ($P$) is calculated as the $P = TP/(TP + FP)$, where $TP$ is the number of true positive and $FP$ is the number of false positive. It is worth noting that the expert confirmed segmentation positions should not be considered being crisp. In the current context a position found by algorithm is counted as correct if it is within $\pm 80$ (about $\pm 2\%$ of the total length) of one of the confirmed locations. The proposed segemenation method is compared with PCAseg, DPCAseg and FuzzyPPCAseg methods, and the results are shown in Figure 13. It can be seen that DiPCAseg and FuzzyPPCAseg show remarkably higher precision than the rest methods. DiPCAseg locates six out of the seven confirmed segmentation positions and FuzzyPPCAseg locates five out of the seven confirmed segmentation positions. There are a case where DiPCAseg correctly locates a segment but not by FuzzyPPCAseg, e.g. at 3800, and the other way round e.g. at 2535. The overall performance of each method is listed in Table 2.

## V. CONCLUSION

Industrial process time series are characterized by its inherent dynamic nature. This paper investigates the problem of partitioning multivariate industrial time series data into piecewise stationary subsequences. The proposed algorithm seeks

the segmentation plan which maximizes the dynamic predictability in each subsequence. The main contribution of this work is the design of the objective function measuring the dynamic predictability using the DiPCA modeling accuracy. Though the proposed method focuses on the dynamic predictability of industrial process data, it is flexible to be extended so as to include more segmentation criteria when the data of interest present other characteristics of the fields, e.g. cross correlation, periodicalness and trending. One limitation of the proposed algorithm is that it cannot guarantee the segmentation plan is global optimal due to the greedy approach it adopts. This might be alleviated by increasing the searching depth under some complexity constraint, or going future to the other end of the spectrum of the dynamic programming based solutions. To tackle the state explosion problem, a possible future direction of this study is to use approximate dynamic programming approach. Finally, the proposed method is implemented in the bottom-up segmentation algorithm and it has been thoroughly tested on simulated data and industrial process data, which shows encouraging results.

## REFERENCES

[1] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, Jul. 2019, pp. 2828–2837.
[2] H. Ren, M. Liu, X. Liao, L. Liang, Z. Ye, and Z. Li, "Anomaly detection in time series based on interval sets," *IEEJ Trans. Electr. Electron. Eng.*, vol. 13, no. 5, pp. 757–762, May 2018.
[3] L. Dobos and J. Abonyi, "On-line detection of homogeneous operation ranges by dynamic principal component analysis based time-series segmentation," *Chem. Eng. Sci.*, vol. 75, pp. 96–105, Jun. 2012.
[4] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 339–367, May 2017.
[5] G. Bontempi, S. Ben Taieb, and Y. A. Le Borgne, "Machine learning strategies for time series forecasting," in *Business Intelligence. eBISS* (Lecture Notes in Business Information Processing), vol. 138, M. A. Aufaure and E. Zimányi, Eds. Berlin, Germany: Springer, 2013.
[6] S. Lu and T. Chai, "Mesoscale particle size predictive model for operational optimal control of bauxite ore grinding process," *IEEE Trans. Ind. Informat.*, early access, 2020, doi: 10.1109/TII.2020.2967067.
[7] J. Yu and X. Yan, "Whole process monitoring based on unstable neuron output information in hidden layers of deep belief network," *IEEE Trans. Cybern.*, early access, 2020, doi: 10.1109/TCYB.2019.2948202.
[8] K. T. Vasko and H. T. T. Toivonen, "Estimating the number of segments in time series data using permutation tests," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 466–473.
[9] J. Abonyi, B. Feil, S. Nemeth, and P. Arva, "Modified Gath–Geva clustering for fuzzy segmentation of multivariate time-series," *Fuzzy Sets Syst.*, vol. 149, no. 1, pp. 39–56, Jan. 2005.
[10] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2001, pp. 289–296.
[11] E. Bingham, A. Gionis, N. Haiminen, H. Hiisilä, H. Mannila, and E. Terzi, "Segmentation and dimensionality reduction," in *Proc. 6th SIAM Int. Conf. Data Mining*, Apr. 2006, pp. 372–383.
[12] J. Zhao and L. Itti, "Decomposing time series with application to temporal segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–9.
[13] Z. Sun, X. Liu, and L. Wang, "A hybrid segmentation method for multivariate time series based on the dynamic factor model," *Stochastic Environ. Res. Risk Assessment*, vol. 31, no. 6, pp. 1291–1304, Aug. 2017.
[14] A. González-Vidal, P. Barnaghi, and A. F. Skarmeta, "BEATS: Blocks of eigenvalues algorithm for time series segmentation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 11, pp. 2051–2064, Nov. 2018.
[15] Y. Hu, P. Guan, P. Zhan, Y. Ding, and X. Li, "A novel segmentation and representation approach for streaming time series," *IEEE Access*, vol. 7, pp. 184423–184437, 2019.
[16] X. Xuan and K. Murphy, "Modeling changing dependency structure in multivariate time series," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2007, pp. 1055–1062.
[17] W.-H. Lee, J. Ortiz, B. Ko, and R. Lee, "Time series segmentation through automatic feature learning," 2018, *arXiv:1801.05394*. [Online]. Available: http://arxiv.org/abs/1801.05394
[18] D. Hallac, P. Nystrup, and S. Boyd, "Greedy Gaussian segmentation of multivariate time series," *Adv. Data Anal. Classification*, vol. 13, no. 3, pp. 727–751, 2019.
[19] H. Kim, H. K. Kim, M. Kim, J. Park, S. Cho, K. B. Im, and C. R. Ryu, "Representation learning for unsupervised heterogeneous multivariate time series segmentation and its application," *Comput. Ind. Eng.*, vol. 130, pp. 272–281, Apr. 2019.
[20] T. Hastie, R. Tibshirani, J. H. Friedman, and J. Franklin, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Berlin, Germany: Springer, 2009.
[21] M. J. Piovoso and K. A. Hoo, "Multivariate statistics for process control," *IEEE Control Syst.*, vol. 22, no. 5, pp. 8–9, Oct. 2002.
[22] J. Abonyi, B. Feil, S. Nemeth, and P. Arva, "Fuzzy clustering based segmentation of time-series," in *Advances in Intelligent Data Analysis V*, M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, Eds. Berlin, Germany: Springer, 2003, pp. 275–285.
[23] S. Spiegel, J. Gaebler, A. Lommatzsch, E. De Luca, and S. Albayrak, "Pattern recognition and classification for multivariate time series," in *Proc. 5th Int. Workshop Knowl. Discovery Sensor Data (SensorKDD)*, New York, NY, USA, 2011, pp. 34–42.
[24] H. Tanatavikorn and Y. Yamashita, "Batch process monitoring based on fuzzy segmentation of multivariate time-series," *J. Chem. Eng. Jpn.*, vol. 50, no. 1, pp. 53–63, 2017.
[25] A. Negiz and A. Cinar, "Monitoring of multivariable dynamic processes and sensor auditing," *J. Process Control*, vol. 8, no. 5, pp. 375–380, 1998.
[26] W. Ku, R. H. Storer, and C. Georgakis, "Disturbance detection and isolation by dynamic principal component analysis," *Chemometric Intell. Lab. Syst.*, vol. 30, no. 1, pp. 179–196, Nov. 1995.
[27] G. Li, B. Liu, S. J. Qin, and D. Zhou, "Dynamic latent variable modeling for statistical process monitoring," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 12886–12891, 2011.
[28] Y. Dong and S. J. Qin, "A novel dynamic PCA algorithm for dynamic data modeling and process monitoring," *J. Process Control*, vol. 67, pp. 1–11, Jul. 2018.
[29] W. J. Krzanowski, "Some exact percentage points of a statistic useful in analysis of variance and principal component analysis," *Technometrics*, vol. 21, no. 2, pp. 261–263, May 1979.
[30] A. Singhal and D. E. Seborg, "Matching patterns from historical data using PCA and distance similarity factors," in *Proc. Amer. Control Conf.*, vol. 2, Jun. 2001, pp. 1759–1764.
[31] L. Dobos and J. Abonyi, "Fisher information based time-series segmentation of streaming process data for monitoring and supporting on-line parameter estimation in energy systems," *Comput. Aided Chem. Eng.*, vol. 29, pp. 1844–1848, 2011.
[32] Z. Bankó and J. Abonyi, "Correlation based dynamic time warping of multivariate time series," *Expert Syst. Appl.*, vol. 39, no. 17, pp. 12814–12823, 2012.
[33] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," *Data Mining Time Databases*, vol. 57, pp. 1–21, Mar. 2003.
[34] J. Abonyi. (Apr. 2020). *Fuzzy Clustering Based Time-Series Segmentation. MATLAB Central File Exchange*. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/47180-fuzzy-clustering-based-time-series-segmentation
[35] A. Bathelt, N. L. Ricker, and M. Jelali, "Revision of the Tennessee Eastman process model," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 309–314, 2015.
[36] L. H. Chiang, E. L. Russell, and R. D. Braatz, "Tennessee Eastman process," in *Fault Detection and Diagnosis in Industrial Systems*, L. H. Chiang, E. L. Russell, and R. D. Braatz, Eds. London, U.K.: Springer, 2001, pp. 103–112.
[37] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *J. Process Control*, vol. 22, no. 9, pp. 1567–1581, Oct. 2012.

**SHAOWEN LU** (Member, IEEE) received the Ph.D. degree in electronic engineering from the Queen Mary University of London. He is currently a Professor with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University. His research interest includes industrial process modeling and simulation. His most recent research has been in the area of industrial time series modeling and analysis, working with both data driven and model driven methods.

**SHUYU HUANG** received the B.S. degree in automation from Northeastern University at Qinhuangdao, China, in 2016. He is currently a Research Student with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, China. His current research interests include time series analysis, data mining, and machine learning.