Pergamon

Computers chem. Engng Vol. 22, No. 4/5, pp. 503–514, 1998
© 1998 Elsevier Science Ltd. All rights reserved
Printed in Great Britain
0098-1354/98 $19.00+0.00

PII: S0098-1354(97)00262-7

Recursive PLS algorithms for adaptive data modeling

S. Joe Qin*

Department of Chemical Engineering, The University of Texas at Austin, Austin, Texas 78712, U.S.A.

(Received 17 June 1996; revised 25 September 1997)

Abstract

Partial least squares (PLS) regression is effectively used in process modeling and monitoring to deal with a large number of variables with collinearity. In this paper, several recursive partial least squares (RPLS) algorithms are proposed for on-line process modeling to adapt process changes and off-line modeling to deal with a large number of data samples. A block-wise RPLS algorithm is proposed with a moving window and forgetting factor adaptation schemes. The block-wise RPLS algorithm is also used off-line to reduce computation time and computer memory usage in PLS regression and cross-validation. As a natural extension, the recursive algorithm is extended to dynamic modeling and nonlinear modeling. An application of the block recursive PLS algorithm to a catalytic reformer is presented to adapt the model based on new data. © 1998 Elsevier Science Ltd. All rights reserved

Keywords: Partial least squares; recursive PLS; cross-validation; forgetting factors; chemical process modeling; dynamic modeling

NOMENCLATURE

$\mathbf{A}_i$	coefficient matrix in an ARX model
a_j, $b_{i,j}$	quadratic Volterra model coefficients
$\mathbf{B}$	diagonal matrix of inner model coefficients
$\mathbf{B}_i$	coefficient matrix in an FIR or ARX model
b_i	inner model coefficient
$\mathbf{C}$	$\in \Re^m \times p$, model coefficient matrix
$\mathbf{C}^{PLS}$	$\in \Re^m \times p$, regression coefficient matrix from PLS
$\mathbf{C}^{PLS}_{new}$	$\in \Re^m \times p$, updated regression coefficient matrix from PLS
$\mathbf{E}_i$	$\in \Re^n \times m$, residual matrix for $\mathbf{X}$
$\mathbf{F}_i$	$\in \Re^n \times p$, residual matrix for $\mathbf{Y}$
J	objective function in PLS
m	number of inputs in $\mathbf{X}$
n	number of samples in $\mathbf{X}$ and $\mathbf{Y}$
n_1	number of samples in $\mathbf{X}_1$ and $\mathbf{Y}_1$
n_u	maximum time lag for inputs in an ARX model
n_y	maximum time lag for outputs in an ARX model
$\mathbf{P}$	$\in \Re^m \times a$, loading matrix for $\mathbf{X}$
$\mathbf{p}_i$	$\in \Re^m$, loading vector for $\mathbf{X}$
p	number of outputs in $\mathbf{Y}$
$\mathbf{Q}$	$\in \Re^{p\times a}$, loading matrix for $\mathbf{Y}$
$\mathbf{q}_i$	$\in \Re^p$, loading vector for $\mathbf{Y}$
$\mathbf{r}_i$	$\in \Re^n$, residuals from inner models
r	rank of the input data matrix
s	number of data blocks
$\mathbf{T}$	$\in \Re^{n\times a}$, score matrix for $\mathbf{X}$
$\mathbf{t}_i$	$\in \Re^n$, score vector for $\mathbf{X}$
$\mathbf{u}_i$	$\in \Re^n$, score vector for $\mathbf{Y}$
$\mathbf{u}(k)$	input vector in an ARX or FIR model
$\mathbf{V}$	noise matrix
$\mathbf{v}(k)$	noise vector
$\mathbf{W}$	$\in \Re^m \times a$, weighting matrix in PLS
$\mathbf{w}_i$	$\in \Re^m$, weighting vector in PLS
w	window size of a moving window
$\mathbf{X}$	$\in \Re^{n\times m}$, input data matrix
$\mathbf{x}(k)$	sample vector for inputs
$\mathbf{Y}$	$\in \Re^{n\times p}$, output data matrix
ϵ	error tolerance for the residual
λ	forgetting factor
$(\bullet)^T$	transpose of a vector or matrix
$(\bullet)^+$	generalized inverse by PLS
$\|\bullet\|$	Frobenius norm of a matrix

* Author to whom correspondence should be addressed. e-mail: qin@che.utexas.edu.

1. Introduction

Since the pioneering work of Wold (1966), partial least squares (PLS) regression has been widely applied in chemometrics (Lindberg *et al.*, 1983; Wold *et al.*, 1984; Geladi and Kowalski, 1986; Fuller *et al.*, 1988; and Haaland and Thomas, 1988; Martin and Naes), steady state process modeling (Piovoso and Owens, 1991), dynamic modeling (Ricker, 1988; Wise and Ricker, 1990; MacGregor *et al.*, 1991), and process monitoring (MacGregor *et al.*, 1991; Negiz and Cinar, 1992; Nomikos and MacGregor, 1995). In typical process applications the input variables are highly correlated. While the ordinary least squares (OLS) regression gives rise to an ill-conditioned problem and large variance, the PLS regression provides a robust solution by orthogonal projection onto latent variables and regression on one-dimensional latent variables. A cross-validation procedure is often used to choose the number of latent variables or factors to use in a model so that the prediction variance is reduced. The final PLS regression is usually biased with reduced variance; as a result, the overall mean square error is minimized (Höskuldsson, 1988). Numerically, the PLS regression is related to singular value decomposition, which is discussed by Ricker (1988) for the single output case and by Kaspar and Ray (1993) for the multiple output case. These results help the user understand the PLS regression with more well known numerical methods.

In most of the PLS applications to date, the PLS regression is a batch-wise modeling approach. In other words, the data are collected and stored in a computer, then the PLS regression is carried out on the whole batch of data. While the batch type PLS circumvents the collinearity problem, it has limitations in the following situations. First, it is difficult to update a PLS model on-line using newly available data. While one could rebuild a new model based on merging the new data and old data, it is computationally inefficient because the old data are modeled repeatedly. Second, in the case of large data sets with many variables and data samples, which is often encountered in process data analysis, the batch PLS algorithm may run out of computer memory for a given computing platform. Third, the typical cross-validation procedure involves time-consuming and repetitive calculation by leaving out a subset of data and modeling on the remaining subsets. It is desirable to improve the computation efficiency by reusing the previous calculation in the procedure. In this paper, the basic recursive partial least squares (RPLS) algorithm proposed by Helland *et al.* (1992) and later modified by Qin (1993) is extended to block-wise RPLS with a moving window and forgetting factor schemes for steady state and dynamic process modeling. The RPLS algorithms can adapt the model based on new data and the old PLS model, thus avoid re-modeling the old data. The main contributions of the paper are summarized in the following aspects:

1. The RPLS algorithm is extended to a block-wise RPLS which builds a PLS sub-model on the new block of data and then combines with the old PLS model.
2. Both a moving window approach and a forgetting factor approach are proposed to adapt a PLS model.
3. The block-wise RPLS is applied to cross-validation in order to reduce computing time and memory usage in the case of large data sets.
4. Some vague treatments of the original RPLS algorithm by Helland *et al.* are clarified and mathematical proofs are provided. These include the treatment of the number of factors for model updating and the treatment of output residuals.

The basic PLS algorithm used in this paper is the NIPALS (nonlinear iterative partial least squares) algorithm (Geladi and Kowalski, 1986). The computation time of the proposed recursive PLS algorithms and that of a batch-wise PLS algorithm are compared on the basis that all algorithms are using the same basic PLS algorithm. Recent work by Dayal and MacGregor (1997) propose to use a faster variation of PLS algorithm, i.e. the kernel algorithm, to further speed up the computation. It is noted that faster variations of PLS algorithms can also be used with the recursive PLS algorithms proposed in this paper.

The remaining sections of the paper is organized as follows. Section 2 presents a recursive PLS algorithm proposed by Helland *et al.* (1992) with a few modifications and mathematical proofs. Section 3 discusses two block-wise recursive PLS algorithms based on a moving window approach and a forgetting factor approach. These two algorithms can be used to adapt process changes and correlation structure changes on-line. In the case of very large data sets for which computing memory or computing time become an issue, efficient cross-validation and final PLS modeling approaches are proposed in Section 4. In Section 5 the use of recursive PLS algorithms in dynamic modeling and nonlinear modeling is discussed. Section 6 presents an application of the block recursive PLS algorithm to model the product property of a catalytic reformer. The final Section 7 presents conclusions and discussions.

2. PLS and recursive PLS

2.1. *PLS regression*

In this section we briefly discuss the traditional batch PLS algorithm in order to derive the recursive PLS algorithm. Given a pair of input and output data matrices $\mathbf{X}$ and $\mathbf{Y}$ and assuming they are linearly related by

$$\mathbf{Y}=\mathbf{X}\mathbf{C}+\mathbf{V} \tag{1}$$

where $\mathbf{V}$ and $\mathbf{C}$ are noise and coefficient matrices, respectively, the PLS regression builds a linear model by decomposing matrices $\mathbf{X}$ and $\mathbf{Y}$ into bilinear terms,

$$\mathbf{X}=\mathbf{t}_1\mathbf{p}_1^T+\mathbf{E}_1 \tag{2}$$

$$\mathbf{Y}=\mathbf{u}_1\mathbf{q}_1^T+\mathbf{F}_1 \tag{3}$$

where $\mathbf{t}_1$ and $\mathbf{u}_1$ are latent score vectors of the first PLS factor, and $\mathbf{p}_1$ and $\mathbf{q}_1$ are corresponding loading vectors. All four vectors are determined by iteration with $\mathbf{t}_1$ and $\mathbf{u}_1$ being eigenvectors of $\mathbf{X}\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T$ and $\mathbf{Y}\mathbf{Y}^T\mathbf{X}\mathbf{X}^T$, respectively. Note that $\mathbf{XX}^T\mathbf{YY}^T$ is the transpose of $\mathbf{YY}^T\mathbf{XX}^T$ and vice versa; therefore, the two matrices have identical eigenvalues. The above two equations formulate a PLS outer model. The latent score vectors are then related by a linear inner model:

$$\mathbf{u}_1 = b_1\mathbf{t}_1 + \mathbf{r}_1 \tag{4}$$

where b_1 is a coefficient which is determined by minimizing the residual $\mathbf{r}_1$. After going through the first factor calculation, the second factor is calculated by decomposing the residuals $\mathbf{E}_1$ and $\mathbf{F}_1$ using the same procedure as for the first factor. This procedure is repeated until all specified factors are calculated. The overall PLS algorithm is summarized in Table 1 according to Geladi and Kowalski (1986) to introduce relations for further derivation. Note that a minor modification is made in this algorithm such that the latent variables $\mathbf{t}_h$ are normalized instead of $\mathbf{w}_h$ and $\mathbf{p}_h$. This modification makes it easier to derive the recursive PLS regression algorithm. As a result, the latent vectors $\mathbf{t}_h(h=1, 2, ...)$, are *orthonormal*.

The total number of factors required in the model is usually determined by cross-validation (Geladi and Kowalski, 1986), although elsewhere an F-test is suggested (Haaland *et al.*, 1988). A standard way of doing cross-validation is to divide the data into s subsets or folds, leave out a subset of data at a time, and build a model with the remaining subsets. The model is then tested on the subset which is not used in modeling. This procedure is repeated until every subset has been left out once. Summing up all the test errors for each factor, a *predicted error sum of squares* (PRESS) is resulted. The optimal number of factors is chosen as the location of the minimum PRESS error. The cross-validation method is computation intensive due to repeated modeling on a portion of the data. More details of the method can be found in Geladi and Kowalski (1986).

Table 1. A traditional batch-wise PLS algorithm

```
1. Scale X and Y to zero-mean and unit-variance.
       Initialize $\mathbf{E}_0 := \mathbf{X}$, $\mathbf{F}_0 := \mathbf{Y}$, and $h := 0$.
2. Let $h := h+1$ and take $\mathbf{u}_h$ as some column of $\mathbf{F}_{h-1}$.
3. Iterate the PLS outer model until it converges:
       $\mathbf{w}_h = \mathbf{E}_{h-1}^T\mathbf{u}_h/\mathbf{u}_h^T\mathbf{u}_h$
       $\mathbf{t}_h = \mathbf{E}_{h-1}\mathbf{w}_h/\|\mathbf{E}_{h-1}\mathbf{w}_h\|$
       $\mathbf{q}_h = \mathbf{F}_{h-1}^T\mathbf{t}_h/\|\mathbf{F}_{h-1}^T\mathbf{t}_h\|$
       $\mathbf{u}_h = \mathbf{F}_{h-1}\mathbf{q}_h$
4. Calculate the X-loadings:
       $\mathbf{p}_h = \mathbf{E}_{h-1}^T\mathbf{t}_h/\mathbf{t}_h^T\mathbf{t}_h = \mathbf{E}_{h-1}^T\mathbf{t}_h$
5. Find the inner model:
       $b_h = \mathbf{u}_h^T\mathbf{t}_h/\mathbf{t}_h^T\mathbf{t}_h = \mathbf{u}_h^T\mathbf{t}_h$
6. Calculate the residuals:
       $\mathbf{E}_h = \mathbf{E}_{h-1} - \mathbf{t}_h\mathbf{p}_h^T$
       $\mathbf{F}_h = \mathbf{F}_{h-1} - b_h\mathbf{t}_h\mathbf{q}_h^T$
7. Return to step 2 until all principal factors are calculated.
```

The robustness of a regression algorithm refers to the insensitivity of the model estimate to ill-conditioning and noise. The robustness of PLS vs OLS can be illustrated geometrically with Fig. 1, which depicts an extreme case of collinear and noisy data with two inputs and one output. All the input data are exactly collinear except for one data point, x, which is corrupted with noise. These data span a two-dimensional subspace **X**. The OLS approach in Fig. 1(a) projects the output **Y** orthogonally to **X**. However, since the data point x is corrupted with random noise which causes its location to be random, the orientation of the plane **X** is heavily affected by the location of x. As a result, the OLS projection $\hat{\mathbf{Y}}_{\mathrm{OLS}}$ is highly sensitive to the location of x, i.e. sensitive to noise. Fig. 1(b) shows the PLS model which requires one factor, i.e. one orthogonal projection to the one-dimensional subspace $\mathbf{t}_1$ in **X**. In this case, the PLS projection $\hat{\mathbf{Y}}_{\mathrm{PLS}}$ is not affected by the location of x, i.e. robust to noise. Although this example is idealized, it illustrates geometrically how PLS is more robust to noise and collinearity than OLS.

2.2. *Recursive PLS regression*

Industrial processes often experience time-varying changes, such as catalytic decaying, drifting, and degradation of efficiency. In these circumstances, a recursive algorithm is desirable to update the model based on new process data that reflect the process changes. Helland *et al.* (1992) introduced a recursive PLS regression algorithm which updates the model based on new data without increasing the size of data matrices. In this paper, we modify and extend the recursive PLS algorithm in the following aspects:

- Provide a recursive PLS algorithm that gives identical results to the traditional PLS by updating the model

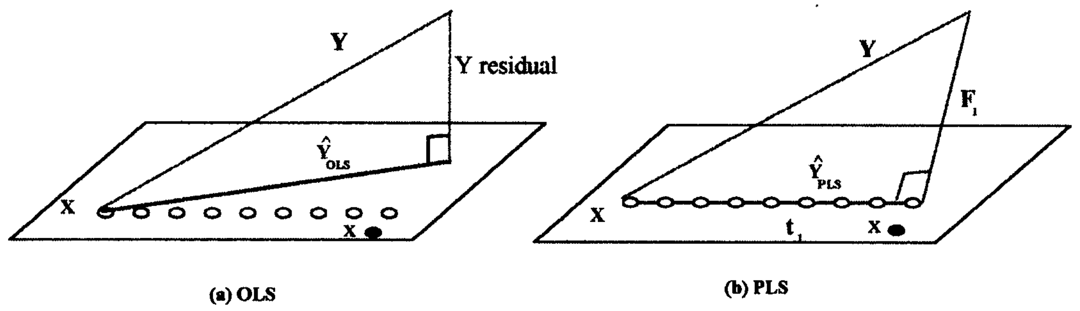
Y
Y residual
$\hat{\mathbf{Y}}_{\mathrm{OLS}}$
X
x
(a) OLS
Y
$\mathbf{F}_1$
$\hat{\mathbf{Y}}_{\mathrm{PLS}}$
X
$\mathbf{t}_1$
x
(b) PLS

Fig. 1. A geometric interpretation of PLS and OLS. The PLS method is robust to collinearity and observations corrupted with noise.

with the number of factors equal to the rank of the **X**. This number is typically larger than that required by cross-validation for prediction, as is shown in Lemma 1 below.
- Consider the case of rank deficient data **X** (Lemma 1) and provide a clear treatment for the output residual (Lemma 2). These points were not clearly considered in the original paper of Helland, *et al.*

Assume that a pair of data matrices $\{\mathbf{X},\mathbf{Y}\}$ has m input variables, p output variables, n samples. To derive the recursive PLS algorithm, we first present the following result.

Lemma 1. *If* $\mathrm{rank}(\mathbf{X})=r\leq m$, *then*

$$\mathbf{E}_r=\mathbf{E}_{r+1}=\ldots=\mathbf{E}_m=\mathbf{0}. \tag{13}$$

Proof of the lemma is given in Appendix A. This lemma indicates that the maximum number of factors does not exceed r. We use the following notation to represent $\{\mathbf{T},\mathbf{W},\mathbf{P},\mathbf{B},\mathbf{Q}\}$ is the PLS results of data $\{\mathbf{X},\mathbf{Y}\}$ by the PLS algorithm,

$$\{\mathbf{X},\mathbf{Y}\}\xrightarrow{\text{PLS}}\{\mathbf{T},\mathbf{W},\mathbf{P},\mathbf{B},\mathbf{Q}\} \tag{14}$$

where

$$\mathbf{T}=[\mathbf{t}_1,\mathbf{t}_2,\ldots,\mathbf{t}_r]$$
$$\mathbf{W}=[\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_r]$$
$$\mathbf{P}=[\mathbf{p}_1,\mathbf{p}_2,\ldots,\mathbf{p}_r]$$
$$\mathbf{B}=diag\{b_1,b_2,\ldots,b_r\}$$
$$\mathbf{Q}=[\mathbf{q}_1,\mathbf{q}_2,\ldots,\mathbf{q}_r]$$

Here we include all possible number of factors equal to the rank of the input matrix, r. This is required by the result of Lemma 1.

(11) and (12) can be rearranged as

$$\mathbf{X}=\mathbf{E}_0=\mathbf{T}\,\mathbf{P}^T+\mathbf{E}_r=\mathbf{T}\,\mathbf{P}^T \tag{15}$$

$$\mathbf{Y}=\mathbf{T}\mathbf{B}\mathbf{Q}^T+\mathbf{F}_r \tag{16}$$

It should be noted that the residual matrix $\mathbf{F}_r$ is generally not zero unless **Y** is exactly in the range space of **X**. However, we can show that $\mathbf{F}_r$ is orthogonal to the scores, as summarized in the following lemma.

Lemma 2. *The output residual* $\mathbf{F}_i$ *is orthogonal to the scores of previous factors* $\mathbf{t}_h$, *i.e.*

$$\mathbf{t}_h^T\mathbf{F}_i=\mathbf{0},\ \text{for } i\geq h \tag{17}$$

Proof of Lemma 2 is given in Appendix A. By minimizing the squared residuals, $\|\mathbf{Y}-\mathbf{X}\mathbf{C}\|^2$, we have

$$(\mathbf{X}^T\mathbf{X})\mathbf{C}=\mathbf{X}^T\mathbf{Y}. \tag{18}$$

The PLS regression coefficient matrix is:

$$\mathbf{C}^{PLS}=(\mathbf{X}^T\mathbf{X})^+\mathbf{X}^T\mathbf{Y} \tag{19}$$

where $(\bullet)^+$ denotes the generalized inverse defined by the PLS algorithm. An explicit expression of the PLS regression coefficient matrix is (Höskuldsson, 1988)

$$\mathbf{C}^{PLS}=\mathbf{W}^*\mathbf{B}\mathbf{Q}^T \tag{20}$$

where

$$\mathbf{W}^*=[\mathbf{w}_1^*,\mathbf{w}_2^*,\ldots,\mathbf{w}_m^*] \tag{21}$$

and

$$\mathbf{w}_i^*=\prod_{h=1}^{i-1}(\mathbf{I}_m-\mathbf{w}_h\mathbf{p}_h^T)\mathbf{w}_i. \tag{22}$$

When a new data pair $\{\mathbf{X}_1,\mathbf{Y}_1\}$ is available and we are interested in updating the PLS model using the augmented data matrices

$$\mathbf{X}_{new}=\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix}\text{ and }\mathbf{Y}_{new}=\begin{bmatrix}\mathbf{Y}\\ \mathbf{Y}_1\end{bmatrix},$$

the resulting PLS model is

$$\mathbf{C}_{new}^{PLS}=\left(\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix}\right)^+\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{Y}\\ \mathbf{Y}_1\end{bmatrix}. \tag{23}$$

Since columns of **T** are mutually orthonormal, the following relation can be derived using (15) and (16) and Lemma 2,

$$\mathbf{X}^T\mathbf{X}=\mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{P}^T=\mathbf{P}\mathbf{P}^T \tag{24}$$

$$\mathbf{X}^T\mathbf{Y}=\mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{B}\mathbf{Q}^T+\mathbf{P}\mathbf{T}^T\mathbf{F}_r=\mathbf{P}\mathbf{B}\mathbf{Q}^T. \tag{25}$$

Therefore, (23) becomes,

$$\mathbf{C}_{new}^{PLS}=\left(\begin{bmatrix}\mathbf{P}^T\\ \mathbf{X}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{P}^T\\ \mathbf{X}_1\end{bmatrix}\right)^+\begin{bmatrix}\mathbf{P}^T\\ \mathbf{X}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{B}\mathbf{Q}^T\\ \mathbf{Y}_1\end{bmatrix}. \tag{26}$$

By comparing (26) with (23), we derive the following theorem.

Theorem 1. *Given a PLS model,* $\{\mathbf{X},\mathbf{Y}\}\xrightarrow{\text{PLS}}\{\mathbf{T},\mathbf{W},\mathbf{P},\mathbf{B},\mathbf{Q}\}$ *and a new data pair* $\{\mathbf{X}_1,\mathbf{Y}_1\}$, *performing PLS regression on data pair*

$$\begin{bmatrix}\mathbf{P}^T\\ \mathbf{X}_1\end{bmatrix},\begin{bmatrix}\mathbf{B}\mathbf{Q}^T\\ \mathbf{Y}_1\end{bmatrix}$$

results in the same regression model as performing PLS regression on data pair

$$\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix},\begin{bmatrix}\mathbf{Y}\\ \mathbf{Y}_1\end{bmatrix}.$$

It is easy to prove this theorem by comparing (26) with (23). Instead of using old data and new data to update the PLS model, the RPLS can update the model using the old model and new data. The RRPLS algorithm is summarized in Table 2.

Remark 1. *It is necessary in step 2 to check whether* $\|\mathbf{E}_r\|\leq\epsilon$, *or the residual is essential zero. Otherwise, (24) is not valid. Note that r can be different during the course of adaptation as more data are available (usually increasing). Helland* et al. *(1992) did not consider this point in their original RPLS algorithm which results in numerical errors in their example, although the errors are minor.*

Remark 2. *Lemma 2 is necessary to guarantee that*

Table 2. The recursive PLS (RPLS) algorithm

1. Formulate the data matrices $\{\mathbf{X},\mathbf{Y}\}$. Scale the data to zero mean and unit variance, or as otherwise specified with a set of weights.

2. Derive a PLS model using the algorithm in Table 1: $\{\mathbf{X},\mathbf{Y}\} \xrightarrow{\text{PLS}} \{\mathbf{T},\mathbf{W},\mathbf{P},\mathbf{B},\mathbf{Q}\}$. Carry out the algorithm until $\|\mathbf{E}_r\| \leq \varepsilon$ ($\varepsilon>0$ is the error tolerance). This means that more factors are calculated than that required in cross-validation to make theorem 1 hold.

3. When a new pair of data, $\{\mathbf{X}_1,\mathbf{Y}_1\}$, is available, scale it the same way as it was done in step 1. Formulate $\mathbf{X}=\begin{bmatrix}\mathbf{P}^T\\ \mathbf{X}_1\end{bmatrix}$, $\mathbf{Y}=\begin{bmatrix}\mathbf{BQ}^T\\ \mathbf{Y}_1\end{bmatrix}$ and return to step 2.

the output residual has no effect on the recursive algorithm.

If we define the number of rows of the data pair as the PLS run-size, the RPLS updates the model with a PLS run-size of $(r+n_1)$, while the regular PLS would update the model with a run-size of $(n+n_1)$. One can easily see that the RPLS algorithm is much more efficient than the regular PLS if $n \gg r$. Note that this is a typical case in process modeling and monitoring where tens of thousands of data samples are available for about a few dozens of process variables.

It should be noted that the recursive PLS algorithm includes the maximum possible number of PLS factors, r. However, to use the model for prediction, the number of factors is determined by cross-validation and is usually less than r. The purpose of carrying more factors than currently needed is not only to satisfy Theorem 1, but also to prepare for process changes in degrees of freedom or variability, which dictate the number of factors to vary. For example, when some variables were correlated in the past, but are not correlated given new data at present, an increase in the number of factors is required.

2.3. *RPLS for data with non-zero mean*

The above RPLS algorithm is derived with the assumption that the data $\mathbf{X}$ and $\mathbf{Y}$ are scaled to zero mean and unit variance. As new data are available, the mean and variance will change over time. Therefore, the scaling procedure in step 1 of the RPLS will not make the new data zero mean and unit variance. The role of unit variance scaling in PLS is to put equal weight on each input variable based on its variance, but the algorithm will still work if the data are not scaled to unit variance. This makes the RPLS algorithm work even though the variance may change over time.

However, if the mean of each variable in the data matrices is not zero, the input–output relationship has to be modified with the following general linear relationship,

$$\mathbf{y}_i=\mathbf{C}\mathbf{x}_i+\mathbf{d}=\begin{bmatrix}\mathbf{C} & \mathbf{d}\end{bmatrix}\begin{bmatrix}\mathbf{x}_i^T & 1\end{bmatrix}^T \quad (27)$$

where $\mathbf{x}_i$ and $\mathbf{y}_i$ represent the ith rows of $\mathbf{X}$ and $\mathbf{Y}$, respectively, and $\mathbf{d} \in \Re^p$ is a vector of intercepts for the general linear model. Therefore, to model data with non-zero mean, we simply apply the RPLS algorithm on the following data pair,

$$\left\{\begin{bmatrix}\mathbf{X} & \frac{1}{\sqrt{n-1}}\mathbf{1}\end{bmatrix}, \mathbf{Y}\right\},$$

where $\mathbf{1} \in \Re^n$ is a vector whose elements are all one. The scaling factor $\frac{1}{\sqrt{n-1}}$ is to make the norm of $\frac{1}{\sqrt{n-1}}\mathbf{1}$ comparable to the norm of the columns of $\mathbf{X}$, as the PLS algorithm is sensitive to how each input variable is scaled.

The above treatment for non-zero mean data is consistent with that commonly used in linear regression. The only difference one can expect is that the PLS algorithm is biased linear regression, making the estimate of the intercept $\mathbf{d}$ also biased. However, the bias is introduced to reduce the variance and minimize the overall mean squared error. In the limit of r factors being used in the PLS model, the PLS regression approaches OLS regression. Another way to interpret the treatment is that PLS is equivalent to a conjugate gradient approach to linear regression (Wold *et al.*, 1984). The effect of this treatment will be demonstrated with an application later in this paper.

3. Block-wise RPLS and on-line adaptation schemes

3.1. *Block-wise RPLS*

Theorem 1 gives a RPLS algorithm which updates the model as soon as some new samples are available. It may be desirable not to update the model until significant amount of data are collected and the process has gone through significant changes. In this case we can accumulate a new block of data, derive a PLS sub-model on the new data block, and then combine it with the existing model. Assuming the PLS sub-model on the new data block is,

$$\{\mathbf{X}_1,\mathbf{Y}_1\} \xrightarrow{\text{PLS}} \{\mathbf{T}_1,\mathbf{W}_1,\mathbf{P}_1,\mathbf{B}_1,\mathbf{Q}_1\} \quad (28)$$

The PLS regression can be calculated from (23) as follows,

$$\mathbf{C}_{new}^{PLS}=(\mathbf{X}_{new}^T\mathbf{X}_{new})^+\mathbf{X}_{new}^T\mathbf{Y}_{new} \quad (29)$$

$$=(\mathbf{PP}^T+\mathbf{P}_1\mathbf{P}_1^T)^+(\mathbf{PBQ}^T+\mathbf{P}_1\mathbf{B}_1\mathbf{Q}_1^T)$$

$$=\left(\begin{bmatrix}\mathbf{P}^T\\ \mathbf{P}_1^T\end{bmatrix}^T\begin{bmatrix}\mathbf{P}^T\\ \mathbf{P}_1^T\end{bmatrix}\right)^+\begin{bmatrix}\mathbf{P}^T\\ \mathbf{P}_1^T\end{bmatrix}^T\begin{bmatrix}\mathbf{BQ}^T\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}$$

Therefore, a PLS model based on two data blocks is equivalent to combining the two sub-models. We have the following theorem.

Theorem 2. *Assuming two PLS models as given in (14) and (28), performing PLS regression on*

$$\begin{bmatrix}\mathbf{P}^T\\ \mathbf{P}_1^T\end{bmatrix},\begin{bmatrix}\mathbf{BQ}^T\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}$$

results in the same regression model as performing PLS regression on the data pair

$$\begin{bmatrix}\mathbf{X}\\ \mathbf{X}_1\end{bmatrix},\begin{bmatrix}\mathbf{Y}\\ \mathbf{Y}_1\end{bmatrix}.$$

As an extension, if there are s blocks of data, and

$$\{\mathbf{X}_i,\mathbf{Y}_i\}\overset{\text{PLS}}{\longrightarrow}\{\mathbf{T}_i,\mathbf{W}_i,\mathbf{P}_i,\mathbf{B}_i,\mathbf{Q}_i\};\ i=1,2,\ldots,s \quad (30)$$

performing PLS regression on all data is equivalent to performing PLS regression on the following pair of matrices

$$\begin{bmatrix}\mathbf{P}_1^T\\ \mathbf{P}_2^T\\ \vdots\\ \mathbf{P}_s^T\end{bmatrix},\begin{bmatrix}\mathbf{B}_1\mathbf{Q}_1^T\\ \mathbf{B}_2\mathbf{Q}_2^T\\ \vdots\\ \mathbf{B}_s\mathbf{Q}_s^T\end{bmatrix}$$

Theorem 2 can be proven by comparing (23) and (29) for two blocks of data, and similar results can be obtained with s blocks. The block-wise RPLS algorithm can be summarized in Table 3.

The procedure of this block-wise RPLS algorithm is illustrated in Fig. 2. Updating the PLS model involves performing PLS on the existing model and the new sub-model, which requires much less computation than updating the PLS using the entire data set. The block-wise RPLS algorithm computes a sub-model with a run-size of n_1 and a updated model with a run-size of $(2r)$. The block RPLS algorithm has its computational advantage for on-line adaptation with a moving window and in cross-validation for off-line PLS modeling, which will be demonstrated in the following sections.

3.2. *On-line adaptation with a moving window*

To adequately adapt process changes, it is desirable to exclude extremely old data because the process has changed. A moving window approach can be used to incorporate new data and drop out old data. The objective function for the PLS algorithm with a moving window can be written as

Table 3. The block-wise RPLS algorithm

1. Formulate the data matrices $\{\mathbf{X},\mathbf{Y}\}$. Scale the data to zero mean and unit variance, or as otherwise specified.
2. Derive a PLS model using the algorithm in Table 1: $\{\mathbf{X},\mathbf{Y}\}\overset{\text{PLS}}{\longrightarrow}\{\mathbf{T},\mathbf{W},\mathbf{P},\mathbf{B},\mathbf{Q}\}$. Carry out the algorithm until $\mathbf{E}_r=\mathbf{0}$.
3. When a new pair of data, $\{\mathbf{X}_1,\mathbf{Y}_1\}$, is available, scale it the same way as it was done in step 1. Perform PLS to derive a sub-model: $\{\mathbf{X}_1,\mathbf{Y}_1\}\overset{\text{PLS}}{\longrightarrow}\{\mathbf{T}_1,\mathbf{W}_1,\mathbf{P}_1,\mathbf{B}_1,\mathbf{Q}_1\}$.
4. Formulate $\mathbf{X}=\begin{bmatrix}\mathbf{P}^T\\ \mathbf{P}_1^T\end{bmatrix}$, $\mathbf{Y}=\begin{bmatrix}\mathbf{BQ}^T\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}$ and return to step 2.

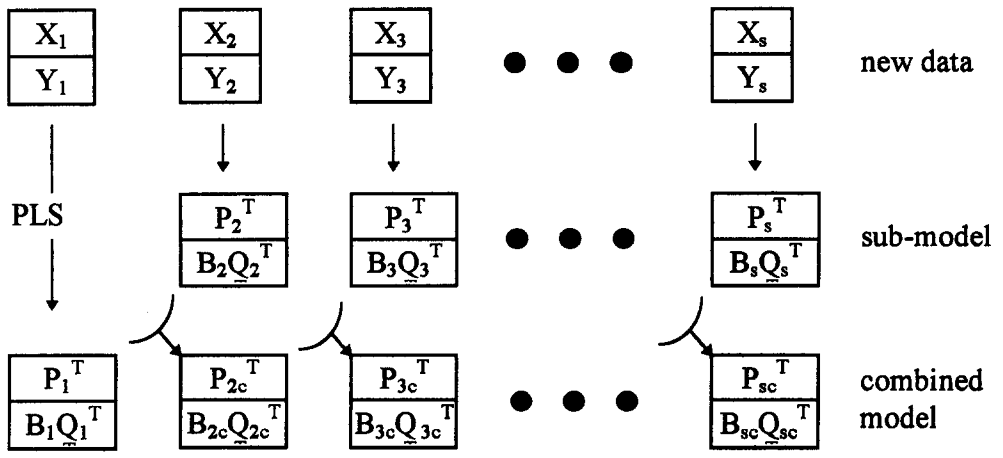

Fig. 2. A recursive process for the block-wise RPLS algorithm.

$$J_{s,w}=\left\|\begin{bmatrix}\mathbf{Y}_s\\ \mathbf{Y}_{s-1}\\ \vdots\\ \mathbf{Y}_{s-w+1}\end{bmatrix}-\begin{bmatrix}\mathbf{X}_s\\ \mathbf{X}_{s-1}\\ \vdots\\ \mathbf{X}_{s-w+1}\end{bmatrix}\mathbf{C}\right\|^2 \tag{31}$$

$$=\sum_{i=s-w+1}^{s}\left\|\mathbf{Y}_i-\mathbf{X}_i\mathbf{C}\right\|^2$$

$$=\sum_{i=s-w+1}^{s}\left\|\mathbf{T}_i(\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C})+\mathbf{F}_{ri}\right\|^2$$

$$=\sum_{i=s-w+1}^{s} trace\{[\mathbf{T}_i(\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C})+\mathbf{F}_{ri}]^T$$
$$[\mathbf{T}_i(\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C})+\mathbf{F}_{ri}]\}$$

where w is the number of blocks in the window and s represents the current block of data. By using Lemma 2,

$$\mathbf{T}_i^T\mathbf{F}_{ri}=\mathbf{0} \tag{32}$$

and $\mathbf{T}_i^T\mathbf{T}_i=\mathbf{I}$, we obtain,

$$J_{s,w}=\sum_{i=s-w+1}^{s} trace\{[\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C}]^T[\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C}]\}$$
$$+trace\{\mathbf{F}_{ri}^T\mathbf{F}_{ri}\} \tag{33}$$

$$=\sum_{i=s-w+1}^{s}\left\|\mathbf{B}_i\mathbf{Q}_i^T-\mathbf{P}_i^T\mathbf{C}\right\|^2+\left\|\mathbf{F}_{ri}\right\|^2$$

$$=\left\|\begin{bmatrix}\mathbf{B}_s\mathbf{Q}_s^T\\ \mathbf{B}_{s-1}\mathbf{Q}_{s-1}^T\\ \vdots\\ \mathbf{B}_{s-w+1}\mathbf{Q}_{s-w+1}^T\end{bmatrix}-\begin{bmatrix}\mathbf{P}_s^T\\ \mathbf{P}_{s-1}^T\\ \vdots\\ \mathbf{P}_{s-w+1}^T\end{bmatrix}\mathbf{C}\right\|^2+\sum_{i=s-w+1}^{s}\left\|\mathbf{F}_{ri}\right\|^2$$

Since the second term on the right hand side of the above equation is a constant, it can be dropped out of the objective function. Therefore, minimizing the objective function in (31) is equivalent to minimizing that in (33), except that the number of rows in (33) can be much fewer than that in (31). We can simply perform PLS regression on the following pair of matrices

$$\begin{bmatrix}\mathbf{P}_s^T\\ \mathbf{P}_{s+1}^T\\ \vdots\\ \mathbf{P}_{s-w+1}^T\end{bmatrix},\begin{bmatrix}\mathbf{B}_s\mathbf{Q}_s^T\\ \mathbf{B}_{s+1}\mathbf{Q}_{s+1}^T\\ \vdots\\ \mathbf{B}_{s-w+1}\mathbf{Q}_{s-w+1}\end{bmatrix}$$

as the input and output matrices, respectively. When a new block of data ($s+1$) is available, a PLS sub-model is first derived to obtain $\mathbf{P}_{s+1}^T$ and $\mathbf{B}_{s+1}\mathbf{Q}_{s+1}^T$. Then they are augmented into the top row of the above matrices and the bottom row is dropped out. The window size w, which is the number of blocks, controls how old the data that are kept in the window. The smaller the window size, the faster the model adapts new data and forgets old data. Assuming each data block has n_1 samples, the block-wise RPLS update the model with a run-size of (rw), while the regular PLS would update the model for a run-size of n_1w. Clearly, the RPLS algorithm with a moving window is advantageous when $n_1>r$.

3.3. *Adaptation with forgetting factors*

An alternative approach to on-line adaptation is to use forgetting factors. The use of forgetting factors is well known in recursive least squares (Ljung, 1987). Here we incorporate a forgetting factor in the block-wise RPLS algorithm to adapt process changes. To derive the recursive regression, we start the PLS modeling on the first data block by minimizing (from (33) after ignoring the constant term):

$$J_1=\|\mathbf{B}_1\mathbf{Q}_1^T-\mathbf{P}_1^T\mathbf{C}\|^2 \tag{34}$$

With s blocks of data available, we minimize the following objective function with a forgetting factor,

$$J_{s,\lambda}=\left\|\begin{bmatrix}1&&&\\&\lambda&&\\&&\cdots&\\&&&\lambda^{s-1}\end{bmatrix}\left(\begin{bmatrix}\mathbf{B}_s\mathbf{Q}_s^T\\ \mathbf{B}_{s-1}\mathbf{Q}_{s-1}^T\\ \vdots\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}-\begin{bmatrix}\mathbf{P}_s^T\\ \mathbf{P}_{s-1}^T\\ \vdots\\ \mathbf{P}_1^T\end{bmatrix}\right)\mathbf{C})\right\|^2 \tag{35}$$

$$=\lambda^2\left\|\begin{bmatrix}1&&&\\&\lambda&&\\&&\cdots&\\&&&\lambda^{s-2}\end{bmatrix}\left(\begin{bmatrix}\mathbf{B}_{s-1}\mathbf{Q}_{s-1}^T\\ \mathbf{B}_{s-2}\mathbf{Q}_{s-2}^T\\ \vdots\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}-\begin{bmatrix}\mathbf{P}_{s-1}^T\\ \mathbf{P}_{s-2}^T\\ \vdots\\ \mathbf{P}_1^T\end{bmatrix}\right)\mathbf{C}\right\|^2+\|\mathbf{B}_s\mathbf{Q}_s^T-\mathbf{P}_s^T\mathbf{C}\|^2$$

$$=\lambda^2J_{s-1,\lambda}+\|\mathbf{B}_s\mathbf{Q}_s^T-\mathbf{P}_s^T\mathbf{C}\|^2$$

where $0<\lambda\leq1$ is the forgetting factor. $J_{s-1,\lambda}$ is the objective function at step $s-1$. This expression indicates that the weights on old data blocks decay exponentially. A smaller λ will forget old data faster. Assuming at step $s-1$ we have a combined model $\{\mathbf{P}_{sc}^T,\mathbf{B}_{sc}\mathbf{Q}_{sc}^T\}$, according to Theorem 2, (35) can be rewritten as

$$J_{s,\lambda}=\lambda^2\|\mathbf{B}_{sc}\mathbf{Q}_{sc}^T-\mathbf{P}_{sc}^T\mathbf{C}\|^2+\|\mathbf{B}_s\mathbf{Q}_s^T-\mathbf{P}_s^T\mathbf{C}\|^2 \tag{36}$$

$$=\left\|\begin{bmatrix}\mathbf{B}_s\mathbf{Q}_s^T\\ \lambda\mathbf{B}_{sc}\mathbf{Q}_{sc}^T\end{bmatrix}-\begin{bmatrix}\mathbf{P}_s^T\\ \lambda\mathbf{P}_{sc}^T\end{bmatrix}\mathbf{C}\right\|^2$$

Therefore, the PLS model at step s can be obtained by performing PLS using

$$\begin{bmatrix}\mathbf{P}_s^T\\ \lambda\mathbf{P}_{sc}^T\end{bmatrix}$$

as the input matrix and

$$\begin{bmatrix}\mathbf{B}_s\mathbf{Q}_s^T\\ \lambda\mathbf{B}_{sc}\mathbf{Q}_{sc}^T\end{bmatrix}$$

as the output matrix. To update a RPLS model with a forgetting factor, one simply needs to derive a sub-model on the current data block, then combines it with the old model with a forgetting factor. The computation effort in updating the model is equivalent to performing a PLS regression with a run-size $2r$.

The forgetting factor approach is computationally more efficient than the moving window approach. Table 4 compares the computation load in terms of PLS run-sizes for the batch PLS, recursive PLS, block RPLS, block RPLS with moving windows, and block RPLS with forgetting factors. Typically, $n_l > r$ and $s > w$. Therefore, the computation load is significantly reduced in the RPLS and the block RPLS with forgetting factors.

4. Cross-validation and final PLS using block RPLS

In process applications, the number of data samples available for modeling is often very large. In this case, we divide the data into s blocks and perform leave-one-block-out cross-validation. After the number of factors is determined through cross-validation, a final PLS model is obtained by performing PLS regression on all available data. Since the regular cross-validation involves modeling the data repeatedly, it is computationally inefficient. In this section, we use the block RPLS to reduce the computation load in cross-validation and final PLS modeling.

Fig. 3 illustrates the use of block RPLS for cross-validation and final PLS modeling to improve the computation efficiency. First, the data are divided into s blocks, as in the regular cross-validation. Then a sub-model is built for each block using PLS regression. Third, we calculate the PRESS error by the leave-one-block-out approach. Assuming we leave the ith block out and build a PLS model on the remaining blocks, the following objective function is minimized (similar to (33)),

Table 4. The PLS run-sizes for the batch PLS, recursive PLS, block RPLS, block RPLS with moving windows, and block RPLS with forgetting factors*

	Batch PLS	Recursive PLS	Block RPLS	Block RPLS with moving windows	Block RPLS with forgetting factors
Sub-model	None	None	n_l	n_l	n_l
Update	$s*n_l$	$r+n_l$	$s*r$	$w*r$	$2*r$

n_l: number of samples in a block; r: rank of the input data matrix; s: number of blocks; w: window size in blocks).

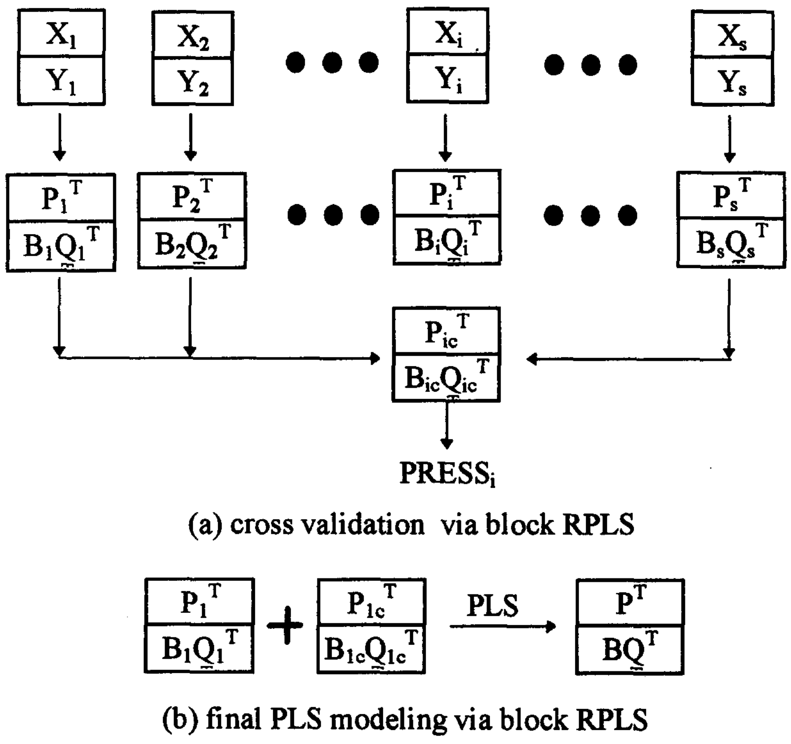

Fig. 3. Block diagrams for cross-validation and final PLS modeling using block-wise RPLS algorithm.

$$J_{ic}=\sum_{\substack{j=1\\j\neq i}}^{s}\|\mathbf{B}_j\mathbf{Q}_j^T-\mathbf{P}_j^T\mathbf{C}\|^2 \qquad (37)$$

which means that we build a PLS model by combining all sub-models except the ith one,

$$\mathbf{C}_{ic}^{PLS}=\left(\sum_{\substack{j=1\\j\neq i}}^{s}\mathbf{P}_j\mathbf{P}_j^T\right)^+\left(\sum_{\substack{j=1\\j\neq i}}^{s}\mathbf{P}_j\mathbf{B}_j\mathbf{Q}_j^T\right) \qquad (38)$$

where $\mathbf{C}_{ic}^{PLS}$ denotes a PLS model derived from all data but the ith block. By leaving out each block in turn, the cross-validated PRESS corresponding to the number of factors is

$$\text{PRESS}(h)=\sum_{i=1}^{s}\text{PRESS}_i=\sum_{i=1}^{s}\|\mathbf{Y}_i-\mathbf{X}_i\mathbf{C}_{ic}^{PLS}\|^2 \quad (39)$$

The number of factors that gives minimum PRESS is used in the final PLS modeling.

The final PLS model can be obtained by simply performing PLS regression on an intermediate model derived in the process of cross-validation. For example, assuming leaving out $\{\mathbf{X}_1,\mathbf{Y}_1\}$ results in a PLS model $\{\mathbf{P}_{1c}^T,\mathbf{B}_{1c}\mathbf{Q}_{1c}^T\}$, the final PLS model can be derived by performing PLS regression on

$$\begin{bmatrix}\mathbf{P}_{1c}^T\\ \mathbf{P}_1^T\end{bmatrix},\begin{bmatrix}\mathbf{B}_{1c}\mathbf{Q}_{1c}^T\\ \mathbf{B}_1\mathbf{Q}_1^T\end{bmatrix}$$

In both cross-validation and final PLS modeling, the amount of computation is significantly reduced for modeling a large number of data samples.

4.1. Example 1.

To demonstrate the effectiveness of the block RPLS algorithm for cross-validation and final PLS modeling, we assume there are 20 blocks of data and each block has 1000 samples. There are overall 10 input variables and one output variable. Assume the input data matrix is full rank. The computation involved in cross-validation and final PLS modeling using the block RPLS algorithm is given in Table 5 with a comparison with the regular cross-validation and PLS approach. Since there is only one output variable, The PLS outer modeling involves no iteration. As a result, the amount of calculation can be exactly measured by the PLS run-size. It can be seen from Table 5 that the amount of computation is significantly reduced by using the block RPLS algorithm.

In the case of multiple output variables, iteration for outer modeling is required. In each iteration, the computation effort is also proportional to the size of the PLS run. In this case, it is somewhat difficult to estimate the exact computation effort in a PLS run due to the unknown number of iterations. However, one can expect that the number of iterations in a smaller run-size would typically be smaller than that in a bigger run-size. Therefore, we can still use the run-size as a measure of computation effort.

An additional feature of the block RPLS algorithm is that it requires to put only a block data in the memory to facilitate the computation. This is advantageous when there are a large number of samples and the computer will run out of memory if a regular PLS algorithm is used. The block RPLS algorithm can avoid the memory shortage problem by building a sub-model for each block and then build the final model by combining sub-models.

5. Dynamic and nonlinear RPLS regression

5.1. Dynamic process modeling

One type of dynamic model is the auto-regressive model with exogenous inputs (ARX, Ljung, 1987),

$$\mathbf{y}(k)=\sum_{i=1}^{n_y}\mathbf{A}_i\mathbf{y}(k-i)+\sum_{j=1}^{n_u}\mathbf{B}_j\mathbf{u}(k-j)+\mathbf{v}(k) \qquad (40)$$

where $\mathbf{y}(k)$, $\mathbf{u}(k)$ and $\mathbf{v}(k)$ are the process output, input, and noise vectors, respectively, with appropriate dimensions for multi-input-multi-output systems. $\mathbf{A}_i$ and $\mathbf{B}_j$ are matrices of model coefficients to be identified. n_y and n_u are time lags for the output and input, respectively. In order for the PLS method to build an ARX model, the following vector of variables is defined,

$$\mathbf{x}^T(k)=[\mathbf{y}^T(k-1),\mathbf{y}^T(k-2),\ldots,\mathbf{y}^T(k-n_y),\mathbf{u}^T(k-1),\mathbf{u}^T(k-2),\ldots,\mathbf{u}^T(k-n_u)] \qquad (41)$$

whose dimension is denoted as m. Then two data matrices can be formulated as follows assuming the number of data records is n,

$$\mathbf{X}=[\mathbf{x}(1),\mathbf{x}(2),\ldots,\mathbf{x}(n)]^T\in\Re^{n\times m} \qquad (42)$$

$$\mathbf{Y}=[\mathbf{y}(1),\mathbf{y}(2),\ldots,\mathbf{y}(n)]^T\in\Re^{n\times p} \qquad (43)$$

where p is the dimension of output vector $\mathbf{y}(k)$. Defining all unknown parameters in the ARX model as,

$$\mathbf{C}=[\mathbf{A}_1,\mathbf{A}_2,\ldots,\mathbf{A}_{n_y},\mathbf{B}_1,\mathbf{B}_2,\ldots,\mathbf{B}_{n_u}]^T\in\Re^{m\times p} \qquad (44)$$

Eq. (40) can be re-written as

$$\mathbf{y}(k)=\mathbf{C}^T\mathbf{x}(k)+\mathbf{v}(k) \qquad (45)$$

and the two data matrices $\mathbf{Y}$ and $\mathbf{X}$ can be related as

$$\mathbf{Y}=\mathbf{XC}+\mathbf{V} \qquad (46)$$

The RPLS algorithms developed in this paper can be readily applied.

It should be noted that the ARX model derived from PLS algorithms is inherently an equation error approach (or series–parallel scheme) in system identification (Ljung, 1987). It has been reported by many researchers (e.g. Qin and McAvoy, 1996) that the ARX model with series–parallel identification scheme tends to emphasize auto-regression terms with poor long-term prediction

Table 5. Comparison of the block-wise RPLS and batch PLS computation load in cross-validation and final PLS modeling

	Traditional PLS	RPLS
CV	20 PLS runs of size 19,000	20 PLS runs of size 1000; 20 PLS runs of size 190
Final model	1 PLS run of size 20,000	1 PLS run of size 20

accuracy. However, a finite impulse response (FIR) model is often preferred and is applicable for stable processes, which can be described as

$$\mathbf{y}(k)=\sum_{j=1}^{N}\mathbf{B}_j\mathbf{u}(k-j)+\mathbf{v}(k) \tag{47}$$

where N is the truncation number that corresponds to the process settling time. Similar to the ARX model, two data matrices **X** and **Y** can be arranged accordingly. It is straight forward to apply the RPLS algorithms to this class of models.

5.2. *Nonlinear process modeling*

Traditional PLS algorithms have been extended to nonlinear modeling and data analysis. There are generally two approaches to extending the traditional PLS to include nonlinearity. One approach is to use nonlinear inner models, such as polynomials (Wold *et al.*, 1989) and neural networks (Qin and McAvoy, 1992; Holcomb and Morari, 1992; Frank, 1994). Another approach is to augment the input matrix with nonlinear functions of the input variables. For example, one may use quadratic combinations of the inputs as additional input to the model to build nonlinearity.

Since the RPLS algorithms proposed in this paper make use of the linear property of the PLS inner models, it is difficult to develop a nonlinear RPLS algorithm with nonlinear inner relations. However, one can always augment the input with nonlinear functions of the inputs to introduce nonlinearity in the model. For example, it is straight forward to include quadratic terms in the input matrix, as it is done in the traditional PLS regression (Wold *et al.*, 1989). If both quadratic inputs and a dynamic FIR formulation is used, the model format for a single-input-single-output process can be represented as,

$$y(k)=y_0+\sum_{j=1}^{N}a_ju(k-j)+\sum_{i=1}^{N}\sum_{j=1}^{N}b_{i,j}u(k-i)u(k-j)+\mathrm{v}(k) \tag{48}$$

where the bias term y_0 is required even though the input and output are scaled to zero mean. The resulting model is actually a second order Volterra series model, as studied by (Pearson *et al.*, 1992). In this configuration, it is necessary to discard terms that have little contribution to the output variables. This issue of discarding unimportant input terms deserves further study.

6. Applications to chemical process modeling

In this section we demonstrate the use of the block recursive PLS algorithm to predict the research octane number for a catalytic reformer. The model output is measured through laboratory test, and it takes a long time to get the laboratory analysis results. We will use the recursive PLS algorithm to estimate the property in real time to eliminate the laboratory test delay. Since it takes several hours to get one laboratory test sample, we are faced with a situation that we may have to build a PLS model based on a modest size data set initially. As more samples are collected over time, the PLS model is updated with newly available data. This is one of the typical situations in which the block recursive PLS can be applied to update the PLS model.

The PLS model uses ten input variables and one output. The input variables are aligned with the laboratory samples based on the time when the laboratory samples were taken. This can be done automatically by matching the time stamps of the laboratory samples and those of the process inputs. Initially a PLS model is built on 66 samples available, which is the first block data. The mean and standard deviation of this block are calculated to normalized the data. Since the mean and variance of the data will change as more data are collected, the meaning of the normalization process is different from that in a typical PLS procedure. Normalization based on the mean is simply to build the model around where the data variation is. A constant input of $1/\sqrt{65}$ in this example is augmented to the PLS model to adapt for the mean changes. The normalization based on the standard deviation is simply to equalize the weights on the variations of each variable.

To build the initial PLS model based on the first data block, a three-fold cross-validation is conducted, which results in seven factors for the PLS model. The actual and predicted outputs are plotted in Fig. 4(a), with the residuals in Fig. 4(b). Fig. 4(c) depicts the PLS regression coefficients for this model, with the eleventh coefficient for the bias input. When the second data block is available, a similar PLS model is build on this block which is shown in Fig. 5. Then the PLS model is updated by combining the two models using the block RPLS algorithm. Fig. 6 gives the PLS model when the sixth data block is available. From Figs. 4(c), 5(c) and 6(c) we notice the changes in the PLS coefficients over time. The updated PLS model after sixth data blocks is shown in Fig. 7(c), and the actual and predicted results for the six blocks of data are shown in Fig. 7(a) and (b).

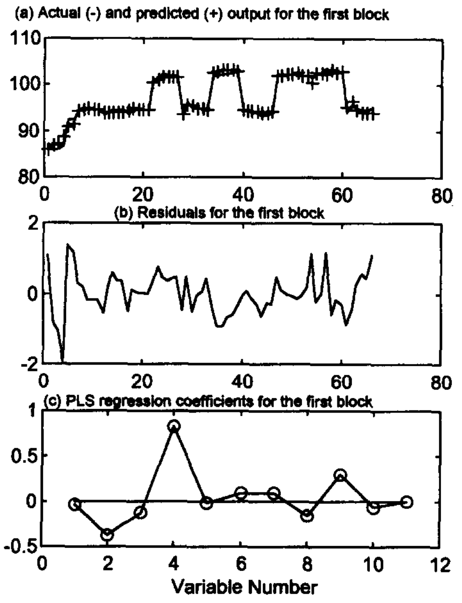

Fig. 4. An initial PLS model from the first data block.

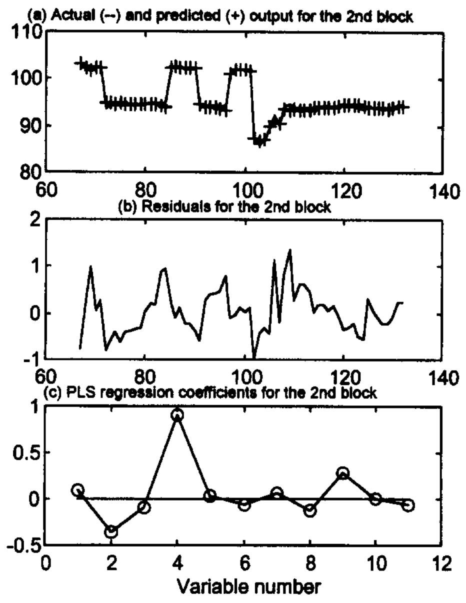

Fig. 5. A PLS model for the second data block.

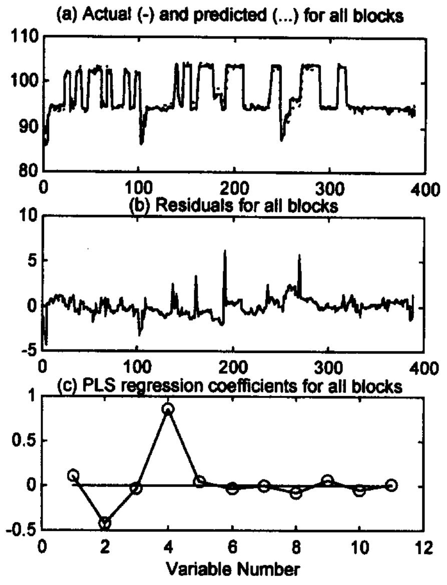

Fig. 7. The block recursive PLS model after six blocks of data have been collected.

With this application we illustrate how the block RPLS algorithm can be used to update a PLS model based on new data.

7. Conclusions

In this paper we present several recursive PLS algorithms for adaptive process modeling based on the initial method of Helland *et al.* (1992). The main results in the paper are:

1. The RPLS algorithm is extended to a block-wise RPLS which builds a PLS sub-model on the new block of data and then combines with the old PLS model.
2. Both a moving window approach and a forgetting factor approach are proposed to update a PLS model.
3. The block-wise RPLS is applied to cross-validation and final PLS modeling in order to reduce computing time and memory usage in the case of large data sets.
4. The use of recursive PLS algorithms in dynamic modeling and nonlinear modeling is discussed.

In deriving the recursive PLS algorithms we modified the PLS algorithm so that the score matrix $\mathbf{T}$ is orthonormal. Some erroneous treatments in the original RPLS algorithm by Helland *et al.* are clarified and mathematical proofs are provided. The number of factors for model updating has to make the input residual $\mathbf{E}$ close to zero, while the number of factors for prediction is determined by cross-validation.

The recursive PLS algorithms can be used on-line or off-line to tackle different problems. In the application example presented in the paper we use the block RPLS algorithm to update the model when more data are available. Other possible applications that can benefit from using the RPLS algorithms are: (1) on-line adaptation of process changes over time, which will use the moving window approach or the forgetting factor approach; and (2) adapting for changes in correlation structure. As process operating conditions may change over time, the correlation structure will change accordingly. For example, some changes can introduce additional degrees of variability, which require additional PLS factors in the model. An interesting issue that deserves further study is how to perform cross-validation on-line to update the number of factors needed for prediction.

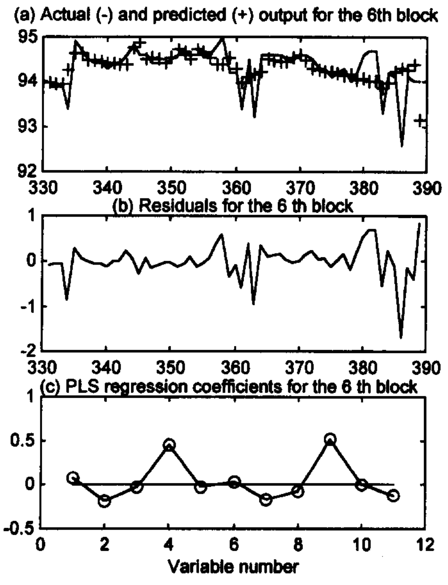

Fig. 6. A PLS model for the sixth data block.

References

Dayal, B. S. and MacGregor, J. F. (1997) Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *J. Process Control* **7**(3), 169–179.

Frank, I. E. (1994) NNPPSS: neural network based on PCR and PLS components nonlinearized by smoothers and splines. Paper presented at *InCINC'94*.

Fuller, M. P., Ritter, G. L. and Draper, C. S. (1988) Partial least-squares quantitative analysis of infrared spectroscopic data. Part I: algorithm implementation; Part II: application to detergent analysis. *Appl. Spectrosc.* **42**, 217–236.

Geladi, P. and Kowalski, B. R. (1986) Partial least-squares regression: a tutorial. *Anal. Chim. Acta* **185**, 1–17.

Haaland, D. M. and Thomas, E. V. (1988) Partial least squares methods for spectral analysis: 1. Relation to other quantitative calibration methods and the extraction of qualitative information. 2. Application to simulated and glass spectral data. *Anal. Chem.* **60**, 1193–1208.

Helland, K., Berntsen, H. E., Borgen, O. S. and Martens, H. (1992) Recursive algorithm for partial least squares regression. *Chemometrics Intell. Lab. Syst.* **14**, 129–137.

Holcomb, T. and Morari, M. (1992) PLS/neural networks. *Computers Chem. Engng* **16**(4), 393–411.

Hŏskuldsson, A. (1988) PLS regression methods. *J. Chemometrics* **2**, 211–228.

Kaspar, M. and Ray, W. H. (1993) Partial least squares modeling as successive singular value decompositions. *Computers Chem. Engng* **17**, 985–989.

Lindberg, W., Persson, J. and Wold, S. (1983) Partial least squares method for spectrofluorimetric analysis of mixtures of humic and ligninsulfonate. *Anal. Chem.* **55**, 643–648.

Ljung, L. (1987) *System Identification: Theory for the User.* Prentice–Hall, Englewood Cliffs, New Jersey.

MacGregor, J. F., Marlin, T. E., Kresta, J. and Skagerberg,. B. (1991) Multivariate statistical methods in process analysis and control. In *Proc. of the Chemical Process Control Conference, CPC-IV,* eds S. Padre Island, Texas, 18–22 Feb.

Martens, H. and Naes, T., Arkun, Y. and Ray, W. H. (1989) *Multivariate Calibration.* Wiley, New York.

Negiz, A. and Cinar, A. (1992) On the detection of multiple sensor abnormalities in multivariate processes In *Proc. of ACC*, Chicago, IL.

Nomikos, P. and MacGregor, J. (1995) Multi-way partial least squares in monitoring batch processes. *Technometrics* **37**, 41–51.

Pearson, R. K., Ogunnaike, B. A. and Doyle, F. J. (1992)Identification of discrete convolution models for nonlinear processes, AIChE Annual Meeting, Paper 125b, Miami, 1–6 Nov.

Piovoso, M. and Owens, A. J. (1991) Sensor data analysis using artificial neural networks. In *Proc. Chemical Process Control (CPC-IV)*. eds. Y. Arkun and W. H. Ray Padre Island, TX, 101-118.

Qin, S. J. (1993) A recursive PLS algorithm for system identification, AIChE Annual Meeting, November 7–12, St. Louis.

Qin, S. J. and McAvoy, T. J. (1992) Nonlinear PLS modeling using neural networks. *Computers Chem. Engng* **16**(4), 379–391.

Qin, S. J. and McAvoy, T. J. (1996) Building nonlinear FIR models via a neural net PLS approach. *Computers Chem. Engng* **20**, 147–159.

Ricker, N. L. (1988) Use of biased least-squares estimators for parameters in descrete-time pulse-response models. *Ind. Engng Chem. Res.* **27**(2), 343–350.

Wise, B. M. and Ricker, N. L. (1990) The effect of biased regression on the identification of FIR and ARX models, AIChE Annual Meeting, Chicago, IL, November.

Wold, H. (1996) Nonlinear estimation by iterative least squares procedures. In *Research Papers in Statistics*, ed. F. David. Wiley, New York.

Wold, S., Ruke, A., Wold, H. and Dunn III, W. J. (1984) The collinearity problem in linear regression, the partial least squares (PLS) approach to generalized inverses. *SIAM J. Sci. Stat. Computs.* **5**(3), 735–743.

Wold, S., Kettaneh-Wold, N. and Skagerberg, B. (1989) Nonlinear PLS modeling. *Chemomet. Intell. Lab. Syst.* **7**, 53–65.

Appendix A

Proof of Lemma 1

Let $[\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_r] \equiv \mathbf{T}_r$. Since $rank(\mathbf{X}) = r = rank(\mathbf{T}_r)$, there exists $\mathbf{P}_* \in \Re^{m \times r}$ such that,

$$\mathbf{X} = \mathbf{T}_r \mathbf{P}_*^T. \tag{49}$$

The matrix $\mathbf{P}_*$ can be found by,

$$\mathbf{P}_*^T = \arg\min_{\mathbf{P}} \|\mathbf{X} - \mathbf{T}_r \mathbf{P}^T\| = (\mathbf{T}_r^T \mathbf{T}_r)^{-1} \mathbf{T}_r^T \mathbf{X} = \mathbf{T}_r^T \mathbf{X} \tag{50}$$

From the PLS algorithm in Table 1,

$$\mathbf{p}_i = \mathbf{X}_i^T \mathbf{t}_i = \left(\mathbf{X} - \sum_{j=1}^{i-1} \mathbf{t}_j \mathbf{p}_j^T \right)^T \mathbf{t}_i = \mathbf{X}^T \mathbf{t}_i \tag{51}$$

which means,

$$[\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_r] = \mathbf{X}^T \mathbf{T}_r = \mathbf{P}_*. \tag{52}$$

From (11),

$$\mathbf{E}_r = \mathbf{X} - \sum_{i=1}^{r} \mathbf{t}_i \mathbf{p}_i^T = \mathbf{X} - \mathbf{T}_r \mathbf{P}_*^T = \mathbf{0} \tag{53}$$

Further residuals after r factors will be zero according to the PLS algorithm in Table 1.

QED.

Proof of Lemma 2

Using the fact that $\mathbf{t}_h$ are mutually orthonormal and (26), we have

$$\mathbf{t}_h^T \mathbf{F}_i = \mathbf{t}_h^T \left(\mathbf{F}_{h-1} - \sum_{j=h}^{i} b_j \mathbf{t}_j \mathbf{q}_j^T \right) = \mathbf{t}_h^T \mathbf{F}_{h-1} - b_h \mathbf{q}_h^T$$

Substituting (10), (8), and (7) in the above equation, we get

$$\begin{aligned} \mathbf{t}_h^T \mathbf{F}_i &= \mathbf{t}_h^T \mathbf{F}_{h-1} - \mathbf{t}_h^T \mathbf{u}_h \mathbf{q}_h^T = \mathbf{t}_h^T \mathbf{F}_{h-1} - \mathbf{t}_h^T \mathbf{F}_{h-1} \mathbf{q}_h \mathbf{q}_h^T \\ &= \mathbf{t}_h^T \mathbf{F}_{h-1} - \mathbf{t}_h^T \mathbf{F}_{h-1} (\mathbf{t}_h^T \mathbf{F}_{h-1})^T \mathbf{t}_h^T \mathbf{F}_{h-1} / \|\mathbf{t}_h^T \mathbf{F}_{h-1}\|^2 \\ &= \mathbf{t}_h^T \mathbf{F}_{h-1} - \mathbf{t}_h^T \mathbf{F}_{h-1} = \mathbf{0} \end{aligned}$$

QED.