Research Article

An Improved Generalized Predictive Control in a Robust Dynamic Partial Least Square Framework

Jin Xin, Chi Qinghua, Liu Kangling, and Liang Jun

State Key Lab of Industrial Control Technology, Department of Control Science & Engineering, Zhejiang University, Hangzhou 310027, China

Correspondence should be addressed to Liang Jun; jliang@iipc.zju.edu.cn

Received 20 March 2015; Accepted 16 September 2015

Academic Editor: Qing Chang

Copyright © 2015 Jin Xin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To tackle the sensitivity to outliers in system identification, a new robust dynamic partial least squares (PLS) model based on an outliers detection method is proposed in this paper. An improved radial basis function network (RBFN) is adopted to construct the predictive model from inputs and outputs dataset, and a hidden Markov model (HMM) is applied to detect the outliers. After outliers are removed away, a more robust dynamic PLS model is obtained. In addition, an improved generalized predictive control (GPC) with the tuning weights under dynamic PLS framework is proposed to deal with the interaction which is caused by the model mismatch. The results of two simulations demonstrate the effectiveness of proposed method.

1. Introduction

In the past decades, PLS, a multivariable regression method, has been applied to many areas, such as quality prediction, process monitoring, and chemometrics [1]. PLS is not only able to extract principle component from both input and output datasets, but also able to determine the direction on which input and output data have the largest covariance [2]. Considering the advantage of dimension reduction and automatic decoupling, many researchers applied PLS to the modeling and control of dynamic systems. Kaspar and Ray [3, 4] proposed a dynamic PLS framework by utilizing the PLS loading matrices to construct precompensators and post-compensators. Chen et al. [5] proposed another dynamic PLS framework with ARX model. Laurí et al. [6] proposed a PLS-based model predictive control (MPC) relevant identification method.

There are many uncertain factors in industrial fields which make the clean data cannot be acquired directly from actual systems. It is well known that the standard algorithms for PLS regression (NIPALS and SIMPLS) [7] are very sensitive to outliers in the dataset. Outliers can distort the curve of the data. It may lead to undesirable results if standard PLS is directly applied in practical applications. To overcome this problem, several robust PLS methods are proposed in recent years. One of them is obtained by robustifying

the sample covariance matrix between the input and output datasets based on SIMPLS algorithm [8, 9]. Another is PLS calibration with outliers detection, which selects a subset of samples dataset randomly and gets the initial residual set and then detects and discards outliers in the subset [10]. These methods are not suitable for the dynamic PLS control framework because the samples are a time sequence series and cannot be selected randomly. Moreover, these methods only focus on the regression; they do not fit for dynamic modeling.

There is one way to get a robust dynamic PLS model. In contaminated data, several data preprocessing methods could be applied to detect outliers and eliminate the influence of them. Various methods for outliers detection have been proposed in the literature, such as statistics-based algorithms, machine learning-based algorithms, and neural network-based algorithms. In these methods, outliers are the point far away from the majority of the data; however this situation is not always right [11]. Outliers can be close to the major curve of data as well. In order to detect general outliers in the dataset which is used for modeling, the relationship between inputs and outputs should be known. Liu et al. [11, 12] proposed an autoregression structure RBFN to predict the outputs of the system and detected outliers with this predictive model combined with HMM. The limitation is

that the output of the RBFN is composed of the inputs and outputs of the process, but the prediction of system input is useless. Because outliers are usually contained in the output dataset, outliers in input dataset are commonly calculated by the controllers. Therefore, it is not useful to increase the dimension of the network weights to estimate the value of the inputs. In addition, Liu et al. [11] analyzed the performance of the algorithm with single-input single-output (SISO) system at low rate of outliers. There is no analysis with more complex situations such as multiple-input multiple-output (MIMO) systems and high rate of outliers.

On the other hand, many control schemes are applied under the dynamic PLS framework. Kaspar and Ray proposed a PID control scheme under this framework [3], and Chen et al. [5] designed multiloop adaptive PID controllers based on an modified decoupling PLS framework. Hu et al. [13, 14] proposed a multiloop internal model controller in the dynamic PLS framework and achieved better performance for disturbance rejection. Lü and Liang [15] proposed a multiloop constrained MPC scheme. Ideally, with the PLS decoupled structure, there should be no interaction between PLS components [3]. However, there are always some interactions observed due to the plan/model mismatch which is caused by contaminated data and filtering color noises incompletely.

Aiming at these limitations, a robust dynamic PLS modeling and improved GPC control scheme is proposed in this paper. A new RBFN structure is applied to predict the outputs of the system. With the difference between the prediction and the real data, a simple HMM algorithm is used to detect the outliers, and then they are replaced by the mean of clean data nearby. After outliers detection, dynamic PLS modeling would be more robust. Based on this model, a dynamic tuning weight sequence is introduced into the cost functions of improved GPC under the PLS latent space to reduce the interaction of different outputs which is caused by mismatch between the PLS model and real process.

The rest of paper is organized as follows: in Section 2, an improved RBFN training method and an HMM based outliers detection algorithm are presented. In Section 3, dynamic PLS modeling method is introduced and an improved GPC scheme is proposed. In Section 4, two examples are used to demonstrate the effectiveness and better control performance of the proposed method. In the last section, the conclusions are summarized.

2. Outliers Detection for Modeling Data

2.1. Improved Radial Basis Function Network. As a feedforward network, the RBFN neural network has three layers: input layer, hidden layer, and output layer. And, in many practical applications, Gaussian RBF is used in the hidden layer. It can approximate the linear and nonlinear functions with arbitrary precision in theory. When fitting the dynamic systems, several past input and output samples should be used as the input of the RBFN. To introduce the new structure of RBFN, 3 assumptions are firstly given as follows: (1) the system has m inputs and n outputs; (2) only the outputs of the system contain outliers; and (3) each input signal and

FIGURE 1: The structure of proposed RBFN.

each output signals have the same lags, respectively. With assumption (1), arbitrary input and output dimension system could be involved. Outliers in outputs are usually caused by some unmeasured reasons such as unknown sensor failures. Hence it is what we focus on (with assumption (2)). While it is easy to detect outliers in inputs, for the inputs can be calculated with set points and detected values by controllers. In order to simplify the analysis, assumption (3) is employed. Signal with different lags can be treated as a special case of that signal with the same lags. Past samples used as the inputs of the network have the form

$$\mathbf{O} = [\mathbf{x}(t-1), \mathbf{x}(t-2), \dots, \mathbf{x}(t-n_x), \mathbf{y}(t-1), \mathbf{y}(t-2), \dots, \mathbf{y}(t-n_y)]^T, \quad (1)$$

where $\mathbf{x}$ is an m -dimension vector and $\mathbf{y}$ is an n -dimension vector. $[\mathbf{x}(t-1), \mathbf{x}(t-2), \dots, \mathbf{x}(t-n_x)]$ denotes the input values from $t-1$ to $t-n_x$, $[\mathbf{y}(t-1), \mathbf{y}(t-2), \dots, \mathbf{y}(t-n_y)]$ denotes the output values from $t-1$ to $t-n_y$. n_x and n_y denote the lag of input and output, respectively.

The structure of the modified RBFN is shown in Figure 1. n_x past input values and n_y past output values enter the RBFN. The output of RBFN $\hat{\mathbf{y}}(t)$ denotes the predictive vector, which

is calculated by summing the product of the hidden layer outputs φ and the output layer weights ω . The superscripts φ and ω denote which input or output it belongs to, and the subscript denotes which hidden node it belongs to. $\varphi_{i_c}^i$ is computed by the hidden node center vector $\mathbf{C}_{i_c}^i$ with the Gaussian function, where $i = 1, \dots, n_x + n_y$ and $i_c = 1, \dots, C_i$. Then the output of the RBFN is

$$\hat{\mathbf{y}}(t) = \sum_{i=1}^{n_x+n_y} \sum_{i_c=1}^{C_i} \varphi_{i_c}^i \cdot \omega_{i_c}^i. \quad (2)$$

When $\mathbf{O}(i)$ is inputted into the RBFN, there are C_i hidden nodes for it to cluster. The outputs of hidden layers with Gaussian function RBFN for the inputs are

$$\varphi_{i_c}^i = \exp\left(\frac{\|\mathbf{O}(i) - \mathbf{C}_{i_c}^i\|^2}{-2\sigma_i^2}\right), \quad (3)$$

where $i = 1, \dots, n_x + n_y$, $i_c = 1, \dots, C_i$.

It is noted that $\mathbf{C}_{i_c}^i$ is an m -dimension vector for $i = 1, \dots, n_x$ or an n -dimension vector for $i = n_x + 1, \dots, n_x + n_y$ because

$$\begin{aligned} \bar{\varphi}_{Co_i} &= \exp\left(\frac{\|\mathbf{O}(1) - \mathbf{Co}^1\|^2 + \|\mathbf{O}(2) - \mathbf{Co}^2\|^2 + \dots + \|\mathbf{O}(n_x + n_y) - \mathbf{Co}^{n_x+n_y}\|^2}{-2\sigma^2}\right) \\ &= \exp\left(\frac{\|\mathbf{O}(1) - \mathbf{Co}^1\|^2 + \|\mathbf{O}(2) - \mathbf{Co}^2\|^2 + \dots + \|\mathbf{O}(n_x + n_y) - \mathbf{Co}^{n_x+n_y}\|^2}{-2(\sigma_1^2 + \sigma_2^2 + \dots + \sigma_{n_x+n_y}^2)}\right) = \exp\left(\frac{\|\mathbf{O}(1) - \mathbf{Co}^1\|^2}{-2\sigma_1^2}\right) \\ &\quad \cdot \exp\left(\frac{\|\mathbf{O}(2) - \mathbf{Co}^2\|^2}{-2\sigma_2^2}\right) \cdot \dots \cdot \exp\left(\frac{\|\mathbf{O}(n_x + n_y) - \mathbf{Co}^{n_x+n_y}\|^2}{-2\sigma_{n_x+n_y}^2}\right) = \varphi_{Co_1}^1 \cdot \varphi_{Co_2}^2 \cdot \dots \cdot \varphi_{Co_{n_x+n_y}}^{n_x+n_y}, \end{aligned} \quad (4)$$

where $Co_i = 1, 2, \dots, \prod_{i=1}^{n_x+n_y} C_i$.

The output of conventional RBFN can be written as

$$\hat{\mathbf{y}}(t) = \sum_{i_{Co}=1}^{\prod_{i=1}^{n_x+n_y} C_i} \bar{\varphi}_{i_{Co}} \cdot \bar{\omega}_{i_{Co}} = \sum_{Co_1=1}^{C_1} \dots \sum_{Co_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_{Co_1}^1 \cdot \varphi_{Co_2}^2 \cdot \dots \cdot \varphi_{Co_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}, \quad (5)$$

where $\bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}$ is the weight of the output layer. For each $\varphi_{Co_1}^1 \cdot \varphi_{Co_2}^2 \cdot \dots \cdot \varphi_{Co_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}$ there must be time-varying weights ($\bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^i$, $i = 1, \dots, n_x + n_y$) always found to make the following hold:

$$\begin{aligned} &\varphi_{Co_1}^1 \cdot \varphi_{Co_2}^2 \cdot \dots \cdot \varphi_{Co_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}} \\ &= \varphi_{Co_1}^1 \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^1 + \varphi_{Co_2}^2 \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^2 \cdot \dots \\ &\quad + \varphi_{Co_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^{n_x+n_y}. \end{aligned} \quad (6)$$

RBFN input $\mathbf{O}(i)$ represents both $\mathbf{x}$ and $\mathbf{y}$ vectors. The input of the conventional RBFN should be a $(m \cdot n_x + n \cdot n_y)$ -dimensional vector $[\mathbf{O}(1), \mathbf{O}(2), \dots, \mathbf{O}(n_x + n_y)]$, and the center vector of it has the same dimension with it, which is denoted by $\mathbf{Co}_{Co_i} = [\mathbf{Co}^1, \mathbf{Co}^2, \dots, \mathbf{Co}^{n_x+n_y}]$. $\mathbf{Co}^i$ should be chosen from $[\mathbf{C}_1^i, \dots, \mathbf{C}_{C_i}^i]$; therefore for each $\mathbf{Co}^i$ there are C_i possible. In other words, conventional RBFN needs $\prod_{i=1}^{n_x+n_y} C_i$ hidden nodes to be equal to the improved RBFN with $\sum_{i=1}^{n_x+n_y} C_i$ hidden nodes. It is obvious that $\sum_{i=1}^{n_x+n_y} C_i$ is less than $\prod_{i=1}^{n_x+n_y} C_i$; therefore, the improved RBFN has less hidden nodes.

2.2. Convergence of Improved RBFN. Above all, to analyze the convergence of improved RBFN, the relationship between improved RBFN and conventional RBFN is proposed in Lemma 1.

Lemma 1. *The improved RBFN is equivalent to the conventional RBFN.*

Proof. Suppose that $[\mathbf{O}(1), \mathbf{O}(2), \dots, \mathbf{O}(n_x + n_y)]$ are independent of each other. The output of the hidden layer of the conventional RBFN is given by

According to (6), (5) can be rewritten as

$$\begin{aligned} \hat{\mathbf{y}}(t) &= \sum_{Co_1=1}^{C_1} \dots \sum_{Co_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_{Co_1}^1 \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^1 + \varphi_{Co_2}^2 \\ &\quad \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^2 \cdot \dots + \varphi_{Co_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{Co_1 \dots Co_{n_x+n_y}}^{n_x+n_y}. \end{aligned} \quad (7)$$

Equation (7) can be rewritten as

$$\begin{aligned} \hat{\mathbf{y}}(t) &= \sum_{Co_2=1}^{C_2} \dots \sum_{Co_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_1^1 \cdot \bar{\omega}_{1, Co_2 \dots Co_{n_x+n_y}}^1 \\ &\quad + \sum_{Co_2=1}^{C_2} \dots \sum_{Co_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_2^1 \cdot \bar{\omega}_{2, Co_2 \dots Co_{n_x+n_y}}^1 + \dots \end{aligned}$$

$$\begin{aligned}
& + \sum_{C_2=1}^{C_2} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_{C_1}^1 \cdot \bar{\omega}_{C_1, C_2 \dots C_{n_x+n_y}}^1 \\
& + \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_1^2 \cdot \bar{\omega}_{C_1, 1 \dots C_{n_x+n_y}}^2 \\
& + \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_2^2 \cdot \bar{\omega}_{C_1, 2 \dots C_{n_x+n_y}}^2 \\
& + \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \varphi_{C_2}^2 \cdot \bar{\omega}_{C_1, C_2 \dots C_{n_x+n_y}}^2 \\
& + \cdots + \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \varphi_1^{n_x+n_y} \\
& \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, 1}^{n_x+n_y} + \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \varphi_2^{n_x+n_y} \\
& \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, 2}^{n_x+n_y} + \cdots \\
& + \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \varphi_{C_{n_x+n_y}}^{n_x+n_y} \\
& \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, C_{n_x+n_y}}^{n_x+n_y}.
\end{aligned} \tag{8}$$

Comparing (8) and (2), it is obvious that (6) always holds with Gaussian function RBFN:

$$\begin{aligned}
& \sum_{C_2=1}^{C_2} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{1, C_2 \dots C_{n_x+n_y}}^1 = \omega_1^1, \\
& \sum_{C_2=1}^{C_2} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{2, C_2 \dots C_{n_x+n_y}}^1 = \omega_2^1, \\
& \vdots \\
& \sum_{C_2=1}^{C_2} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{C_1, C_2 \dots C_{n_x+n_y}}^1 = \omega_{C_1}^1; \\
& \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{C_1, 1 C_3 \dots C_{n_x+n_y}}^2 = \omega_1^2, \\
& \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{C_1, 2 C_3 \dots C_{n_x+n_y}}^2 = \omega_2^2,
\end{aligned}$$

$$\begin{aligned}
& \vdots \\
& \sum_{C_1=1}^{C_1} \sum_{C_3=1}^{C_3} \cdots \sum_{C_{n_x+n_y}=1}^{C_{n_x+n_y}} \bar{\omega}_{C_1, C_2 C_3 \dots C_{n_x+n_y}}^2 = \omega_{C_2}^2; \\
& \vdots \\
& \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, 1}^{n_x+n_y} = \omega_1^{n_x+n_y}, \\
& \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, 2}^{n_x+n_y} = \omega_2^{n_x+n_y}, \\
& \sum_{C_1=1}^{C_1} \cdots \sum_{C_{n_x+n_y-1}=1}^{C_{n_x+n_y-1}} \bar{\omega}_{C_1 \dots C_{n_x+n_y-1}, C_{n_x+n_y}}^{n_x+n_y} = \omega_{C_{n_x+n_y}}^{n_x+n_y}.
\end{aligned} \tag{9}$$

From (9), we can see the improved RBFN with time-variant output weights is completely equivalent to the conventional RBFN. $\square$

To analyze the convergence of improved RBFN, we give two assumptions as follows: (1) each output must be bounded when $t \rightarrow \infty$. In other words, there must be a constant x , satisfying the relationship that $x_i(t) < x$, $i = 1, \dots, m$; (2) the first sample data of the time series is accurate and $x_i(1) \neq 0$, $i = 1, \dots, m$.

From (6), we know that the output weights of improved RBFN are time-variant, which are decided by the output of hidden nodes. In order to analyze the convergence, we divided the output of improved RBFN into $\prod_{i=1}^{n_x+n_y} C_i$ parts corresponding to the $\prod_{i=1}^{n_x+n_y} C_i$ outputs of hidden nodes of conventional RBFN. We take only one output of the improved RBFN for analysis, and other parts of the RBFN have the same property. From (5) and (6), the i th hidden nodes for a output of the RBFN are denoted by $f(t)$ for the left of (6) at time t , and the right are denoted by $g(t)$:

$$\begin{aligned}
f(t) &= \varphi_{C_1}^1 \cdot \varphi_{C_2}^2 \cdots \varphi_{C_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^i \\
&= \varphi_{Co}(t) \cdot \omega_{Co}, \\
g(t) &= \varphi_{C_1}^1 \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i,1} + \varphi_{C_2}^2 \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i,2} \cdots \\
&+ \varphi_{C_{n_x+n_y}}^{n_x+n_y} \cdot \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i, n_x+n_y} = \bar{\varphi}_{Co}(t) \cdot \bar{\omega}_{Co}^T(t),
\end{aligned} \tag{10}$$

where $\bar{\omega}_{Co}^T(t) = [\bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i,1}, \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i,2}, \dots, \bar{\omega}_{C_1 \dots C_{n_x+n_y}}^{i, n_x+n_y}]^T$ is the time-variant weight vector and $\bar{\varphi}_{Co}(t) = [\varphi_{C_1}^1, \varphi_{C_2}^2, \dots, \varphi_{C_{n_x+n_y}}^{n_x+n_y}]$ is the output of the hidden layer.

From (10), we can get the output of the hidden layer of improved RBFN at time $t + 1$:

$$\begin{aligned} f(t+1) &= \varphi_{Co}(t+1) \cdot \omega_{Co}, \\ g(t+1) &= \bar{\varphi}_{Co}(t+1) \cdot \bar{\omega}_{Co}^T(t+1). \end{aligned} \quad (11)$$

From (6), (10), and (11), we know that $f(t) = g(t)$ and $f(t+1) = g(t+1)$, and then the output weights of the improved RBFN at time $t + 1$ and t have the relation as follows:

$$\begin{aligned} \bar{\omega}_{Co}^T(t+1) &= \varphi_{Co}(t+1) \cdot \omega_{Co} \cdot \bar{\varphi}_{Co}^{-1}(t+1) \\ &= \varphi_{Co}(t+1) \cdot \frac{f(t)}{\varphi_{Co}(t)} \cdot \bar{\varphi}_{Co}^{-1}(t+1) \\ &= \varphi_{Co}(t+1) \cdot \frac{g(t)}{\varphi_{Co}(t)} \cdot \bar{\varphi}_{Co}^{-1}(t+1) \\ &= \varphi_{Co}(t+1) \cdot \frac{\bar{\varphi}_{Co}(t) \cdot \bar{\omega}_{Co}^T(t)}{\varphi_{Co}(t)} \\ &\quad \cdot \bar{\varphi}_{Co}^{-1}(t+1). \end{aligned} \quad (12)$$

So the deviation of the improved RBFN at time $t + 1$ is

$$\begin{aligned} |\sigma(t+1)| &= |g(t+1) - f(t+1)| = |\bar{\varphi}_{Co}(t+1) \\ &\quad \cdot \bar{\omega}_{Co}^T(t+1) - f(t+1)| = \left| \bar{\varphi}_{Co}(t+1) \cdot \varphi_{Co}(t+1) \right. \\ &\quad \cdot \frac{\bar{\varphi}_{Co}(t) \cdot \bar{\omega}_{Co}^T(t)}{\varphi_{Co}(t)} \cdot \bar{\varphi}_{Co}^{-1}(t+1) - f(t+1) \left. \right| \\ &= \left| \bar{\varphi}_{Co}(t+1) \cdot \varphi_{Co}(t+1) \cdot \omega_{Co} \cdot \frac{\bar{\varphi}_{Co}(t) \cdot \bar{\omega}_{Co}^T(t)}{\varphi_{Co}(t) \cdot \omega_{Co}} \right. \\ &\quad \cdot \bar{\varphi}_{Co}^{-1}(t+1) - f(t+1) \left. \right| = \left| f(t+1) \cdot \frac{g(t)}{f(t)} \right. \\ &\quad \cdot \bar{\varphi}_{Co}^{-1}(t+1) - f(t+1) \left. \right| = \left| \frac{f(t+1) \cdot [g(t) - f(t)]}{f(t)} \right| \\ &= \left| \frac{f(t+1) \cdot \sigma(t)}{f(t)} \right| = \left| \frac{f(t+1) \cdot f(t) \cdot \sigma(t-1)}{f(t-1) \cdot f(t)} \right| \\ &= \dots = \left| \frac{f(t+1) \cdot \dots \cdot f(2) \cdot \sigma(1)}{f(1) \cdot \dots \cdot f(t)} \right| \\ &= \left| \frac{f(t+1) \cdot \sigma(1)}{f(1)} \right|. \end{aligned} \quad (13)$$

Since $f(t+1)$ is the hidden layer output of conventional RBFN and approximation property of conventional RBFN, there is relationship that $f(t+1) = y_i(t+1) + \varepsilon$, where $y_i(t+1)$

denotes the true value of i th output. ε denotes an arbitrarily small value. $|\sigma(t+1)|$ can be written as

$$\begin{aligned} |\sigma(t+1)| &= \left| \frac{[g(1) - f(1)] \cdot [y_i(t+1) + \varepsilon]}{f(1)} \right| \\ &\leq \left| \left[\frac{g(1)}{f(1)} - 1 \right] [y_i(t+1) + \varepsilon] \right|, \end{aligned} \quad (14)$$

where y_i is the true value of system output. According to assumption (2), we know that $y_i(1) \neq 0$ and $y_i(1)$ is accurate. Thus $y_i(1)$ is used to replace $g(1)$ and the equation is obtained as follows:

$$\frac{g(1)}{f(1)} - 1 = \frac{y_i(1)}{y_i(1) + \varepsilon} - 1 = \frac{-\varepsilon}{y_i(1) + \varepsilon}. \quad (15)$$

Because of the approximation property of the conventional RBFN, $y_i(1)$ is bounded. $-\varepsilon/(y_i(1) + \varepsilon)$ also can be arbitrarily small when $t \rightarrow \infty$. Thus the conclusion of the convergence of improved RBFN is proved.

2.3. Outliers Detection Using Hidden Markov Model. The real value of $\hat{\mathbf{y}}(t)$ is denoted by $\mathbf{y}(t)$. From the approximation property of the RBFN, the prediction $\hat{\mathbf{y}}(t)$ should be close to its true value, $\mathbf{y}(t)$. For the j th output, it has the following relationship:

$$\hat{y}_j(t) = y_j(t) + \hat{\varepsilon}_j(t), \quad (16)$$

where $j = 1, \dots, n$ and $\hat{\varepsilon}_j(t)$ is the estimation of the white noise which obeys the Gaussian distribution $N(0, U_j)$. The probability density function of $\hat{\varepsilon}_j(t)$ is Gaussian-type membership function and given by [11, 12]

$$\begin{aligned} P(\hat{\varepsilon}_j(t) | \mu_j, U_j) \\ = \exp\left(-\frac{1}{2} (\hat{y}_j(t) - y_j(t))^T U_j^{-1} (\hat{y}_j(t) - y_j(t))\right), \end{aligned} \quad (17)$$

where $\mu_j = 0$. The higher the probability is, the closer $\hat{\varepsilon}_j(t)$ is equal to 0 (the closer $\hat{y}_j(t)$ is equal to $y_j(t)$). In other words, if $P(\hat{\varepsilon}_j(t) | \mu_j, U_j)$ equals or is close to 0 at time t , $y_j(t)$ is considered to be a possible outlier.

In order to make outliers detection accurately, HMM is introduced. HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved states. One of the problems in HMM is choosing a most likely state chain with the given observation chain and model. This problem can be solved by Viterbi algorithm [16]. In the observation chain, state transition probabilities are the only parameters. $P(\hat{\varepsilon}_j(t) | \mu_j, U_j)$ is fit for this character. Hence, an HMM is introduced to detect whether the data are outliers or not. It is noted that the dimension of the chains is the same and equal to the dimension of system output.

An HMM for outliers detection is characterized by the following [17]:

- (1) Outliers detection is done separately for each system output.

- (2) $P(\hat{\varepsilon}_j(t) \mid \mu_j, U_j)$ represents the chain which has been observed. The hidden chain is the state of the observation.
- (3) There are two states in the HMM. “1” denotes that the state is normal, and “0” means it is outlier.
- (4) The transition probability of hidden chain is defined as follows:

$$\begin{aligned} a_{01} &= \frac{N(a_{01})}{N(a_{01} + a_{00})}, \\ a_{11} &= \frac{N(a_{11})}{N(a_{11} + a_{10})}, \\ a_{00} &= 1 - a_{01}, \\ a_{10} &= 1 - a_{11}, \end{aligned} \quad (18)$$

where $a_{i_s j_s}$ ($i_s, j_s = 0$ or 1) denotes the transition probability from state i_s to j_s and $N(a_{i_s j_s})$ denotes the total transition times.

To find the best detection result, the probability is defined as

$$\begin{aligned} \phi_{i_s}(t) &= \max_{s(1), s(2), \dots, s(t-1)} P[s(1), s(2), \dots, s(t)] \\ &= i_s, p(1), p(2), \dots, p(t), \end{aligned} \quad (19)$$

where $\{p(1), p(2), \dots, p(t)\}$ is a sequence of observation probabilities and calculated by (17); the element in sequence $S = \{s(1), s(2), \dots, s(t)\}$ denotes the state; and $\phi_{i_s}(t)$ is the best score (highest probability) for finding the best state sequence. Since the outliers are detected in a time sequence and the previous data before time t have already been detected, the conventional Viterbi algorithm can be simplified. Equation (19) is rewritten as

$$\phi_{i_s}(t) = \max P[s(t) = i_s, p(t)]. \quad (20)$$

State transition probability $a_{i_s j_s}$ is used to express all influences of historical observations probability and states, and (20) can be rewritten as

$$\begin{aligned} \phi_1(t) &= a_{i_s 1} p_1(t); \\ \phi_0(t) &= a_{i_s 0} p_0(t). \end{aligned} \quad (21)$$

$p_1(t)$ obtained by (17) is the observation probability that state is “1” and $p_0(t) = 1 - p_1(t)$. Then the state of the data at time t can be determined:

If $\phi_1(t) < \phi_0(t)$, then $s(t) = 0$, and $x(t)$ is an outlier.

Else then $s(t) = 1$, $x(t)$ is normal.

Then the data which are detected as outliers are removed and replaced by the mean of clean data nearby.

3. Dynamic PLS Modeling and Controller Design

3.1. Dynamic PLS Modeling. The conventional PLS consists of an outer model which extracts the latent variables and

an inner model to estimate the relation between the input and output scores. Considering an $l \times m$ dimension input dataset $\mathbf{X}$ and an $l \times n$ dimension output dataset $\mathbf{Y}$, l , m , and n denote the number of samplings, input dimension, and output dimension, respectively. It is also assumed that the dataset has been processed by the outliers detection algorithm and scaled to zero mean and unit variance. As for outer model, it is built by decomposing matrices $\mathbf{X}$ and $\mathbf{Y}$ as follows:

$$\begin{aligned} \mathbf{X} - \mathbf{E}^* &= \mathbf{T}\mathbf{P}^T = \sum_{r=1}^R \mathbf{t}_r \mathbf{p}_r^T, \\ \mathbf{Y} - \mathbf{F}^* &= \mathbf{U}\mathbf{Q}^T = \sum_{r=1}^R \mathbf{u}_r \mathbf{q}_r^T, \end{aligned} \quad (22)$$

where R is the number of latent variables; $\mathbf{p}_r$ and $\mathbf{q}_r$ are the r th loading vectors of loading matrices $\mathbf{P}$ and $\mathbf{Q}$, respectively; $\mathbf{t}_r$ and $\mathbf{u}_r$ are the r th score vectors of matrices $\mathbf{T}$ and $\mathbf{U}$, respectively; and $\mathbf{E}^*$ and $\mathbf{F}^*$ are residual matrices of $\mathbf{X}$ and $\mathbf{Y}$, respectively.

In inner model, the score matrices $\mathbf{P}$ and $\mathbf{Q}$ are related by a diagonal matrix $\mathbf{B}$ which is obtained by least squares (LS) method:

$$\begin{aligned} \mathbf{u}_r &= \mathbf{b}_r \mathbf{t}_r, \\ \mathbf{U} &= \mathbf{T}\mathbf{B}, \end{aligned} \quad (23)$$

$$\mathbf{Y} - \mathbf{F}^* = \mathbf{T}\mathbf{B}\mathbf{Q}^T = \sum_{r=1}^R \mathbf{b}_r \mathbf{t}_r \mathbf{q}_r^T,$$

where b_r is the r th element of inner relation matrices $\mathbf{B}$.

It is noted that PLS has the effect of dimensionality reduction. $R < \max(m, n)$ is met in most cases. And nonsquare and ill-conditioned system control problem can be dealt with due to the fact that PLS decomposes the multivariate regression problem into a series of univariate regression problems [2].

The major drawback of PLS is that it is not suitable for dynamic systems. Researchers have proposed many different dynamic PLS models by employing structures like time series term or dynamic filters into PLS structure [4, 18]. In this paper, integrating autoregressive exogenous (ARX) model is applied into inner model of PLS [19] to represent the dynamic character of process. It can be expressed as follows:

$$\begin{aligned} \mathbf{u}_r &= \mathbf{H}_r(\mathbf{t}_r) \\ \mathbf{U} &= \mathbf{H}(\mathbf{T}), \end{aligned} \quad (24)$$

where $\mathbf{H}(\mathbf{T}) = \text{diag}(\mathbf{H}_1(\mathbf{t}_1), \mathbf{H}_2(\mathbf{t}_2), \dots, \mathbf{H}_R(\mathbf{t}_R))$. $\mathbf{H}_r(\cdot)$ denotes the ARX model of the r th subsystem in the latent variable space. With (24), (23) is transformed to

$$\mathbf{Y} - \mathbf{F}^* = \mathbf{H}(\mathbf{T})\mathbf{Q}^T = \sum_{r=1}^R \mathbf{H}_r(\mathbf{t}_r) \mathbf{q}_r^T. \quad (25)$$

FIGURE 2: PLS control framework proposed by Kaspar and Ray.

FIGURE 3: GPC in dynamic PLS framework.

An LS algorithm is used to identify the parameters of ARX model in the latent space.

3.2. Dynamic PLS Framework for Controller Design. The controller design in latent variable space proposed by Kaspar and Ray [4] is shown in Figure 2. W_x and W_y are scaling matrices of X and Y ; G_c , G_p , and G_d are the transfer function matrices of the controller in latent variable space, the controlled plant, and disturbances, respectively. P and Q are the loading matrices of the input data and output data, respectively. Q^+ is the appropriate inverse of Q . Since the MIMO system is decomposed into multiple SISO subsystems in latent variable space, the original MIMO controller problem is decomposed into multiple independent SISO control problems. A variety of conventional control schemes could be applied in this control framework [4]. In this paper, GPC would be employed.

3.3. Improved GPC Controller Design in Dynamic PLS Framework. GPC is one of the model predictive control methods. It has been successfully implemented in many industry and academia, showing good performance and a certain degree of robustness [20]. The GPC controller in dynamic PLS is proposed by Chi et al. [2]. This method has some special advantages, such as simple controller design procedure and low computing time. The predictive model used in GPC is

called controlled autoregressive integrated moving averaged (CARIMA) model:

$$A(z^{-1})Y(t) = B(z^{-1})z^{-d}X(t-1) + \frac{C(z^{-1})\xi(t)}{\Delta}, \quad (26)$$

where $A(z^{-1}) = I + A_1 z^{-1} + \dots + A_{n_a} z^{-n_a}$, $B(z^{-1}) = I + B_1 z^{-1} + \dots + B_{n_b} z^{-n_b}$, $C(z^{-1}) = C_0 + C_1 z^{-1} + \dots + C_{n_c} z^{-n_c}$, and $\Delta = 1 - z^{-1}$.

As demonstrated by Lauri et al. [21], a filtered version ARX is equivalent to the CARIMA model for both SISO case and MIMO case. Hence, the ARX models which are obtained in the latent variable space (Section 3.1) can be used for GPC. Replacing the controller G_c in Figure 2, the GPC scheme in dynamic PLS framework is illustrated in Figure 3. The setpoints in the original space are transferred into latent variable space; then the output scores are subtracted to be the inputs of the GPC controllers. The outputs of the GPC controllers are transferred back to the original space as the input of the real process.

The cost function for the r th GPC controller is

$$J_r = \sum_{j=1}^{N_{1r}} \delta_r(j) \|u_{\text{set}_r}(t+j) - \hat{u}_r(t+j)\|_2^2 + \sum_{j=1}^{N_{2r}} \lambda_r(j) \|\Delta t_r(t+j-1)\|_2^2, \quad (27)$$

where $\hat{u}_r(t+j)$ is the predictive output of j step ahead. N_{1r} and N_{2r} are the control horizon and predictive horizon, respectively. To make the discussion simple, let $N_1 = \min_r N_{1r}$, $N_2 = \min_r N_{2r}$ in the following discussion. $\delta_r(j)$ and $\lambda_r(j)$ are weighting sequences. $\delta_r(j)$ ($j = 1, \dots, N_1$) is usually a constant weighting sequence that considers future behavior of controllers. It is chosen based on the system features and experience of the engineers while, in this paper, it is dynamically tuned based on the residual of setpoints and output of the systems. In order to denote time-varying weighting sequence, it is rewritten as $\delta_r(t+j)$. Then, (27) is rewritten as

$$J_r = \sum_{j=1}^{N_1} \delta_r(t+j) \cdot \|u_{\text{set}_r}(t+j) - \hat{u}_r(t+j)\|_2^2 + \sum_{j=1}^{N_2} \lambda_r(j) \cdot \|\Delta t_r(t+j-1)\|_2^2. \quad (28)$$

$\delta_r(t+j)$ is calculated by the following equation:

$$\delta_r(t+j) = \delta_r^0 + \sum_{p=1}^R \theta_{rp} \|u_{\text{set}_p}(t+j) - \hat{u}_p(t+j)\|_2^2, \quad (29)$$

where δ_r^0 is a threshold and θ_{rp} is a weight coefficient. And $\theta_{rp} = 0$, when $r = p$.

δ_r affects the weight on the residual of setpoints and prediction outputs. In the case of incompletely decoupling caused by model mismatch, this method can restrain the interaction in the presence of part coupling. The principle of interaction rejection with $\delta_r(t+j)$ is described as follows. When the residual of p th output and setpoint increases, the weight $\delta_r(t+j)$ is adjusted with (29). The adjustment speed of outputs except p th output is accelerated because weight $\delta_r(t+j)$ ($r \neq p$) is increased according to (29). Then adjustment time and overshoot value will be reduced. In this way, the p th output interaction on other output would be reduced.

Take a two-input two-output latent variable model as example. When the 1st output deviation between $u_{\text{set}_1}(t+j)$ and $\hat{u}_1(t+j)$ is produced, the weight $\delta_2(t+j)$ of 2nd output is increased. Conversely, when the 1st output deviation is reduced to zero, $\delta_2(t+j)$ is reduced to the threshold δ_2^0 . $\delta_2(t+j)$ is adjusted with $\|u_{\text{set}_1}(t+j) - \hat{u}_1(t+j)\|_2^2$ as shown in Figure 4.

According to the prediction scheme of GPC, $\delta_r(t+j)$ in (29) consists of $\hat{u}_p(t+j)$ of other outputs. The cost function (28) is a multivariate minimization problem. These variables influence each other, which reflects variables interactions due to the plan/model mismatch. There are R cost functions for the R GPC controllers in latent space; these series of minimized problem $\min(J_r)$ ($r = 1, \dots, R$) are equal to the minimized problem of $\min(J)$ as follows:

$$J_1 + J_2 + \dots + J_R = J. \quad (30)$$

FIGURE 4: $\delta_2(t+j)$ is adjusted with the 1st output deviation.

And the total cost function J is

$$J = \sum_{r=1}^R \sum_{j=1}^{N_1} \delta_r(t+j) \cdot \|u_{\text{set}_r}(t+j) - \hat{u}_r(t+j)\|_2^2 + \sum_{r=1}^R \sum_{j=1}^{N_2} \lambda_r(j) \cdot \|\Delta t_r(t+j-1)\|_2^2. \quad (31)$$

Equation (31) is rewritten in the form of matrix as

$$\mathbf{J} = (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}})^T \boldsymbol{\delta} (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}) + \Delta \mathbf{T}^T \boldsymbol{\lambda} \Delta \mathbf{T}, \quad (32)$$

where $\boldsymbol{\delta} = \begin{bmatrix} \delta_1 & \dots & 0 \\ 0 & \ddots & \delta_R \end{bmatrix}$, $\boldsymbol{\delta}_r = \begin{bmatrix} \delta_r(t+1) & \dots & 0 \\ 0 & \ddots & \delta_r(t+N_1) \end{bmatrix}$, $r = 1, \dots, R$.

Let $\mathbf{z}_{pj} = [0, \dots, 0, 1, 0, \dots, 0]^T$, $\theta_{rpj} = \theta_{rp} \mathbf{z}_{pj}^T$, $\theta_{rj} = [\theta_{r1j}, \dots, \theta_{rRj}]$, and $\mathbf{z}_{rj} = [\mathbf{z}_{r1j}^T, \dots, \mathbf{z}_{rRj}^T]^T$; (29) is rewritten as

$$\delta_r(t+j) = \delta_r^0 + \sum_{p=1}^R \theta_{rp} (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}})^T \mathbf{z}_{pk}^T \mathbf{z}_{pk} (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}) = \delta_r^0 + (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}})^T \theta_{rj} \mathbf{z}_{rj} (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}). \quad (33)$$

Let $\boldsymbol{\delta}_r^0 = \begin{bmatrix} \delta_r^0 & \dots & 0 \\ 0 & \ddots & \delta_r^0 \end{bmatrix}$, $\boldsymbol{\theta}_r = \begin{bmatrix} \theta_{r1} & \dots & 0 \\ 0 & \ddots & \theta_{rN_1} \end{bmatrix}$, $\bar{\mathbf{U}}_{\text{set}_r} - \bar{\hat{\mathbf{U}}}_r = \begin{bmatrix} (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}) & \dots & 0 \\ 0 & \ddots & (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}) \end{bmatrix}$, and $\mathbf{z}_r = \begin{bmatrix} \mathbf{z}_{r1} & \dots & 0 \\ 0 & \ddots & \mathbf{z}_{rN_1} \end{bmatrix}$; then the r th output weight is $\boldsymbol{\delta}_r = \boldsymbol{\delta}_r^0 + (\bar{\mathbf{U}}_{\text{set}_r} - \bar{\hat{\mathbf{U}}}_r)^T \boldsymbol{\theta}_r \mathbf{z}_r (\bar{\mathbf{U}}_{\text{set}_r} - \bar{\hat{\mathbf{U}}}_r)$.

Let $\boldsymbol{\delta}^0 = \text{diag}[\boldsymbol{\delta}_1^0, \dots, \boldsymbol{\delta}_R^0]$, $\boldsymbol{\theta} = \text{diag}[\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_R]$, $\mathbf{Z} = \text{diag}[\mathbf{z}_1, \dots, \mathbf{z}_R]$, and $\bar{\mathbf{U}}_{\text{set}} - \bar{\hat{\mathbf{U}}} = \text{diag}[\bar{\mathbf{U}}_{\text{set}_1} - \bar{\hat{\mathbf{U}}}_1, \dots, \bar{\mathbf{U}}_{\text{set}_R} - \bar{\hat{\mathbf{U}}}_R]$; weight matrix has the form

$$\boldsymbol{\delta} = \boldsymbol{\delta}^0 + (\bar{\mathbf{Y}} - \bar{\mathbf{W}})^T \boldsymbol{\theta} \mathbf{Z} (\bar{\mathbf{Y}} - \bar{\mathbf{W}}). \quad (34)$$

Replace the weight factor in (32) with (34); the cost function of improved MIMO GPC in latent variable space is written as follows:

$$\mathbf{J} = (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}})^T \left(\boldsymbol{\delta}^0 + (\bar{\mathbf{Y}} - \bar{\mathbf{W}})^T \boldsymbol{\theta} \mathbf{Z} (\bar{\mathbf{Y}} - \bar{\mathbf{W}}) \right) \cdot (\mathbf{U}_{\text{set}} - \hat{\mathbf{U}}) + \Delta \mathbf{T}^T \boldsymbol{\lambda} \Delta \mathbf{T}. \quad (35)$$

Equation (35) is a nonlinear optimal problem; interior point method [22] is a typical method to solve this problem.

From the discussion above, the improved GPC has 4 types of parameters (predictive horizon N_1 , control horizon N_2 , weight threshold δ_r^0 , and weight coefficient θ_{rp}) which would influence the control result. They should be chosen properly. For each step of the control period, the following procedures are done for control incremental update.

- (1) Get the prediction $\hat{\mathbf{U}}$ of the outputs.
- (2) Calculate (35) by interior point method.
- (3) Send the first element of every $\Delta \mathbf{T}$ to the score vector $\mathbf{T}$ as $t_r(t+1) = t_r(t) + \Delta t_r(t)$, where $r = 1, \dots, R$.

4. Examples

In this section, two examples are presented to demonstrate the advantages of proposed modeling and control strategy.

4.1. Example 1. The first example is an industrial-scale polymerization reactor control problem, which is proposed by Chien et al. [23]. The basic control strategy is similar to the terpolymerization reactor example by Ogunnaike et al. [24]. The advanced product quality control loops are used to track the setpoints of the reactor condition control loops. The process model for the reactor setpoints control loops is

$$\mathbf{Y}(s) = \begin{bmatrix} \frac{22.89e^{-0.2s}}{4.572s + 1} & \frac{-11.64e^{-0.4s}}{1.807s + 1} \\ \frac{4.689e^{-0.2s}}{2.174s + 1} & \frac{5.80e^{-0.4s}}{1.801s + 1} \end{bmatrix} \mathbf{X}(s) + \begin{bmatrix} \frac{-4.243e^{-0.4s}}{3.445s + 1} \\ \frac{-0.601e^{-0.4s}}{1.982s + 1} \end{bmatrix} \mathbf{d}(s). \quad (36)$$

Controlled variables are the two measurements representing the reactor condition; and manipulated variables are the setpoints of two reactor feed flow loops; load disturbance is the purge flow of the reactor.

In order to build up the dynamic PLS model, two random step signals with magnitude $[-0.5 \ 0.5]$ are applied to excite the system. And random signals with $[-0.05 \ 0.05]$ and $[-0.1 \ 0.1]$ are added to both disturbance inputs separately in order to simulate the disturbance influence. The generated datasets are plotted in Figure 5.

In order to demonstrate the performance of outliers detection method, a number of outliers are added to the outputs of the modeling dataset and several performance criteria are involved to evaluate the performance.

In this example, the number of latent variables is 2. And an outlier is defined as 3 times bigger than the true value to simulate the failure that some samples are measured abnormally. Outliers are randomly added to the outputs at different rate. The data with more than 50% outliers are considered invalid because the useful information it

FIGURE 5: Modeling dataset for Wood and Berry process.

carried is not sufficient. Since the numbers of outliers are added randomly, the results of different simulations may be different. For example, the result with 100 discontinuous outliers is better than the continuous ones. A discontinuous outlier can be detected and imputed easily based on the adjacent true points.

For each rate of outliers, 3 performance criteria are calculated to evaluate the performance of outliers detection method. Let H , α , β , and χ denote the number of samples, outliers that are detected, error detection (normal samples detected to be outliers), and missing detection (outliers that are not detected), respectively. The criteria are described as follows:

- (1) Accurate rate is the ratio of actual outliers among $(\alpha - \beta)$ and actual outliers $(\alpha - \beta + \chi)$:

$$\text{accurate rate} = \frac{\alpha - \beta}{\alpha - \beta + \chi}. \quad (37)$$

- (2) Error detection means the true value is considered as an outlier. Error rate is the ratio of the number of error detections (β) and the number of actual true samples $(H - \alpha + \beta - \chi)$:

$$\text{error rate} = \frac{\beta}{H - \alpha + \beta - \chi}. \quad (38)$$

- (3) Missing detection means outliers are considered as true samples. Missing rate is the ratio of the number of missing detections (χ) and the number of actual outliers $(\alpha - \beta + \chi)$:

$$\text{missing rate} = \frac{\chi}{\alpha - \beta + \chi}. \quad (39)$$

FIGURE 6: Detected results for 10% outliers.

TABLE 1: The detection result of different rates of outliers.

Percentage of outliers	Output number	Accurate rate	Error rate	Missing rate
1%	Output 1	100%	2.68%	0%
	Output 2	90%	1.24%	10%
5%	Output 1	68.75%	6.22%	31.25%
	Output 2	77.08%	4.08%	22.92%
10%	Output 1	36.96%	5.51%	63.04%
	Output 2	16.44%	1.01%	83.56%
25%	Output 1	11.32%	2.60%	88.68%
	Output 2	14.05%	2.77%	85.95%
50%	Output 1	7.09%	2.50%	92.91%
	Output 2	12.76%	3.02%	87.24%

The simulation result is shown in Table 1. It is clear that the less outliers contained, the better accurate rate is obtained. It is noted that the results in Table 1 could have a bit different because of the randomness of outliers.

The detected results of 10% outliers are shown in Figures 6(b) and 6(d). As observed in Figure 6 and Table 1, most outliers have been detected out correctly when the rate of outliers is small. That is because when outliers' rate increases, the probability that samples nearby outliers are also outliers is bigger. That is to say, the useful information nearby would be missing; the RBFN method would not get enough information to predict the next output correctly. The samples of y_1 from 600 to 750 are all detected as true values because the values of these samples are very small, and the outliers are submerged in the noises.

The data with 10% outliers would be used for further dynamic PLS modeling. The outliers that have been detected as outliers would be replaced with the corresponding samples next to them. In order to evaluate the robustness of dynamic PLS modeling with outliers detection, a criterion of comparison is applied [25]. The angle between loading vectors of the improved method with outliers detection and conventional method is chosen and is defined as $\text{ang}_r = \text{ang}(\mathbf{q}_r, \mathbf{q}'_r)$, where $\text{ang}_r(\mathbf{q}_r, \mathbf{q}'_r)$ is the function that calculates the angle between the loading vectors $\mathbf{q}_r$ and $\mathbf{q}'_r$. $\mathbf{q}_r$ denotes the loading vector which is derived from the data without outliers. $\mathbf{q}'_r$ denotes the loading vectors of modeling method with outliers detection or conventional modeling method.

TABLE 2: The comparison of the two methods.

	Conventional method	Improved method
Ang_1	13.8	2.46
Ang_2	9.31	2.75

FIGURE 7: The validation for the improved dynamic PLS modeling method and conventional method.

The simulation result is shown in Table 2. From Table 2, it is clear that the improved method has smaller angle than the conventional method. A model validation test of the two methods is done and shown in Figure 7. The other 500 samples without outliers are used to perform this test. The improved dynamic PLS without outliers detection method and conventional method are denoted as ODPLS and DPLS for short. The integral of squared error (ISE) is applied to evaluate the deviation between real data and modeling result which is described as

$$\text{ISE}_i = \int_0^\infty [\hat{y}(\tau)_i - y_i(\tau)]^2 d\tau, \quad (40)$$

where $y_i(\tau)$ denotes the real data, $\hat{y}(\tau)_i$ denotes the modeling result data, and $i = 1, \dots, n$ denotes the output number.

FIGURE 8: The simulation result of SGPC with ODPLS and DPLS.

FIGURE 9: The simulation result of MGPC with ODPLS and DPLS.

In Figure 7, the two ISEs of ODPLS are 1.23×10^3 and 0.09×10^3 , while ISEs of DPLS are 1.38×10^4 and 0.12×10^4 . It is clear that the dynamic PLS modeling with outliers detection is more robust than conventional method.

In addition, the SISO GPC (SGPC) controllers and improved MIMO GPC (MGPC) controller in latent variable space are implemented with the two modeling methods. The results are shown in Figures 8 and 9. The predictive horizon N_1 and control horizon N_2 have the same value in the 4 simulations ($N_1 = 2$, $N_2 = 1$). It is noted that the SGPC and MGPC with DPLS model would be unstable when N_1 and N_2 are bigger because of the mismatch between the DPLS model and the real process which is caused by outliers. It is

FIGURE 10: Modeling dataset for Wood and Berry process.

observed that SGPC and MGPC with ODPLS obtain better control performance than that with DPLS in terms of rise time and ISE. That is because ODPLS partly removes outliers away and make the modeling more accurate than DPLS. With the same model (DPLS or ODPLS), it can also be observed that MGPC obtains better result than SGPC in terms of rise time and ISE. In summary, it can be concluded that MGPC with ODPLS has the least interaction between the two outputs among the 4 simulations.

4.2. Example 2. The methanol-water distillation column model reported by Wood and Berry [26] is a typical 2-by-2 process with strong interaction. The reflux and the reboiler steam flow rates are the process inputs. The compositions of the top and bottom products are the process outputs. The discrete sample time $T_s = 1$ min. The transfer function is given as [6]

$$\mathbf{G}_p(s) = \begin{bmatrix} \frac{12.8e^{-s}}{16.7s + 1} & \frac{-18.9e^{-3s}}{21s + 1} \\ \frac{6.6e^{-7s}}{10.9s + 1} & \frac{-19.4e^{-3s}}{14.4s + 1} \end{bmatrix}. \quad (41)$$

In order to build up the dynamic PLS model, two 1000 sampling random step signals with $[-0.5 \ 0.5]$ are applied to excite the system and generate the outputs. And white noise in which the SNR (signal to noise ratio) is 10 DB is added to both outputs in order to simulate the actual outputs. The generated dataset is plotted in Figure 10.

In this example, the number of latent variables is 2. And a zero value or a small value is defined as an outlier to simulate the measuring equipment failure that some samples are detected abnormally. 10% and 25% rates of outliers are added. The effect of RBFN parameters on the outliers detection result is shown in Tables 3 and 4. These parameters include

FIGURE 11: Detected results for 10% outliers with RBFN parameters (4, 4, 2).

TABLE 3: Detection result of different RBFN parameters with 10% outliers.

	RBFN parameters (n_x, n_y, C)				
	(3, 3, 2)	(4, 4, 2)	(5, 5, 2)	(3, 3, 3)	(3, 3, 4)
Output 1					
Accurate rate	84.95%	82.61%	86.73%	84.69%	83.87%
Error rate	17.14%	14.86%	14.74%	16.09%	15.44%
Missing rate	15.05%	17.39%	13.27%	15.31%	16.13%
Output 2					
Accurate rate	79.79%	75.76%	82.80%	88.30%	82.61%
Error rate	11.17%	10.44%	18.82%	16.93%	12.95%
Missing rate	20.21%	24.24%	17.20%	11.70%	17.39%

TABLE 4: Detection result of different RBFN parameters with 25% outliers.

	RBFN parameters (n_x, n_y, C)				
	(3, 3, 2)	(4, 4, 2)	(5, 5, 2)	(3, 3, 3)	(3, 3, 4)
Output 1					
Accurate rate	74.07%	75.18%	75.57%	27.86%	15.56%
Error rate	16.21%	13.82%	3.41%	1.31%	3.08%
Missing rate	25.93%	24.82%	24.43%	72.14%	84.44%
Output 2					
Accurate rate	69.50%	67.65%	66.67%	43.61%	13.33%
Error rate	12.27%	10.42%	12.62%	8.03%	7.89%
Missing rate	30.50%	32.35%	33.33%	56.39%	86.67%

the number of input and output time delays (n_x, n_y) and hidden node center (C), where the hidden node center of each RBFN input has the same number C . Comparing Tables 3 and 4, RBFN parameters have more impact on the detection result at 25% rate of outliers detection than at 10%. That is because great rate of outliers makes outliers detection more difficult and results in the poor robustness of RBFN. It is obvious that RBFN with parameters (4, 4, 2) has better outliers detection result shown in Table 4. It is noted that the real detected result such as that in Table 4 or Table 3 cannot be obtained in the real process. Parameters may be selected based on the ratio of ISE and integral of output (ISE/IO) as shown in Table 5 which would be fitting the samples properly. Shown in Figure 11 are the detected results of 10% outliers with RBFN parameters (4, 4, 2) which has better fitting result and is used for the further control simulation.

TABLE 5: ISE/IO of different RBFN parameters.

RBFN parameters (n_x, n_y, C)	10% outliers		25% outliers	
	Output 1	Output 2	Output 1	Output 2
(3, 3, 2)	1.60	1.26	3.29	2.74
(4, 4, 2)	1.29	1.24	3.23	2.74
(5, 5, 2)	1.45	1.17	3.24	2.74
(3, 3, 3)	1.51	1.42	3.24	2.75
(3, 3, 4)	1.67	1.29	3.26	2.77

FIGURE 12: The simulation result of MGPC with different parameters.

Figure 12 shows the control result of MGPC with different parameters, where the solid line is the result of parameter, $\delta_1^0 = 1$, $\delta_1^1 = 1$, $\theta_{12} = 2$, $\theta_{22} = 2$ (1, 1, 2, 2), and dotted line is the result of parameter, $\delta_1^0 = 1$, $\delta_1^1 = 2$, $\theta_{12} = 5$, $\theta_{22} = 5$ (1, 2, 5, 5). It is clear to see the interaction between the 2 outputs at times 250 and 750, where y_2 is disturbed by the setpoint changing of y_1 . From Figure 12, control result of dotted line has less overshoot and shorter convergence time. It can conclude that the parameters δ_r^0 and θ_{rp} in (31) have a great influence on the control result. With appropriate parameters in (31) a better control result can be obtained.

FIGURE 13: The simulation result of disturbance rejection.

In real process, there are some disturbance signals under the control framework. In order to get the disturbance rejection performance, y_1 and y_2 are separately set at 0.1 and 0, and a 50% change is added to y_1 . The result is shown in Figure 13. MGPC has less disturbance than SGPC for both y_1 and y_2 . And the convergence time of MGPC is also shorter.

5. Conclusions

Since dynamic PLS modeling is sensitive to outliers in the data, a new RBFN structure with HMM to detect outliers is proposed to deal with contaminated data. The improved RBFN considered past samples as the input of the network for the dynamic system prediction. It has less hidden nodes; the equivalent to the conventional RBFN and convergence is proved. With HMM, there is no need to preselect a threshold to detect outliers. With the processed data in this method, a more robust dynamic PLS model is obtained. Unfortunately, there is still mismatch between the model and the real process, caused by many reasons. The inner model cost function in dynamic PLS is treated as a special MIMO problem and a GPC based on the adaptive weights is used to tackle this problem. Two examples are used to demonstrate the performance of outlier detecting, accurate modeling, and control.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61174114), the Research Fund for the Doctoral Program of Higher Education in China

(20120101130016), and the Zhejiang Provincial Science and Technology Planning Projects of China (2014C31019).

References

- [1] Y. Yao and F. Gao, "A survey on multistage/multiphase statistical modeling methods for batch processes," *Annual Reviews in Control*, vol. 33, no. 2, pp. 172–183, 2009.
- [2] Q. Chi, Z. Fei, Z. Zhao, L. Zhao, and J. Liang, "A model predictive control approach with relevant identification in dynamic PLS framework," *Control Engineering Practice*, vol. 22, no. 1, pp. 181–193, 2014.
- [3] M. H. Kaspar and W. H. Ray, "Chemometric methods for process monitoring and high-performance controller design," *AIChE Journal*, vol. 38, no. 10, pp. 1593–1608, 1992.
- [4] M. H. Kaspar and W. H. Ray, "Dynamic PLS modelling for process control," *Chemical Engineering Science*, vol. 48, no. 20, pp. 3447–3461, 1993.
- [5] J. Chen, Y.-C. Cheng, and Y. Yea, "Multiloop PID controller design using partial least squares decoupling structure," *Korean Journal of Chemical Engineering*, vol. 22, no. 2, pp. 173–183, 2005.
- [6] D. Laurí, M. Martínez, J. V. Salcedo, and J. Sanchis, "PLS-based model predictive control relevant identification: PLS-PH algorithm," *Chemometrics and Intelligent Laboratory Systems*, vol. 100, no. 2, pp. 118–126, 2010.
- [7] S. de Jong, "SIMPLS: an alternative approach to partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 3, pp. 251–263, 1993.
- [8] J. González, D. Peña, and R. Romera, "A robust partial least squares regression method with applications," *Journal of Chemometrics*, vol. 23, no. 2, pp. 78–90, 2009.
- [9] M. Hubert and K. V. Branden, "Robust methods for partial least squares regression," *Journal of Chemometrics*, vol. 17, no. 10, pp. 537–549, 2003.
- [10] X. Bao and L. Dai, "Partial least squares with outlier detection in spectral analysis: a tool to predict gasoline properties," *Fuel*, vol. 88, no. 7, pp. 1216–1222, 2009.
- [11] F. Liu, Z. Mao, and W. Su, "Outlier detection for process control data based on a non-linear auto-regression hidden markov model method," *Transactions of the Institute of Measurement and Control*, vol. 34, no. 5, pp. 527–538, 2011.
- [12] F. Liu, M. Zhizhong, and L. Lei, "RBFN-based prediction for time series of process industries," *Journal of Northeastern University (Natural Science)*, vol. 31, no. 12, pp. 1693–1696, 2010.
- [13] B. Hu, P. Zheng, and J. Liang, "Multi-loop internal model controller design based on a dynamic PLS framework," *Chinese Journal of Chemical Engineering*, vol. 18, no. 2, pp. 277–285, 2010.
- [14] B. Hu, Z. Zhao, and J. Liang, "Multi-loop nonlinear internal model controller design under nonlinear dynamic PLS framework using ARX-neural network model," *Journal of Process Control*, vol. 22, no. 1, pp. 207–217, 2012.
- [15] Y. Lü and J. Liang, "Multi-loop constrained iterative model predictive control using ARX-PLS decoupling structure," *Chinese Journal of Chemical Engineering*, vol. 21, no. 10, pp. 1129–1143, 2013.
- [16] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [17] J. A. Bilmes, "What HMMs can do," *IEICE Transactions on Information and Systems*, vol. E89-D, no. 3, pp. 869–891, 2006.

- [18] S. J. Qin and T. J. McAvoy, "Nonlinear PLS modeling using neural networks," *Computers and Chemical Engineering*, vol. 16, no. 4, pp. 379–391, 1992.
- [19] S. Lakshminarayanan, S. L. Shah, and K. Nandakumar, "Modeling and control of multivariable processes: dynamic PLS approach," *AIChE Journal*, vol. 43, no. 9, pp. 2307–2322, 1997.
- [20] J. H. Lee, "Model predictive control: review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.
- [21] D. Laurí, J. V. Salcedo, S. García-Nieto, and M. Martínez, "Model predictive control relevant identification: multiple input multiple output against multiple input single output," *IET Control Theory & Applications*, vol. 4, no. 9, pp. 1756–1766, 2010.
- [22] A. Antoniou and W.-S. Lu, *Optimization: Algorithms and Engineering Applications*, Springer, New York, NY, USA, 2007.
- [23] I.-L. Chien, H.-P. Huang, and J.-C. Yang, "A simple multiloop tuning method for PID controllers with no proportional kick," *Industrial and Engineering Chemistry Research*, vol. 38, no. 4, pp. 1456–1468, 1999.
- [24] B. A. Ogunnaike, J. P. Lemaire, M. Morari, and W. H. Ray, "Advanced multivariable control of a pilot-plant distillation column," *AIChE Journal*, vol. 29, no. 4, pp. 632–640, 1983.
- [25] J. A. Gil and R. Romera, "On robust partial least squares (PLS) methods," *Journal of Chemometrics*, vol. 12, no. 6, pp. 365–378, 1998.
- [26] R. K. Wood and M. W. Berry, "Terminal composition control of a binary distillation column," *Chemical Engineering Science*, vol. 28, no. 9, pp. 1707–1717, 1973.