A FRAMEWORK FOR PLS-SIM INTEGRATION

Riccardo Muradore [*,**], **Fabrizio Bezzo** [**]

[*] *European Southern Observatory, Garching bei München, Germany*
[**] *Dipartimento di Principi e Impianti di Ingegneria Chimica, University of Padova, Italy*

Abstract: A novel algorithm is presented for the design of inferential estimators for process monitoring and control. The algorithm aims at integrating Partial Least Squares (PLS) techniques and Subspace Identification Methods (SIM) to exploit the main advantages of both methodologies. In particular, the algorithm will retain the PLS computational robustness in dealing with large sets of correlated inputs and outputs, whilst profiting by the SIM dynamic description of the system being investigated.

Keywords: PLS, subspace identification methods

1. INTRODUCTION

Partial Least Squares (PLS) is a very widespread technique to deal with large numbers of (possibly) correlated data ((Geladi and Kowalski, 1986), (Helland, 1988), (Helland, 1990)). However, a potential drawback of PLS consists in its inherently static nature. A potential solution is to include lagged variables in the input matrix in order to take into account past inputs and outputs (Ku *et al.*, 1995), (Lakshminarayanan *et al.*, 1997). A different approach is to merge Subspace Identification Methods (SIM) with Principal Component Analysis (PCA) regression (Negiz and Çinar, 1997), (Wang and Qin, 2002), (Treasure *et al.*, 2004). A comparison between static and dynamic regression can be found in (Shi and MacGregor, 2000). The goal of this paper is to suggest a methodology to combine PLS and SIM. In particular, the iterative algorithm structure of the PLS is preserved, but in the each estimation step SIM is used instead of static regression.

2. PRELIMINARIES

In this section, the PLS regression and SIM are briefly reviewed. The purpose is to highlight and to compare their pros and cons in order to better understand the proposed algorithm.

Let $\{\mathbf{x}(\cdot)\} \in \mathbb{R}^m$ and $\{\mathbf{y}(\cdot)\} \in \mathbb{R}^p$ be two jointly stationary second–order ergodic processes and X, Y two realizations [1]

$$X := [\, x(t_0) \quad x(t_1) \quad \cdots \quad x(t_{N-1}) \,] \in \mathbb{R}^{m \times N},$$
$$Y := [\, y(t_0) \quad y(t_1) \quad \cdots \quad y(t_{N-1}) \,] \in \mathbb{R}^{p \times N}.$$

The goal is to construct a static (using PLS) or a dynamic (using SIM) model mapping future values of $\mathbf{x}(t)$ in an estimation $\hat{\mathbf{y}}(t)$ of $\mathbf{y}(t)$. Observe that $\mathbf{x}$ and $\mathbf{y}$ do not necessarily have the meaning of input and output of the plant. In fact, they can be thought to be two different sets of measurements, i.e. process variables and quality variables. To obtain good product and/or to keep the system

[1] From now on, bold letters are used for random vectors whereas italic letters for the sample values.

under control, it is necessary to estimate the set of quality variables (having usually a small sample time) using the process variables (having a high sample time).

For sake of readability, the random vectors $\mathbf{x}(\cdot)$ and $\mathbf{y}(\cdot)$ are assumed to be scaled to zero-mean and unit variance.

2.1 A review of PLS: NIPALS algorithm

The nonlinear iterative partial least squares (NIPALS) algorithm (Geladi and Kowalski, 1986) is written here with the rows and columns reversed with respect to the usual notation and without normalization. The goal is to give for each equation a geometric interpretation that will be useful in the following. Here, the Greek letter Σ is used to indicate variance and covariance matrices. Moreover, $\mathbf{E}[\cdot]$ and $\mathbf{E}[\cdot|\cdot]$ are the expectation and the conditional expectation, respectively. The slightly different symbol $\hat{\mathbf{E}}[\cdot|\cdot]$ is introduced for the linear projection operator (in the Gaussian case, $\hat{\mathbf{E}}[\cdot|\cdot] = \mathbf{E}[\cdot|\cdot]$).

After setting $Y_1 = Y$, $X_1 = X$, the following iterations are repeated for each loading, $k = 1, 2, \ldots$. Observe that the subscript in each variable means the loading number, whereas the apex the number of iteration.

(1) Initialization: set s_k^0 be equal to a row of Y_k.
(2) Linear regression step: find the loading vector $\hat{w}_k^i = \hat{\Sigma}_{X_k s_k^i} \hat{\Sigma}_{s_k^i}^{-1}$ that regresses the rows of X_k on s_k^i:

$$X_k = \hat{w}_k^i s_k^i + E_k^i,\ \hat{w}_k^i = \frac{X_k (s_k^i)^T}{s_k^i (s_k^i)^T}. \quad (1)$$

The matrix E_k^i is the estimation error.
(3) Estimation step: compute the score t_k^i as the LSE of s_k^i based on X_k using the model (1)

$$t_k^i := \hat{s}_k^i = \left((\hat{w}_k^i)^T \hat{w}_k^i\right)^{-1} (\hat{w}_k^i)^T X_k.$$

(4) Linear regression step: find the loading vector $q_k^i = \Sigma_{t_k^i}^{-1} \Sigma_{t_k^i Y_k}$ that regresses the rows of Y_k on t_k^i

$$Y_k = \hat{q}_k^i t_k^i + W_k^i,\ \hat{q}_k^i = \frac{Y_k^i (t_k^i)^T}{t_k^i (t_k^i)^T}. \quad (2)$$

The matrix W_k^i is the estimation error.
(5) Estimation step: compute the score s_k^{i+1} as the LSE of t_k^i based on Y_k using the model (2)

$$s_k^{i+1} := \hat{t}_k^i = \left((\hat{q}_k^i)^T \hat{q}_k^i\right)^{-1} (\hat{q}_k^i)^T \mathbf{y}_k.$$

(6) Check convergence: compare two subsequent realizations of s_k (s_k^i, s_k^{i+1}). If $\|s_k^{i+1} - s_k^i\| > \varepsilon$, then set $i := i + 1$ and go back to step 2; else, $t_k := t_k^i$ and go to step 7.
(7) Linear regression step: find the loading vector $p_k = \Sigma_{t_k}^{-1} \Sigma_{t_k X_k}$ regressing the rows of X_k on t_k

$$X_k = \hat{p}_k t_k + V_k,\ \hat{p}_k = \frac{X_k (t_k)^T}{t_k (t_k)^T}.$$

(8) Matrix updating: compute the residual matrices X_{k+1} and Y_{k+1}

$$\begin{aligned} X_{k+1} &= X_k - \hat{\mathbf{E}}[X_k|t_k] = X_k - \hat{q}_k t_k \\ Y_{k+1} &= Y_k - \hat{\mathbf{E}}[Y_k|t_k] = Y_k - \hat{p}_k t_k. \end{aligned}$$

They comprise the information of the original matrices X and Y not explained by the first k loadings.
(9) Check the stopping rule: it is possible to compute loadings until the maximum from p and m is achieved. Otherwise, the procedure can be stopped when the explained variance on the Y–block is big enough or when the increment between two steps is not significative.

At the end of the procedure, the original matrices are written as

$$\begin{aligned} X &= \textstyle\sum_{k=1}^N \hat{q}_k t_k + E = QT + E \\ Y &= \textstyle\sum_{k=1}^N \hat{p}_k t_k + F = PT + F \end{aligned}$$

where N is the number of loadings and E and F are the error matrices. These equations have good numerical properties even if the original data are collinear. From the above matrices, the estimated linear static model becomes

$$\mathbf{y}(t) = A\mathbf{x}(t).$$

2.2 A review of SIM

A innovation state–space model of $\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$ is given by [2]

$$\mathbf{z}(t) = \left[\begin{array}{c|c} A & K \\ \hline C & I \end{array}\right] \mathbf{e}(t) \quad (4)$$

where the innovation process $\mathbf{e}$ has variance equal to Π. The unknowns are the matrices A, K, C, Π, where K is the steady–state Kalman gain.

[2] Here and in the following, the compact notation $y = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] u$ is used instead of the state–space model description

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t). \end{cases} \quad (3)$$

Subspace methods for time–series can be used to estimate all these matrices, (Van Overschee and De Moor, 1993), (Lindquist and Picci, 1996).

As before an estimator of $\mathbf{y}(t)$ based on $\mathbf{x}(t)$ needs to be designed. The model (4) can be written as

$$\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\end{bmatrix} = \left[\begin{array}{c|cc} A & K_1 & K_2 \\ \hline L & I & 0 \\ C & 0 & I \end{array}\right] \begin{bmatrix}\mathbf{e}_1(t)\\ \mathbf{e}_2(t)\end{bmatrix}, \quad (5)$$

$$\mathsf{Var}\begin{bmatrix}\mathbf{e}_1(t)\\ \mathbf{e}_2(t)\end{bmatrix} = \begin{bmatrix}\Pi_{11} & \Pi_{12}\\ \Pi_{21} & \Pi_{22}\end{bmatrix}.$$

In order to find the dynamic system mapping $\mathbf{x}(t)$ into $\mathbf{y}(t)$, a transformation matrix to obtain uncorrelated noises $\mathbf{e}_\mathbf{x}$, $\mathbf{e}_\mathbf{y}$ is first performed

$$\begin{bmatrix}\mathbf{e}_\mathbf{x}(t)\\ \mathbf{e}_\mathbf{y}(t)\end{bmatrix} = \begin{bmatrix} I & 0\\ -\Pi_{21}\Pi_{11}^{-1} & I\end{bmatrix}\begin{bmatrix}\mathbf{e}_1(t)\\ \mathbf{e}_2(t)\end{bmatrix}.$$

After setting $J = \Pi_{21}\Pi_{11}^{-1}$, the system (5) becomes

$$\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{y}(t)\end{bmatrix} = \left[\begin{array}{c|cc} A & K_\mathbf{x} & K_\mathbf{y} \\ \hline L & I & 0 \\ C & J & I \end{array}\right] \begin{bmatrix}\mathbf{e}_\mathbf{x}(t)\\ \mathbf{e}_\mathbf{y}(t)\end{bmatrix} \quad (6)$$

$$\mathsf{Var}\begin{bmatrix}\mathbf{e}_\mathbf{x}(t)\\ \mathbf{e}_\mathbf{y}(t)\end{bmatrix} = \begin{bmatrix}\Pi_\mathbf{x} & 0\\ 0 & \Pi_\mathbf{y}\end{bmatrix}, \quad (7)$$

where $K_\mathbf{x} := K_1$ and $K_\mathbf{y} := K_2 + K_1 J$. Substituting the $\mathbf{x}$–measurement equation into the state equation and into the $\mathbf{y}$–measurement equation (quality measurement equation), a system mapping $\mathbf{x}(t)$ and $\mathbf{e}_\mathbf{y}(t)$ into $\mathbf{y}(t)$ is obtained. To get the estimator $\hat{\mathbf{y}}(t)$ for $\mathbf{y}(t)$ based on past measurements $\mathbf{x}(t_0)$,...,$\mathbf{x}(t-1)$, it is necessary to

(1) compute the state estimation $\hat{\mathbf{x}}(t)$ based on $\mathbf{x}(t_0)$, ..., $\mathbf{x}(t-1)$ using a (one–step–ahead or *a priori*) steady–state Kalman filter,
(2) substitute the state estimation $\hat{\mathbf{x}}(t)$ in the quality measurement equation.

At the end, the estimator takes the form

$$\hat{\mathbf{y}}(t) = \left[\begin{array}{c|c} A - GL & G \\ \hline C - JL & J \end{array}\right]\mathbf{x}(t) \quad (8)$$

where G is the Kalman gain

$$G = [A - K_\mathbf{y} L]\, PL^T \left[LPL^T + \Pi_\mathbf{x}\right]^{-1} + K_\mathbf{y}$$

and P is the positive–definite solution of the algebraic Riccati equation

$$P = (A - K_\mathbf{y}L)\left(P - PL\left(LPL^T + \Pi_\mathbf{x}\right)^{-1} L^T P\right)\cdot \\ \cdot (A - K_\mathbf{y}L)^T + K_\mathbf{y}\Pi_\mathbf{y}K_\mathbf{y}^T.$$

It is important to highlight that numerical robustness of SIMs is strongly dependent on the data. If there are collinear components in the matrix Z (as it often happens in industrial data), the computation may be ill-posed. This means that even if the estimator (8) is correct from a theoretical point of view, it could still give wrong results because of numerical issues.

To take advantage of the above methods, an estimator like (8) is derived by following the PLS–philosophy. The idea is to replace linear regressions with subspace identifications in the steps 2, 4, 7 in the PLS iterative algorithm.

3. DYNAMIC LINEAR PLS ALGORITHM

The algorithm is partitioned as the static PLS algorithm in Section 2.1, and it is explained by the flowchart in Figure 1.

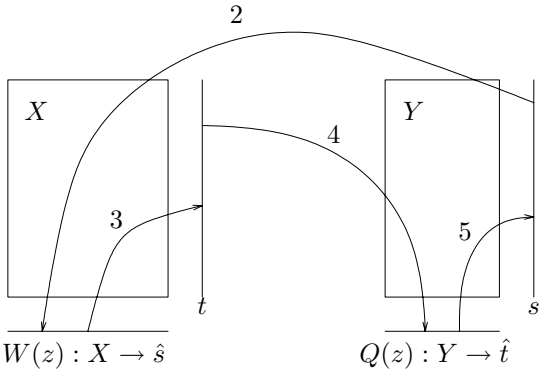

Fig. 1. Flowchart of the Dynamic PLS

With respect to the standard PLS, there are two important differences:

a) static linear regressions are now substituted by system identifications.
b) instead of (dynamically) regress X_k on s_k^i in step 2 and Y_k on t_k^i in step 4, the algorithm regresses s_k^i on X_k and t_k^i on Y_k, respectively. This avoids the inversion of transfer functions.

Thus, the algorithm is structured as follows:

(1) Initialization: set $\mathbf{s}_k^0$ equal to a variable of the vector $\mathbf{y}_k$ and let s_k^0 be the corresponding trajectory.
(2) SIM step: identify the state–space system mapping $\mathbf{x}_k$ into $\mathbf{s}_k^i$. The first step is to find the stochastic system describing the enhanced output vector $\begin{bmatrix}\mathbf{x}_k\\ \mathbf{s}_k^i\end{bmatrix}$ using the data $\begin{bmatrix}X_k\\ s_k^i\end{bmatrix}$

$$\begin{bmatrix}\mathbf{x}_k(t)\\ \mathbf{s}_k^i(t)\end{bmatrix} = \left[\begin{array}{c|cc} A_k^i & K_{1,k}^i & K_{2,k}^i \\ \hline L_k^i & I & 0 \\ c_k^i & 0 & 1 \end{array}\right] \begin{bmatrix}\mathbf{e}_{1,k}^i(t)\\ \mathbf{e}_{2,k}^i(t)\end{bmatrix}$$

$$\mathsf{Var}\begin{bmatrix}\mathbf{e}_{1,k}^i\\ \mathbf{e}_{2,k}^i\end{bmatrix} = \Pi_k^i = \begin{bmatrix}\Pi_{11}^i & \Pi_{12}^i\\ \Pi_{21}^i & \Pi_{22}^i\end{bmatrix}.$$

The quality measurement $\mathbf{s}_k^i(t)$ is now a scalar fictitious variable. In order to find the system mapping $\mathbf{x}_k$ into $\mathbf{s}_k^i$, one needs to do the same manipulations described in Section 2.2. All matrices have a subscript related to the loading number (k) and an apex related to the iteration number (i). Since the meaning of the matrices is the same, the mathematics is not reported. The model is given by

$$\mathbf{s}_k^i(t) = \underbrace{W_k^i(z)\mathbf{x}_k(t)}_{\hat{\mathbf{s}}_k^i(t)} + \mathbf{n}_k^i(t)$$

$$W_k^i(z) = \left[\begin{array}{c|c} A_k^i - G_k^i L_k^i & G_k^i \\ \hline c_k^i - J_k^i L_k^i & J_k^i \end{array}\right] \quad (9)$$

where the model noise $\mathbf{n}_k^i(t)$ is orthogonal to $\hat{\mathbf{s}}_k^i(t)$.

(3) Estimation step: compute the scores t_k^i. By redefining the variable $\hat{\mathbf{s}}_k^i$ as the score variable $\mathbf{t}_k^i$, it is possible to obtain a realization $t_k^i(t)$, $t = t_0, \ldots, t_{N-1}$ as the output of the system (9) driven by $X_k(t)$, $t = t_0, \ldots, t_{N-1}$. The k–score data matrix at the i–step is defined as

$$t_k^i := [\, t_k^i(t_0) \quad t_k^i(t_1) \quad \cdots \quad t_k^i(t_{N-1}) \,] \in \mathbb{R}^{1\times N}.$$

(4) SIM step: identify the state–space system mapping $\mathbf{y}_k$ into $\mathbf{t}_k^i$. The first step is to find the stochastic system describing the enhanced output vector $\begin{bmatrix} \mathbf{y}_k \\ \mathbf{t}_k^i \end{bmatrix}$ using the data $\begin{bmatrix} Y_k \\ t_k^i \end{bmatrix}$:

$$\begin{bmatrix} \mathbf{y}_k(t) \\ \mathbf{t}_k^i(t) \end{bmatrix} = \left[\begin{array}{c|cc} F_k^i & G_{1,k}^i & G_{2,k}^i \\ \hline M_k^i & I & 0 \\ h_k^i & 0 & 1 \end{array}\right] \begin{bmatrix} \mathbf{w}_{1,k}^i(t) \\ \mathbf{w}_{2,k}^i(t) \end{bmatrix}$$

$$\mathsf{Var}\begin{bmatrix} \mathbf{w}_{1,k}^i(t) \\ \mathbf{w}_{2,k}^i(t) \end{bmatrix} = \begin{bmatrix} \Lambda_{11,k}^i & \Lambda_{12,k}^i \\ \Lambda_{21,k}^i & \Lambda_{22,k}^i \end{bmatrix}.$$

The model takes the form

$$\mathbf{t}_k^i(t) = \underbrace{Q_k^i(z)\mathbf{y}_k(t)}_{\hat{\mathbf{t}}_k^i(t)} + \mathbf{v}_k^i(t)$$

$$Q_k^i(z) = \left[\begin{array}{c|c} F_k^i - G_k^i M_k^i & G_k^i \\ \hline h_k^i - V_k^i M_k^i & V_k^i \end{array}\right] \quad (10)$$

where the model noise $\mathbf{v}_k^i(t)$ is orthogonal to $\hat{\mathbf{t}}_k^i(t)$.

(5) Estimation step: compute the scores s_{k+1}^i. By redefining the variable $\hat{\mathbf{t}}_k^i(t)$ as the score variable $\mathbf{s}_{k+1}^i$, it is possible to obtain a new realization $s_k^{i+1}(t)$, $t = t_0, \ldots, t_{N-1}$ as the output of the system (10) driven by $Y_k(t)$, $t = t_0, \ldots, t_{N-1}$. The s_k^{i+1} score matrix is

$$s_k^{i+1} := [\, s_k^{i+1}(t_0) \quad s_k^{i+1}(t_1) \quad \cdots \quad s_k^{i+1}(t_{N-1}) \,].$$

(6) Check convergence: compare two subsequent realizations of $\mathbf{s}_k$. If $\|s_k^{i+1} - s_k^i\| > \varepsilon$ then $i := i + 1$ and go back to step 2; else, $t_k = t_k^i$ (i.e. $\mathbf{t}_k = \mathbf{t}_k^i$) and go to step 7.

(7) SIM step: identify the state–space systems mapping $\mathbf{t}_k$ into $\hat{\mathbf{x}}_k$ and of $\hat{\mathbf{y}}_k$

$$\hat{\mathbf{x}}_k(t) = \left[\begin{array}{c|c} A_{\mathbf{x},k} & B_{\mathbf{x},k} \\ \hline C_{\mathbf{x},k} & D_{\mathbf{x},k} \end{array}\right] \mathbf{t}_k(t) \quad (11)$$

$$\hat{\mathbf{y}}_k(t) = \left[\begin{array}{c|c} A_{\mathbf{y},k} & B_{\mathbf{y},k} \\ \hline C_{\mathbf{y},k} & D_{\mathbf{y},k} \end{array}\right] \mathbf{t}_k(t). \quad (12)$$

Moreover, since it is necessary to move from the real measurement $X_k(t)$ to the fictitious variable $t_k(t)$, the system mapping $\mathbf{x}_k$ into $\mathbf{t}_k$

$$\mathbf{t}_k(t) = \left[\begin{array}{c|c} A_{\mathbf{t},k} & B_{\mathbf{t},k} \\ \hline C_{\mathbf{t},k} & D_{\mathbf{t},k} \end{array}\right] \mathbf{x}_k(t) = \left[\begin{array}{c|c} A_k^i - G_k^i L_k^i & G_k^i \\ \hline c_k^i - J_k^i L_k^i & J_k^i \end{array}\right] \mathbf{x}_k(t) \quad (13)$$

is also computed.

(8) Matrix updating: compute the residual matrices X_{k+1} and Y_{k+1} by subtracting the estimations obtained using (11) and (12), respectively,

$$X_{k+1} := X_k - \left[\begin{array}{c|c} A_{\mathbf{x},k} & B_{\mathbf{x},k} \\ \hline C_{\mathbf{x},k} & D_{\mathbf{x},k} \end{array}\right] t_k \quad (14)$$

$$Y_{k+1} := Y_k - \left[\begin{array}{c|c} A_{\mathbf{y},k} & B_{\mathbf{y},k} \\ \hline C_{\mathbf{y},k} & D_{\mathbf{y},k} \end{array}\right] t_k. \quad (15)$$

(9) Check the stopping rule. The procedure is stopped when the explained variance on the Y–block is big enough or when the increment between two steps is not significative.

Let N be the number of dynamic loadings computed. The plant is written as

$$\mathbf{x}(t) = \sum_{k=1}^{N} \left[\begin{array}{c|c} A_{\mathbf{x},k} & B_{\mathbf{x},k} \\ \hline C_{\mathbf{x},k} & D_{\mathbf{x},k} \end{array}\right] \mathbf{t}_k(t) + \mathbf{e}(t) \quad (16)$$

$$\mathbf{y}(t) = \sum_{k=1}^{N} \left[\begin{array}{c|c} A_{\mathbf{y},k} & B_{\mathbf{y},k} \\ \hline C_{\mathbf{y},k} & D_{\mathbf{y},k} \end{array}\right] \mathbf{t}_k(t) + \mathbf{f}(t) \quad (17)$$

where $\mathbf{e}(t)$ and $\mathbf{f}(t)$ are the approximation errors. Note that to compute the estimation $\hat{\mathbf{y}}(t)$ of $\mathbf{y}(t)$ based on the measurements $\mathbf{x}(t_0)$, $\mathbf{x}(t_1)$, …, $\mathbf{x}(t-1)$, the transfer functions (13) from $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, …, $\mathbf{x}_N(t)$ to $\mathbf{t}_1(t)$, $\mathbf{t}_2(t)$, …, $\mathbf{t}_N(t)$ are also needed (see the equations at the top of the next page).

When the scores are available, it is possible to compute the estimation according to formula:

$$\hat{\mathbf{y}}(t) = \sum_{k=1}^{N} \left[\begin{array}{c|c} A_{\mathbf{y},k} & B_{\mathbf{y},k} \\ \hline C_{\mathbf{y},k} & D_{\mathbf{y},k} \end{array}\right] \mathbf{t}_k(t) = \sum_{k=1}^{N} G_{\mathbf{t}_k\mathbf{y}}(z)\mathbf{t}_k(t).$$

$$
\begin{aligned}
\mathbf{t}_1(t) &= \left[\begin{array}{c|c} A_{\mathbf{t},1} & B_{\mathbf{t},1} \\ \hline C_{\mathbf{t},1} & D_{\mathbf{t},1} \end{array}\right] \mathbf{x}_1(t) = \left[\begin{array}{c|c} A_{\mathbf{t},1} & B_{\mathbf{t},1} \\ \hline C_{\mathbf{t},1} & D_{\mathbf{t},1} \end{array}\right] \mathbf{x}(t) && =: G_{\mathbf{xt}_1}(z)\mathbf{x}(t) \\
\mathbf{t}_2(t) &= \left[\begin{array}{c|c} A_{\mathbf{t},2} & B_{\mathbf{t},2} \\ \hline C_{\mathbf{t},2} & D_{\mathbf{t},2} \end{array}\right] \mathbf{x}_2(t) = \left[\begin{array}{c|c} A_{\mathbf{t},2} & B_{\mathbf{t},2} \\ \hline C_{\mathbf{t},2} & D_{\mathbf{t},2} \end{array}\right] \left\{\mathbf{x}(t) - \left[\begin{array}{c|c} A_{\mathbf{x},1} & B_{\mathbf{x},1} \\ \hline C_{\mathbf{x},1} & D_{\mathbf{x},1} \end{array}\right] \mathbf{t}_1(t)\right\} && =: G_{\mathbf{xt}_2}(z)\mathbf{x}(t) \\
&\vdots \\
\mathbf{t}_n(t) &= \left[\begin{array}{c|c} A_{\mathbf{t},n} & B_{\mathbf{t},n} \\ \hline C_{\mathbf{t},i} & D_{\mathbf{t},n} \end{array}\right] \mathbf{x}_n(t) = \left[\begin{array}{c|c} A_{\mathbf{t},n} & B_{\mathbf{t},n} \\ \hline C_{\mathbf{t},n} & D_{\mathbf{t},n} \end{array}\right] \left\{\mathbf{x}(t) - \textstyle\sum_{k=1}^{n-1} \left[\begin{array}{c|c} A_{\mathbf{x},k} & B_{\mathbf{x},k} \\ \hline C_{\mathbf{x},k} & D_{\mathbf{x},k} \end{array}\right] \mathbf{t}_k\right\} && =: G_{\mathbf{xt}_n}(z)\mathbf{x}(t)
\end{aligned}
$$

Remark 1. Note that the updating of X_k in equation (14) is not necessary because the only necessary updating is the one related to Y_k (this fact also holds in the standard PLS). If the input matrix is not updated, the new scores $\bar{\mathbf{t}}_i$ (which differ from $\mathbf{t}_i$ in equation (13)) can be calculated by

$$
\bar{\mathbf{t}}_i(t) = \left[\begin{array}{c|c} \bar{A}_{\mathbf{t},i} & \bar{B}_{\mathbf{t},i} \\ \hline C_{\mathbf{t},i} & D_{\mathbf{t},i} \end{array}\right] \mathbf{x}(t)
$$

without computing $\mathbf{x}_2(t)$, $\mathbf{x}_3(t)$, ..., $\mathbf{x}_n(t)$. Then it is possible to resort to the more compact block diagram of figure 3 instead of using the one obtained following the previous construction, see figure 2.

4. AN IMPROVEMENT: DYNAMIC PLS WITH INPUTS

Until now, we have used as input and output data only process and quality variables. It is also possible to introduce the controlled variables (if they are available) in the model by resorting to the *subspace identification with inputs* algorithms, (Van Overschee and De Moor, 1994). The above algorithm changes only in step 2 where the identified plant becomes

$$
\left[\begin{array}{c} \mathbf{x}_k(t) \\ \mathbf{s}_k^i(t) \end{array}\right] = \left[\begin{array}{c|cc:c} A_k^i & B_k^i & K_{1,k}^i & K_{2,k}^i \\ \hline L_k^i & 0 & I & 0 \\ c_k^i & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} \mathbf{u}(t) \\ \mathbf{e}_{1,k}^i(t) \\ \mathbf{e}_{2,k}^i(t) \end{array}\right]
$$

$$
\mathsf{Var} \left[\begin{array}{c} \mathbf{e}_{1,k}^i \\ \mathbf{e}_{2,k}^i \end{array}\right] = \Pi_k^i = \left[\begin{array}{cc} \Pi_{11}^i & \Pi_{12}^i \\ \Pi_{21}^i & \Pi_{22}^i \end{array}\right].
$$

Therefore the dynamic PLS takes the form

$$
\begin{aligned}
\mathbf{t}_k(t) &= \left[\begin{array}{c|cc:c} A_{\mathbf{t},k} & B_{\mathbf{t},k}^{\mathbf{u}} & B_{\mathbf{t},k}^{\mathbf{x}} \\ \hline C_{\mathbf{t},k} & 0 & D_{\mathbf{t},k} \end{array}\right] \left[\begin{array}{c} \mathbf{u}(t) \\ \mathbf{x}_k(t) \end{array}\right] \\
\hat{\mathbf{y}}(t) &= \sum_{k=1}^{n} \left[\begin{array}{c|c} A_{\mathbf{y},k} & B_{\mathbf{y},k} \\ \hline C_{\mathbf{y},k} & D_{\mathbf{y},k} \end{array}\right] \mathbf{t}_k(t)
\end{aligned}
$$

where the controlled inputs and the process variables are first projected into the score space and then are used to estimate the quality variables.

5. CONCLUSION

The paper has presented and discussed a novel algorithm to integrate PLS and SIM methodologies. The main benefit is that a truly dynamic description of the system being considered is feasible without losing the numerical advantages of a PLS approach. Even if some work needs to be done to define the theoretical foundation for the proposed algorithm and demonstrate its convergence, the first simulations have produced promising results. The method is effective for the same reasons as the PLS algorithm. Recursively, it constructs subspaces that become after each iteration closer to each other (s_k^i). The only difference is that the computation also takes into account the process dynamics. In other words, the data fitting obtained by identification is more accurate that the static data fitting obtained by using the least-squares method. Future work will include the application of the suggested algorithm to process data to assess its effectiveness when compared to standard PLS.

REFERENCES

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

Helland, I.S. (1988). On the structure of partial least squares regression. *Commun. Statist. B–Simulation Comput.* **17**(2), 581–607.

Helland, I.S. (1990). Partial least squares regression and statistical models. *Scand. J. Statist.* **17**, 97–114.

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

Lakshminarayanan, S., S.L. Shah and K. Nadakumar (1997). Modeling and control of multivariable processes: dynamic pls approach. *AIChE J.* **43**(9), 2307–2322.

Lindquist, A. and G. Picci (1996). Canonical correlation analysis, approximate covariance extension, and identification of stationary time series. *Automatica* **32**(5), 709–733.

Negiz, A. and A. Çinar (1997). Statistical monitoring of multivariable dynamic pro-

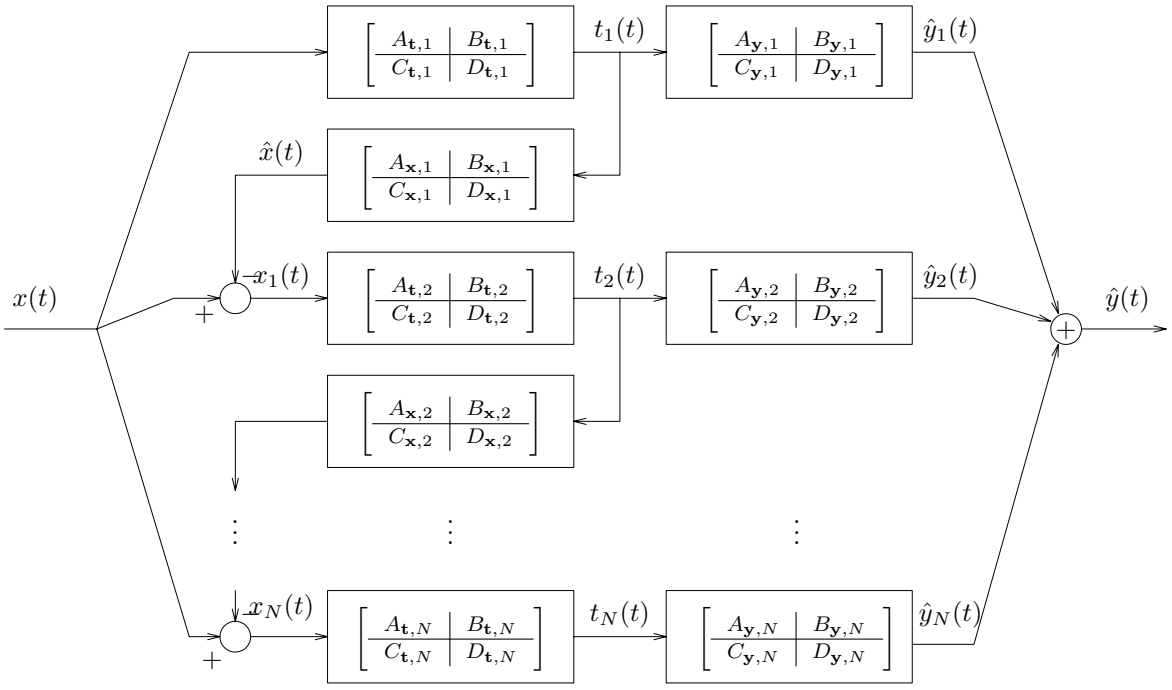

Fig. 2. Block diagram of the *basic* Dynamic PLS

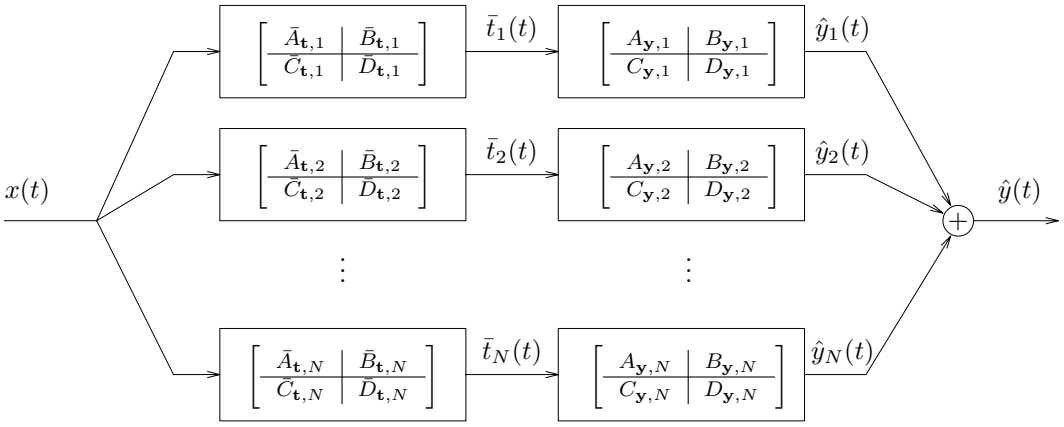

Fig. 3. Block diagram of the *compact* Dynamic PLS

cesses with state-space models. *AIChE J.* **43**(8), 2002–2020.

Shi, R. and J.F. MacGregor (2000). Modeling of dynamic systems using latent variable and subspace methods. *J. of Chemometrics* **14**, 423–439.

Treasure, R.J., U. Kruger and J.E. Cooper (2004). Dynamic multivariate statistical process control using subspace identification. *J. Process Control* **14**, 279–292.

Van Overschee, P. and B. De Moor (1993). Subspace algorithms for the stochastic identification problem. *Automatica* **29**(3), 649–660.

Van Overschee, P. and B. De Moor (1994). N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica* **30**(1), 75–93.

Wang, J. and S.J. Qin (2002). A new subspace identification approach based on principal component analysis. *Journal of Process Control* **12**, 841–855.