Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 54(2), 166–177(2013)

Dražen Slišković, Ratko Grbić, Željko Hocenski

Adaptive Estimation of Difficult-to-Measure Process Variables

DOI 10.7305/automatika.54-2.147
UDK 681.518.2.08:658.5
IFAC 3.2; 1.1

Original scientific paper

There exist many problems regarding process control in the process industry since some of the important variables cannot be measured online. This problem can be significantly solved by estimating these difficult-to-measure process variables. In doing so, the estimator is in fact an appropriate mathematical model of the process which, based on information about easy-to-measure process variables, estimates the current value of the difficult-to-measure variable. Since processes are usually time-varying, the precision of the estimation based on the process model which is built on old data is decreasing over time. To avoid estimator accuracy degradation, model parameters should be continuously updated in order to track process behavior. There are a couple of methods available for updating model parameters depending on the type of process model. In this paper, PLSR process model is chosen as the basis of the difficult-to-measure process variable estimator while its parameters are updated in several ways – by the moving window method, recursive NIPALS algorithm, recursive kernel algorithm and Just-in-Time learning algorithm. Properties of these adaptive methods are explored on a simulated example. Additionally, the methods are analyzed in terms of computational load and memory requirements.

Key words: Process variable estimation, Adaptive estimator, Moving window, Recursive algorithms, JITL algorithm

Adaptivna estimacija teško-mjerljivih procesnih veličina. Problemi s upravljanjem mnogih procesa u industriji vezani su s nemogućnošću on-line mjerenja nekih važnih procesnih veličina. Ovaj se problem može u značajnoj mjeri riješiti estimacijom ovih teško-mjerljivih procesnih veličina. Estimator je pri tome odgovarajući matematički model procesa koji na temelju informacije o ostalim (lako-mjerljivim) procesnim veličinama procjenjuje trenutni iznos teško-mjerljive veličine. Budući da su procesi po prirodi promjenjivi, točnost estimacije zasnovane na modelu procesa izgrađenog na starim podacima u pravilu opada s vremenom. Kako bi se ovo izbjeglo, parametre modela procesa je potrebno kontinuirano prepodešavati kako bi model što bolje opisivao (trenutno) vladanje procesa. Ovisno o tipu matematičkog modela, za prepodešavanje njegovih parametara na raspolaganju je više metoda. Kao osnova estimatora teško-mjerljive veličine u radu se koristi PLSR model procesa, dok se njegovi parametri prepodešavaju na više načina – metodom pomičnog prozora, rekurzivnim NIPALS algoritmom, rekurzivnim kernel algoritmom te Just-in-Time Learning metodom. Svojstva navedenih metoda adaptacije PLSR modela procesa ispitana su na odabranom primjeru. Nadalje, metode adaptacije su analizirane i s obzirom na računalnu i memorijsku zahtjevnost.

Ključne riječi: estimacija procesne veličine, adaptivni estimator, pomični prozor, rekurzivni algoritmi, JITL algoritam

1 INTRODUCTION

Some of the problems regarding efficient process control come from the fact that some of the important process variables cannot be measured online (so-called difficult-to-measure process variables – DTM variables) [1]. If a process variable cannot be measured, its estimation is an alternative procedure. The estimation is based on other process variables that are measured by sensors in the plant (so-called easy-to-measure variables - ETM variables) which are in correlation with the DTM variable and on a mathematical model of the process [1–3].

The estimator is in fact an appropriate mathematical model of the process. In practice, the model is usually not available. The most economic way to build a process model is to use measured data which can be taken from the plant database. Various statistical methods and/or computational learning methods can be used in model developing [3–5].

Process model building based on plant data is a challenging task because plant data are usually of low infor-

mativity. The measured data contain a great number of samples which are "information poor" due to the presence of different errors such as measurement noise, missing values and outliers [6]. Apart from that, all possible working conditions of the process are generally not available (e.g. not recorded in the database). Additionally, the relationship between variables can change over time due to different external influences or process changes so the data from the process database are not relevant anymore (so-called historical data). Therefore, only the most recent data represent the true state of the process and should be used for offline modeling procedure [3].

Since the measured plant data are usually of low quality, data preprocessing is of a great importance in process model building [6, 7]. The estimation of DTM variable is usually performed on a great number of ETM variables which results in high dimensional and highly correlated input space for the modeling procedure. Because of that, models with two levels of projection are generally used [8].

After the model has been developed, the estimator of the DTM variable can be implemented in the plant Supervisory Control and Data Acquisition (SCADA) system. Such a computer program is usually called a soft-sensor since it substitutes some physical sensor/procedure by a software routine. In this paper terms soft-sensor and DTM variable estimator are interchanged, although soft-sensor can also be a back-up sensor for a hardware sensor. Samples of the ETM variables are continuously fed into the estimator during online operation of the estimator. Since it is possible that samples contain different errors, samples must be checked online and preprocessed before estimation procedure is performed in order to achieve an accurate estimation of the DTM variable [9, 10].

As already mentioned, industrial processes exhibit some kind of time-varying behavior and it is necessary to ensure that the soft-sensor retains its estimation precision during online operation. To avoid degradation of estimator precision, adaptive models (adaptive soft sensors) are used. The process model that is built offline serves only as an initial process model whose parameters are updated during process operation (online) in order to track process time-varying behavior. In that way, the model can describe current process input-output relationship constantly during the long time period of soft-sensor exploitation [9]. Therefore, this paper deals with the properties of different adaptive model building approaches which are especially important for plant soft-sensor implementation.

The paper is organized as follows. Section 2 describes the basics of process modeling and process model adaptation. Methods for the model adaptation of the PLSR model are described in Section 3 in more details. These adaptive methods are applied to a chosen example in Section 4, followed by the numerical results and accompanying discussion. Section 5 provides a summary and conclusion of this paper.

2 ADAPTIVE ESTIMATOR

Initial methods that were used for DTM variable estimator development can be found in the field of multivariate statistical analysis. Rapid expansion of artificial neural networks and other intelligent methods led to the expansion of different hybrid methods [3–5]. Estimators based on these (static) models usually have fixed parameters. This property causes slow degradation of soft-sensor performance during a longer period of time. Apart from temporal process changes, a reason for estimator imprecision can be low informativity of the (initial) data set that is used in offline process model building. The initial data set can contain a lot of different errors and disturbances or it does not contain all possible working regimes of the process that can appear later in soft-sensor exploitation.

These issues can be (partially) solved with the use of the adaptive estimator in which parameters are updated (recalculated) during online estimator operation. This was recognized by different authors [11–16] but there are still some unaddressed issues in soft-sensor development and maintenance [3].

2.1 Model Structuring

Data based modeling is in fact a search for the function $\boldsymbol{f}_m(\cdot)$ (i.e. process model) that approximates the unknown natural functional dependence between process input-output variables. This procedure involves different steps (see [1, 3, 4]) and it is often performed in several iterations.

An important step of the modeling procedure is model structuring. Optimal model structure is usually unknown due to the lack of process knowledge. Therefore, the general model structure is used [8]:

$$\hat{y} = \boldsymbol{f}_m(\mathbf{x}, \boldsymbol{\Theta}) = \sum_{k=1}^{K} \boldsymbol{\nu}_k(\mathbf{x}, \boldsymbol{\Theta}), \tag{1}$$

where $\mathbf{x}$ stands for the input variables vector, $\boldsymbol{\Theta}$ is the vector of parameters of the k-th basis function $\boldsymbol{\nu}_k$, K is the number of basis functions and is the model output (estimated value of the DTM process variable). This general structure can approximate any continuous function with a superposition of a finite number of basic nonlinear continuous functions. The optimal dimension of the model (number of basis function K) and parameters ($\boldsymbol{\Theta}$) should be estimated from the available data [17, 18].

In prediction model building, the parameters are usually determined by regression in which all model parameters are estimated based on minimization of the output

error of approximation. However, this approach generally fails when plant data are available for modeling due to low plant data informativity as already stated in Section 1 [4]. Additionally, to ensure necessary robustness of the soft-sensor, a great number of ETM variables are used in modeling which results in a high dimensional and highly correlated input space [17].

To obtain a process model under these undesirable circumstances, methods based on the input space projection into a latent subspace should be used [8, 17]. The related model is a composition of two functions and can be presented with:

$$\hat{y} = \boldsymbol{f}_m(\mathbf{x}, \boldsymbol{\Theta}) = \boldsymbol{f}_r(\boldsymbol{\varphi}_j(\mathbf{x}, \boldsymbol{\alpha}_j); \boldsymbol{\beta}), \quad j = 1, 2, \ldots, J, \tag{2}$$

where $\boldsymbol{\varphi}(\cdot)$ is the input space into latent space projection function, $\boldsymbol{f}_r(\cdot)$ is the latent space into output space projection function and J is the dimension of input space projection (i.e. number of latent variables). The vector of model parameters $\boldsymbol{\Theta}$ is in this case divided into the input projection vectors α_j and the vector of regression parameters $\boldsymbol{\beta}$. Apart from the selection of appropriate functions, this model structure requires the definition of two separate criteria for parameter estimation [17]. There are a number of different linear or nonlinear methods available for model building which differ in the selected functions and criteria that are used in the model (2) [4, 17].

As stated before, model adaptation is necessary to ensure long-term soft-sensor accuracy. Linear models are usually used for the purpose of adaptive soft-sensor development due to two main reasons:

- online model parameter updating usually provides satisfactory description of the nonlinear processes even when linear models are used,
- it is much easier to estimate the parameters of linear models than nonlinear models in online operation, especially if recursive techniques are applied.

The linear model, based on the input space projection, is a special case of (2) and can be presented as:

$$\hat{y} = \sum_{j=1}^{J} \beta_j z_j = \sum_{j=1}^{J} \beta_j \sum_{i=1}^{m} \alpha_{ij} x_i, \tag{3}$$

or in matrix form:

$$\hat{y} = \mathbf{z}\boldsymbol{\beta} = \mathbf{x}\mathbf{A}\boldsymbol{\beta}, \tag{4}$$

where m is the number of ETM variables, $\mathbf{x} = [x_1, \ldots, x_m]$ is a sample of ETM variables, $\mathbf{z}^{1\mathrm{x}J}$ is the vector of latent variables, $\mathbf{A}^{m\mathrm{x}J}$ is the projection matrix that contains the column vectors $\boldsymbol{\alpha}_j$ and $\boldsymbol{\beta}^{J\mathrm{x}1}$ is the vector of regression coefficients. Since both parts of the model (2) are linear in nature, this two-level model can be presented as a one-level model in a way such that the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are combined into a single vector of parameters $\mathbf{b}^{m\mathrm{x}1}$:

$$\hat{y} = \mathbf{x}\mathbf{b}. \tag{5}$$

This model form is more suitable for online implementation.

Linear methods for process modeling with input space projection are based on methods developed in the field of multivariate statistical analysis, such as Principal Component Analysis (PCA), Partial Least Squares (PLS) and Continuum Regression (CR) [17]. The prediction capabilities of the built model are very important in soft-sensor development. Generally speaking, models based on the PCA method have lower prediction capabilities than PLS and CR based models [8]. Although CR models usually outperform PLS based models, PLS based models are predominant in practical applications due to difficult parameter estimation of the CR method. Apart from that, the recursive CR method has not yet been developed. Thus, PLS based models can be regarded as the referent models for adaptive estimator development. Details about parameter estimation of PLS based models are presented in Section 3.

2.2 Model Adaptation

Adaptive models are models which posses the ability to automatically change its properties during online operation [9]. To perform model adaptation, the model has to be extended with a mechanism that performs model changing according to the selected criteria and current state of the process. Samples of input and output variables that are acquired during online operation give information about the current state of the process. Generally, model adaptation is performed through model parameters update or through model structure and parameters change. In practice, only model parameters are usually updated starting from parameters estimated on the initial data set (i.e. in offline phase of model building). However, when only model parameters are updated it is assumed that the model structure is correctly selected, i.e. it is capable enough to describe the process in all possible working conditions. Thus, model structuring should be carefully performed during (offline) initial model building.

For online model parameter updating, different criteria and techniques are used. In general, an adaptive procedure can be presented as a function $\boldsymbol{g}(\cdot)$ which adapts the model $\boldsymbol{f}_m(\cdot)$ during online operation [9]:

$$\boldsymbol{f}_{m,k}(\boldsymbol{\Theta}_k) = \boldsymbol{g}(\boldsymbol{f}_{m,k-1}(\boldsymbol{\Theta}_{k-1}), D_k), \tag{6}$$

where k refers to the current step of model adaptation. Adaptation is based on the model that was derived in the

last adapting procedure $\boldsymbol{f}_{m,k-1}(\cdot)$ and the available data D_k.

In principle, batch techniques (that are used in offline model building) can be used for model adaptation such that the model is recalculated from scratch after each new sample is acquired. In that case, the adaptation function has only the input D_k. Because the data set used for model parameter estimation D_k is continuously increasing, this approach is impractical for online use. In addition, the influence of the online acquired samples is continuously decreasing so the model is not adapting to the current state of the process.

In order to avoid the aforementioned drawbacks, and to still implement batch techniques in adaptive model building, the moving window method emerged [9]. Model parameter estimation is performed only on a limited number of samples defined by the window size W:

$$D_k = \{[\mathbf{x}_{k+1-W}, \ldots, \mathbf{x}_k], [y_{k+1-W}, \ldots, y_k]\}, \quad (7)$$

where $\mathbf{x}_k$ and y_k are the last acquired sample of ETM and DTM variables (see Fig. 1). As the new samples are acquired and added to the window, the oldest are removed from the window. The size of the window W determines the number of the most recent complete input-output samples that are used in adaptation, i.e. the speed of adaptation. The moving window method applied to PLSR model is discussed in Section 3.2 in more details.

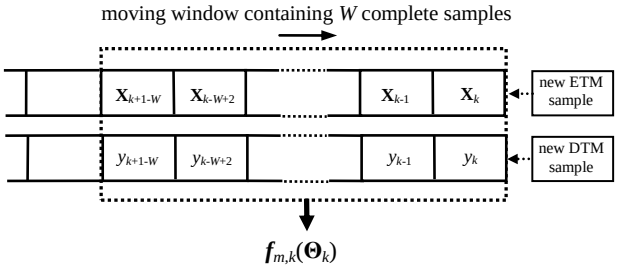


Fig. 1. Moving window method

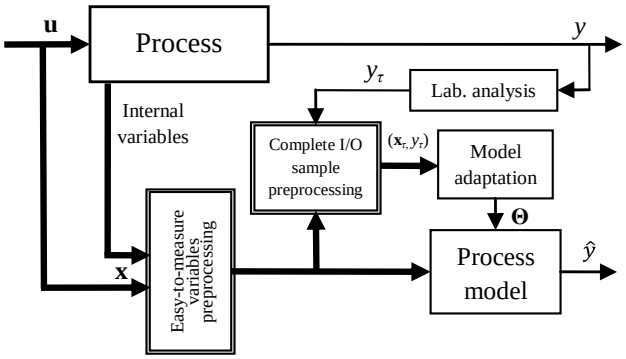
Recursive techniques use the model calculated in the previous step of adaptation $\boldsymbol{f}_{m,k-1}(\cdot)$ and the new data available from that point in time D_k (only one complete sample $\{\mathbf{x}_k, y_k\}$ in the case of sample-wise adaptation) [9]. A forgetting factor is usually used to down-weight the previous model. It defines distribution between learning of new information and forgetting the old information, similarly to the window size in moving window method.

The window size in a moving window approach and the forgetting factor in recursive techniques are usually determined in the offline phase of model building. Although the size of the window and the forgetting factor value can be changed during online operation to obtain a more accurate soft-sensor [9, 13], this is rarely found in practice.

2.3 Additional Tasks in Estimator Implementation

After the estimator has been developed its performance should be monitored. When the need for adaptation is detected, an adaptation procedure should be triggered. This can be accomplished by comparing the estimator output with the results of laboratory analysis, or in the case when soft-sensor works as a back-up sensor, with the values obtained by a hardware sensor. Therefore, laboratory analysis cannot be completely dismissed but its frequency could be reduced drastically and consequently reducing the cost of the laboratory analysis.

Practically, model adaptation is usually performed at regular intervals, so there is no need for some special monitoring procedure. However, adaptation must be performed on reliable samples, i.e. erroneous samples must be detected and discarded to prevent the estimator from adapting to disturbances. Therefore, quality preprocessing of the incoming data is very important in the adaptation procedure [10]. Due to the long duration of laboratory analysis, ETM variables can be additionally preprocessed when used for model adaptation, in comparison to the situation when they are used for DTM variable estimation (see Fig. 2).



u – input process variables, **x** – easy-to-measure variables, y – actual (unmeasurable) value of difficult-to-measure variable, y_τ – value of difficult-to-measure variable obtained by laboratory analysis, $\hat{y}$ – estimated value of difficult-to-measure variable, $(\mathbf{x}_\tau, y_\tau)$ – preprocessed input-output sample, τ – duration of laboratory analysis, $\boldsymbol{\Theta}$ – model parameters.

Fig. 2. Principal schema of adaptive estimator with data preprocessing

Apart from model adaptation, parameters of preprocessing methods as well as scaling parameters should be updated during online operation of the estimator. More about practical development and maintenance of soft-sensors can be found in [19–21].

3 METHODS FOR MODEL ADAPTATION

In the last few years an increased use of different intelligent methods can be noticed, such as Neuro-Fuzzy methods and other hybrid methods in soft sensor development [3, 4]. Nevertheless, PLS based soft sensors are still very common in practical use [22–25], due to its simplicity and clear mathematical background. In this paper, PLS method is the basis for adaptive model building.

3.1 PLSR Model

As already mentioned in Section 2.1, models with two levels of projection can give desired estimator accuracy even if plant data are used for process modeling and when the DTM variable is estimated from a great number of ETM variables. In a single output PLSR model, linear regression is performed on the latent variables obtained by linear input space projection in the directions obtained by PLS method. More precisely, the directions of projection $\boldsymbol{\alpha}$ are obtained by maximizing the following criterion [8, 26]:

$$\max_{\boldsymbol{\alpha}_j}[\mathrm{corr}^2(\mathbf{y}, \mathbf{X}\boldsymbol{\alpha}_j)\mathrm{var}(\mathbf{X}\boldsymbol{\alpha}_j)], \tag{8}$$

where $\mathbf{X}^{n\mathrm{x}m}$ and $\mathbf{y}^{n\mathrm{x}1}$ are available data containing n complete input-output samples for modeling. The parameters $\boldsymbol{\beta}$ of the regression part of the model are obtained by minimizing criterion:

$$\max_{\boldsymbol{\beta}}(\|\mathbf{y} - \hat{\mathbf{y}}\|^2). \tag{9}$$

Solutions of the optimization problem (8) are in fact eigenvectors of the matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{y})^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{y})$, so the directions of projection can be found by solving the eigenvalue problem. The most popular algorithms for calculating directions of projection are the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm [27] and kernel algorithm [28].

The general PLS algorithm is based on the decomposition of the predictor matrix $\mathbf{X}^{n\mathrm{x}m}$ (ETM variables) and the response matrix $\mathbf{Y}^{n\mathrm{x}p}$ (DTM variables) into sums of rank one component matrices [26, 27]:

$$\mathbf{X} = \sum_{i=1}^{J} \mathbf{t}_i \mathbf{p}_i^{\mathrm{T}} + \mathbf{E} = \mathbf{T}\mathbf{P}^{\mathrm{T}} + \mathbf{E}, \tag{10}$$

$$\mathbf{Y} = \sum_{i=1}^{J} \mathbf{u}_i \mathbf{q}_i^{\mathrm{T}} + \mathbf{F} = \mathbf{U}\mathbf{Q}^{\mathrm{T}} + \mathbf{F}, \tag{11}$$

where $\mathbf{t}^{n\mathrm{x}1}$ and $\mathbf{u}^{n\mathrm{x}1}$ are latent score vectors, $\mathbf{p}^{m\mathrm{x}1}$ and $\mathbf{q}^{p\mathrm{x}1}$ are corresponding loading vectors, $\mathbf{E}^{n\mathrm{x}m}$ and $\mathbf{F}^{n\mathrm{x}p}$ are the input and output residual matrices and J is number of latent variables. This decomposition can be obtained by the following steps [13]:

1. scale matrices $\mathbf{X}$ and $\mathbf{Y}$ such that columns have zero mean and unit variance, set $j = 1$,
2. compute the following quantities: $\mathbf{w}_j$, $\mathbf{t}_j$, $\mathbf{q}_j$ and $\mathbf{p}_j$ using either NIPALS or kernel algorithm,
3. deflate $\mathbf{X}$ and $\mathbf{Y}$ by subtracting the computed latent vectors from them:

$$\mathbf{X}_{j+1} = \mathbf{X}_j - \mathbf{t}_j \mathbf{p}_j^{\mathrm{T}}, \tag{12}$$

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j - \mathbf{t}_j \mathbf{q}_j^{\mathrm{T}}, \tag{13}$$

4. increase j by 1 and go to step 2 to compute the next latent vector.

Vectors $\mathbf{w}_j$ are loading weights which provide orthogonal $\mathbf{t}_j$ scores [26]. Model prediction for the new sample $\mathbf{x}^{1\mathrm{x}m}$ is given by:

$$\hat{\mathbf{Y}} = \mathbf{x}\mathbf{B}_{\mathrm{PLS}} = \mathbf{x}\mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{B}\mathbf{Q}^{\mathrm{T}}, \tag{14}$$

where $\mathbf{B}_{\mathrm{PLS}}$ is the regression matrix relating input and output, $\mathbf{W}$, $\mathbf{P}$ and $\mathbf{Q}$ are the matrices containing vectors $\mathbf{w}$, $\mathbf{p}$ and $\mathbf{q}$, and $\mathbf{B}$ is the matrix of regression coefficients regarding $\mathbf{T}$ and $\mathbf{U}$:

$$\mathbf{U} = \mathbf{T}\mathbf{B}. \tag{15}$$

The single output PLSR model is usually used in soft-sensor development. In terms of model with the two levels of projection (2), the PLSR model is used in the following form when estimation of DTM variable based on sample of ETM variable $\mathbf{x} = [x_1, \ldots, x_m]$ is needed:

$$\hat{\mathbf{Y}} = \mathbf{z}\boldsymbol{\beta} = \mathbf{x}\mathbf{A}_{\mathrm{PLS}}\boldsymbol{\beta} = \mathbf{x}\mathbf{b}_{\mathrm{PLS}}, \tag{16}$$

where $\mathbf{A}_{\mathrm{PLS}}^{m\mathrm{x}J}$ is the projection matrix and $\mathbf{b}_{\mathrm{PLS}}^{m\mathrm{x}1}$ are the overall model parameters in terms of the PLS criterion. Similarly to (14), the projection matrix $\mathbf{A}_{\mathrm{PLS}}$ is obtained as $\mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}$. Projection to the latent subspace is obtained as:

$$\mathbf{Z} = \mathbf{X}\mathbf{A}_{\mathrm{PLS}}, \tag{17}$$

while the coefficients of the regression part of the model (16) are obtained from the latent variable matrix $\mathbf{Z}$ according to the following equation:

$$\boldsymbol{\beta} = (\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{y}. \tag{18}$$

It is especially important that the ETM variables as well as the number of latent variables J are correctly selected during offline modeling procedure to ensure that the model can describe process input-output relationship satisfactorily. The model structure is usually not changed during online operation, i.e. only model parameters are updated.

An important issue in model building is data standardization which has to be performed before model parameter estimation/updating. It corresponds to data centering

and scaling to the unit variance [13–15]. The data matrix $\mathbf{X}_0^{nxm}$, which is used for initial model building, is standardized according to the following equation:

$$\mathbf{X} = (\mathbf{X}_0 - \mathbf{1}_n\overline{\mathbf{x}}^{\mathrm{T}})\mathbf{\Sigma}^{-1}, \tag{19}$$

where $\mathbf{X}$ is standardized matrix, $\mathbf{1}_n = [1, 1, \ldots, 1]^{\mathrm{T}}$ is a vector of length n, $\overline{\mathbf{x}} = [\overline{x}_1, \ldots, \overline{x}_m]^{\mathrm{T}}$ is a vector of mean values for each column in $\mathbf{X}_0$ and $\mathbf{\Sigma} = \mathrm{diag}\{\sigma_1, \ldots, \sigma_m\}$ is a diagonal matrix with the standard deviations of each column in $\mathbf{X}_0$. The vector $\mathbf{y}_0$ can be standardized in a similar way.

3.2 Moving Window Method

One of the simplest approaches for model adaptation is moving window method which is often called quasi-recursive technique. As the new samples are acquired, the window slides along the data so that newest samples are included and the oldest are excluded from the model (see for example [24]). Model parameters can be recalculated with every new sample that is acquired (sample wise) or after a certain number of data is acquired (block wise). This adaptive mechanism can be easily and directly applied to the PLSR model type.

Since process variables can change significantly over a long time period, it is necessary to recursively update mean values and standard deviations of variables used in modeling and prediction as the new samples are acquired [9, 15]:

$$\overline{\mathbf{x}}_k = \frac{N-1}{N}\overline{\mathbf{x}}_{k-1} + \frac{1}{N}\mathbf{x}_k, \tag{20}$$

$$\begin{aligned}\sigma_{i,k}^2 &= \frac{N-2}{N-1}\sigma_{i,k-1}^2 + \\ &+(\overline{x}_{i,k} - \overline{x}_{i,k-1})^2 + \frac{1}{N-1}(x_{i,k} - \overline{x}_{i,k})^2,\end{aligned} \tag{21}$$

where k is step of updating, $\sigma_{i,k}$ and $\overline{x}_{i,k}$ are the standard deviation and mean value of i-th variable in the k-th step and $N = W$ is the number of samples in the window. Another approach is to calculate the mean values and standard deviations with every model update from the data inside window.

The advantage of the moving window method lies in the fact that methods for offline model building can be used, so it can easily be implemented. A drawback of the method is the appropriate selection of the window size. Window size defines speed of adaptation. The longer the window the old process data dominate while with a shorter window information about near past is dominant. Therefore, the current state of the process can be described if the model is estimated on a narrow window. However, data in a narrow window can sometimes insufficiently represent the state of the process (there are not enough samples or there are a lot of disturbances superimposed on the samples in a window). This can result in poor parameter estimation or model adaptation to some disturbing operating conditions. On the other hand, if the window is too long, the model will adapt slowly and will have a lower accuracy. Appropriate window selection, i.e. trade-off between forgetting of old information and learning of new information, is therefore very important in adaptive model building based on data. Another possible drawback of the moving window method is relatively high memory requirement if a long window is used.

3.3 Recursive NIPALS Algorithm

The first recursive partial least square (RPLS) algorithm was proposed in [11]. Improvements of this algorithm with some other extensions were reported in [14]. Recursive PLS algorithm consists of these steps:

1. Scale initial data matrices $\mathbf{X}_0$ and $\mathbf{Y}_0$ to zero mean and unit variance according to equation (19), set $k = 0$,
2. Derive PLS model by NIPALS algorithm: $\{\mathbf{X}_k, \mathbf{Y}_k\} \rightarrow \{\mathbf{T}_k, \mathbf{W}_k, \mathbf{P}_k, \mathbf{B}_k, \mathbf{Q}_k\}$ and increase k by one,
3. When a new pair of data $\{\mathbf{x}, \mathbf{y}\}$ is available, scale it and formulate:

$$\mathbf{X}_k = \begin{bmatrix}\lambda\mathbf{P}_{k-1}^{\mathrm{T}} \\ \mathbf{x}\end{bmatrix}, \mathbf{Y}_k = \begin{bmatrix}\lambda\mathbf{B}_{k-1}\mathbf{Q}_{k-1}^{\mathrm{T}} \\ \mathbf{y}\end{bmatrix}, \tag{22}$$

4. return to step 2.

The forgetting factor λ defines the strength of adaptation. A smaller λ means faster adaptation to the new data. For $\lambda = 1$, new sample and old samples (old model) have the same influence on the resulting model. It also has to be pointed out that the number of latent variables that are used in the updating procedure must be equal or greater than the rank of matrix $\mathbf{X}_k$. However, only the most important latent variables are used for prediction. Block-wise RPLS, derived in [14], builds a PLS model on the new block of data and then combines it with old PLS model. It is equal to the presented algorithm, only equation (22) is changed to:

$$\mathbf{X}_k = \begin{bmatrix}\lambda\mathbf{P}_{k-1}^{\mathrm{T}} \\ \mathbf{P}_k^{\mathrm{T}}\end{bmatrix}, \mathbf{Y}_k = \begin{bmatrix}\lambda\mathbf{B}_{k-1}\mathbf{Q}_{k-1}^{\mathrm{T}} \\ \mathbf{B}_k\mathbf{Q}_k^{\mathrm{T}}\end{bmatrix}. \tag{23}$$

Equations (20) and (21) can be used for recursive online data scaling (in this case N is the total number of data points). However, the influence of the new data points diminishes with increasing N, so exponentially weighted scaling can be used to track mean values and standard deviations of the data [15]:

$$\overline{\mathbf{x}}_k = \lambda\overline{\mathbf{x}}_{k-1} + (1-\lambda)\mathbf{x}_k, \tag{24}$$

$$\sigma_{i,k}^2 = \lambda(\sigma_{i,k-1}^2 + (\overline{x}_{i,k} - \overline{x}_{i,k-1})^2) + \\ +(1-\lambda)(x_{i,k} - \overline{x}_{i,k})^2. \tag{25}$$

3.4 Recursive Kernel PLS Algorithm

Recursive exponentially weighted kernel PLS algorithm was proposed in [13]. Since the basis of this recursive algorithm is the kernel algorithm [28], covariance data matrices $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ play a central role in the algorithm.

In [29], an improved kernel algorithm is proposed which consists of the following steps:

1. Calculate covariance matrices $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$, set $j = 1$,
2. Calculate $\mathbf{Y}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ as:

$$(\mathbf{Y}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{Y})_j = (\mathbf{X}^{\mathrm{T}}\mathbf{Y})_j^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{Y})_j, \tag{26}$$

3. Calculate vector $\mathbf{q}_j$ which corresponds to the largest eigenvalue of $\mathbf{Y}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ and after that $\mathbf{w}_j$ as:

$$\mathbf{w}_j = \mathbf{X}^{\mathrm{T}}\mathbf{Y}\mathbf{q}_j, \quad \mathbf{w}_j = \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|}, \tag{27}$$

4. If $j = 1$ set $\mathbf{r}_1 = \mathbf{w}_1$, else calculate $\mathbf{r}_j$ as:

$$\mathbf{r}_j = \mathbf{w}_j - \sum_{i=1}^{j-1} \mathbf{p}_i\mathbf{w}_j\mathbf{r}_i, \tag{28}$$

5. Calculate the score vector $\mathbf{t}_j$ and loadings vectors $\mathbf{p}_j$ and $\mathbf{q}_j$ of the input and output data as follows:

$$\mathbf{t}_j = \mathbf{X}\mathbf{r}_j, \tag{29}$$

$$\mathbf{p}_j^{\mathrm{T}} = \frac{\mathbf{r}_j^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})}{\mathbf{r}_j^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{r}_j}, \tag{30}$$

$$\mathbf{q}_j^{\mathrm{T}} = \frac{\mathbf{r}_j^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{Y})_j}{\mathbf{r}_j^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}}\mathbf{X})\mathbf{r}_j}, \tag{31}$$

6. Deflate the covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{Y})_{j+1} = (\mathbf{X}^{\mathrm{T}}\mathbf{Y})_j - \mathbf{p}_j\mathbf{q}_j^{\mathrm{T}}(\mathbf{t}_j^{\mathrm{T}}\mathbf{t}_j), \tag{32}$$

7. Set $j = j + 1$ and go to step 2 while $j <= J$,
8. Calculate the overall model parameters as:

$$\mathbf{B}_{\mathrm{PLS}} = \mathbf{W}(\mathbf{P}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{Q}^{\mathrm{T}} = \mathbf{R}\mathbf{Q}^{\mathrm{T}}, \tag{33}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_l]$, $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_l]$, $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_l]$ and $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_l]$.

Generally, the kernel algorithm is many times faster than the NIPALS algorithm [13]. Since only the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{Y}$ has to be deflated, the improved kernel algorithm is 2-5 times faster than the original kernel algorithm proposed in [28].

To make this algorithm recursive, the following update of the covariance matrices is proposed when new sample $\{\mathbf{x}, \mathbf{y}\}$ is available [13]:

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})_k = \lambda(\mathbf{X}^{\mathrm{T}}\mathbf{X})_{k-1} + \mathbf{x}^{\mathrm{T}}\mathbf{x}, \\ (\mathbf{X}^{\mathrm{T}}\mathbf{Y})_k = \lambda(\mathbf{X}^{\mathrm{T}}\mathbf{Y})_{k-1} + \mathbf{x}^{\mathrm{T}}\mathbf{y}, \tag{34}$$

where λ is a forgetting factor just like in the recursive PLS proposed in [14]. In that way old data are exponentially discounted. Naturally, for $\lambda = 1$ there is no discounting of the past data.

Mean values and standard deviations of variables can be updated from the covariance matrix $(\mathbf{X}^{\mathrm{T}}\mathbf{X})_k$ irrespective of whether a constant or variable forgetting factor is used [13].

3.5 Just-In-Time Learning

Just-In-Time Learning (JITL, also known as lazy learning or model on demand) can be used to develop adaptive process model [30–32]. In contrast to the "classical" adaptive modeling where the initial model is built offline and updated during online operation, in JITL the model is built from scratch whenever the estimation of DTM variable is needed. Such modeling can cope with nonlinear and time-varying behaviour successfully although simple linear models are used for describing process input-output relationship. JITL for soft-sensing consists of the following steps [30]:

1. If complete input-output sample is available, store it in the process database.
2. When prediction of the DTM variable based on input sample $\mathbf{x}_k$ is needed, process model is dynamically built from the similar samples $\{\mathbf{X}_s, \mathbf{y}_s\}$ stored in the database (see Fig. 3).
3. The constructed model is discarded after its use for the DTM variable estimation since it is only valid for the operating condition characterized by the current query.

Suppose that the database consists of the samples collected in the matrix $\mathbf{X}^{nxm}$ and vector $\mathbf{y}^{nx1}$. When an estimation based on newly collected sample $\mathbf{x}_k$ is needed, the matrix $\mathbf{X}$ is searched for the samples $\mathbf{x}_i$ that are similar to the sample $\mathbf{x}_k$ according to the following measure [30]:

$$s_i = \gamma\sqrt{e^{-d_i^2}} + (1-\gamma)\cos\theta_i, \quad i = 1, \ldots, n, \tag{35}$$

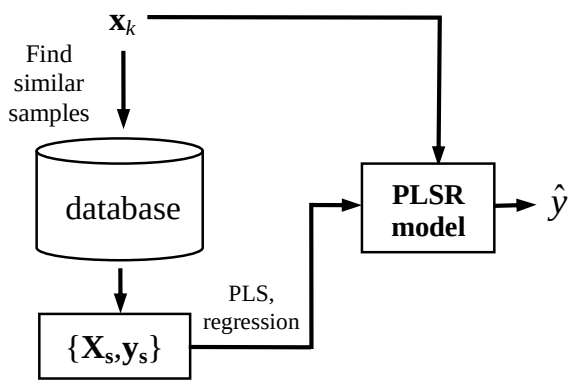


Fig. 3. Just-in-time learning method for adaptive PLSR model building

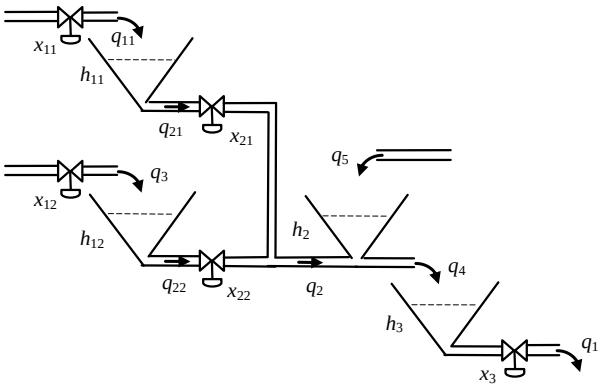


Fig. 4. Principal schema of the fluid storage process

where γ is a weight parameter, d_i is Euclidean distance between $\mathbf{x}_k$ and $\mathbf{x}_i$ and $\cos\theta_i$ is the cosine of the angle between $\Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\Delta\mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$:

$$d_i = \|\mathbf{x}_k - \mathbf{x}_i\|_2 , \tag{36}$$

$$\cos\theta_i = \frac{\Delta\mathbf{x}_k^{\mathrm{T}} \Delta\mathbf{x}_i}{\|\Delta\mathbf{x}_k\|_2 \cdot \|\Delta\mathbf{x}_i\|_2}. \tag{37}$$

If $\cos\theta_i$ is a negative number, the sample is discarded. Only the most relevant samples $\{\mathbf{x}_i, y_i\}$ that correspond to the largest values of s_i are used for PLSR model building. The resulting model is called JIT-PLSR model.

JIT-PLSR model can have better prediction capabilities than recursive PLS, especially when there are abrupt changes in process characteristic. A potential drawback of this approach is large memory requirement. Besides that, since the model has to be built upon every query, online computational load becomes very large [32].

4 NUMERICAL RESULTS

The PLS method for process model building and different methods for model adaptation discussed in Section 3 were analyzed on a simulated example of the fluid storage process (see Fig. 4). This process is nonlinear with a number of (highly) correlated process variables. Different errors can be simulated which are usually found in real data sets, like sensors drifts, noise, outliers, missing values and so on. Apart from that, other different scenarios can be simulated, like sensors faults, operating point changes or other time-varying process behaviors [10]. Therefore, it is appropriate for adaptive estimator efficiency testing. The level of the fluid in the third tank (h_2) is supposed to be a DTM process variable, i.e. its measurements are available with much lower frequency than other process variables. The flow rates q and positions x of the controlled valves are supposed to be ETM process variables, so a total of 13 ETM variables are available. During simulation, which covered a time period of 25 hours, variables were sampled every 6 seconds. However, every 50^{th} sample of DTM variable was used for model building/adaptation while the rest of DTM samples were used only for model testing. In order to achieve more realistic situation, noise and outliers were added to the flow measurements. Gaussian noise with zero mean and appropriate variance was superimposed to all flow measurements. Values of approximately 8% of randomly chosen samples were changed by adding/subtracting about 10% of the true value in order to simulate outlying observations which can appear in real measurements and can negatively affect modeling procedure. Operating point change was introduced at 5000^{th} sample by adding a slow ramp to the x_{11} variable in order to simulate non-stationary process behavior.

The first 4500 samples, which were used for initial process model building, were preprocessed offline by wavelet denoising [6]. Due to the lower sampling frequency of level h_2, only 90 complete input-output samples were available for offline process model building. The initial PLSR model was determined by kernel PLS algorithm (see Subection 3.4). The rest of the data (10500 samples) were used for non-adaptive and adaptive process models testing. Models that serve as DTM estimators were tested in online manner, i.e. as the new samples of ETM variables were acquired, prediction of DTM variable were made and compared. If a complete input-output sample was available, adaptation procedure was triggered in the case of adaptive models. In all experiments the moving average filter was implemented for online data denoising of ETM variables [6, 10]. The number of latent variables in PLSR model was fixed to 6.

4.1 Moving Window PLSR Model

The estimation result of the non-adaptive PLSR model is shown in Fig. 5 together with the true values of DTM variable. The estimation result of the adaptive PLSR model that was updated by moving window method (window covers 50 samples) using kernel PLS algorithm is shown in Fig. 6. It can be noticed that the estimation precision of the non-adaptive PLSR model is decreased after 5000^{th} sample while adaptive PLSR model successfully tracks process changes. This is clearer in Fig. 7 where absolute error of the models estimation is shown.

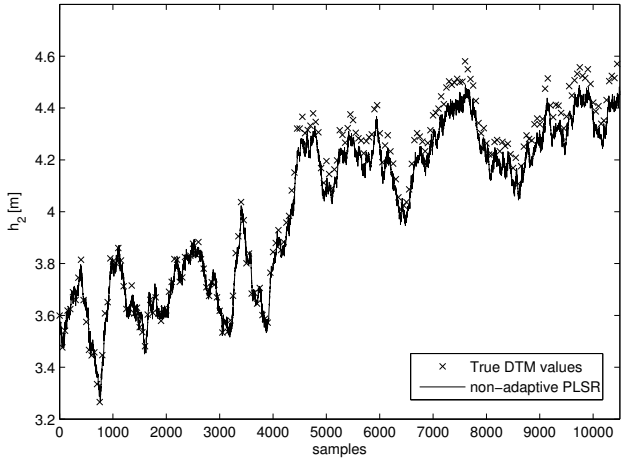


Fig. 5. DTM variable estimation by non-adaptive PLSR model

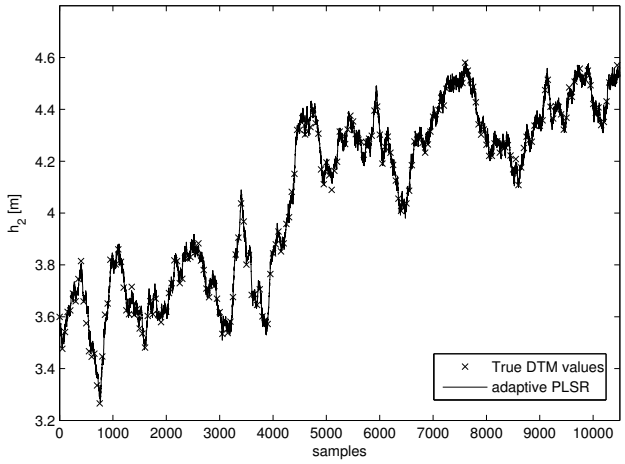


Fig. 6. DTM variable estimation by moving window adaptive PLSR model

The influence of the window size in the moving window adaptation method is shown in Table 1 in terms of the mean squared error (MSE) calculated over the samples of

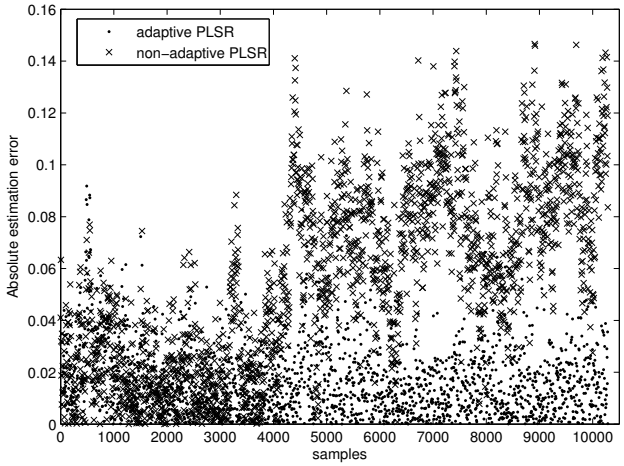


Fig. 7. Absolute error of h_2 estimation

DTM variable that are not used in adaptation procedure. It can be noticed that adaptation with too wide a window

Table 1. Comparison of moving window PLSR models with different window size

Model	Window size	MSE
adaptive PLSR	30	$5.3905 \cdot 10^{-4}$
adaptive PLSR	50	$4.8909 \cdot 10^{-4}$
adaptive PLSR	80	$5.2155 \cdot 10^{-4}$
Non-adaptive PLSR	-	$43.6791 \cdot 10^{-4}$

(80 samples) causes slow model adaption and therefore a higher estimation error. On the other hand, too narrow a window (30 samples) causes quick adaptation but due to the limited number of samples and errors in samples, the model can have a higher estimation error than the model with a moderate window size (50 samples). Optimal window size selection depends on the properties of the modeled process. Hereby, process operator experience about process time-varying behavior can greatly help in the window size selection.

4.2 Recursive NIPALS Algorithm

Recursive NIPALS algorithm was implemented according to (22). DTM variable estimation and absolute error of the estimation are not showed since they are very similar to Fig. 6 and Fig. 7. Table 2 shows the mean squared error of PLSR models updated by recursive NIPALS algorithm with different forgetting factors. As stated before, the forgetting factor λ defines the strength of adaptation since it down weights the old model. It can be concluded that choosing the proper forgetting factor is crucial for the model estimation precision. Apart from that, the presented results in Table 2 show that recursive update of

the variables mean values according to (24) should be performed since smaller estimation error is achieved.

Table 2. Comparison of PLSR models updated by recursive NIPALS algorithm

Forgetting factor λ	Means update	MSE
0.92	No	$6.0150 \cdot 10^{-4}$
0.92	Yes	$5.9201 \cdot 10^{-4}$
0.98	No	$4.0759 \cdot 10^{-4}$
0.98	Yes	$4.0031 \cdot 10^{-4}$
0.99	No	$4.1137 \cdot 10^{-4}$
0.99	Yes	$3.9885 \cdot 10^{-4}$

4.3 Recursive Kernel PLSR Model

Recursive exponentially weighted kernel PLS algorithm was implemented according to (24)-(34). Table 3 shows mean squared error of such adaptive PLSR models. According to Table 3, similar results to the recursive NIPALS PLSR models or moving window PLSR models were obtained. The forgetting factor λ plays the same role as in recursive NIPALS algorithm although it weights data samples directly. Smaller forgetting factor means that model relies more on the new samples and quickly discounts the old data. On the other hand, a bigger forgetting factor means that model discounts the old samples more slowly and for $\lambda = 1$ there is no discounting of the old samples. The optimal value of forgetting factor is different for different time-varying processes. Therefore, its role is similar to the size of window in moving window adaptation method. Variable means should be recursively updated since higher estimation precision is obtained.

Table 3. Comparison of recursive kernel PLSR models

Forgetting factor λ	Means update	MSE
0.92	No	$4.9058 \cdot 10^{-4}$
0.92	Yes	$4.7868 \cdot 10^{-4}$
0.98	No	$4.5478 \cdot 10^{-4}$
0.98	Yes	$4.4339 \cdot 10^{-4}$
0.99	No	$5.0668 \cdot 10^{-4}$
0.99	Yes	$4.9415 \cdot 10^{-4}$

4.4 JIT-PLSR Model

JITL method was implemented according to the algorithm and equations presented in Section 3.5. Results of the JIT-PLSR model testing are presented in Table 4 for different number of samples that were used for local model building. It can be concluded that the number of samples which are used in local model building should be carefully chosen. The presented results are slightly worse than results obtained by moving window PLSR, recursive NIPALS PLSR or recursive kernel PLSR models (see Table 1, 2 and 3). This can be explained by the fact that some of the old samples can be involved in local modeling procedure although they do not represent the true state because the operating point is constantly changing in the test data. However, this model can be useful if abrupt changes are present in the data, such as change to some old process operating point which was already recorded in the process database.

Table 4. Comparison of JIT-PLSR models

Number of samples	MSE
30	$7.2063 \cdot 10^{-4}$
60	$6.0739 \cdot 10^{-4}$
80	$6.1806 \cdot 10^{-4}$

4.5 Other Evaluation Aspects of Model Adaptation Methods

Although all adaptive models have similar prediction accuracy, they are very different with respect to computational load and memory requirements. This is very important from the practical point of view when the estimator is actually implemented in plant computer system. Table 5 shows time in seconds that was needed for model testing on a standard PC[1]. Model testing includes data preprocessing, model adaptation and model prediction over available data.

Table 5. Duration of model testing in seconds

Model	Time in seconds
Moving window PLSR model with window size 50	16.29
Recursive NIPALS PLSR model	19.05
Recursive kernel PLSR model	16.67
JIT-PLSR model with number of samples equal to 60	35.08

Table 6. Memory requirements of different adaptive methods

Adaptive model type	Memory requirements
Moving window PLSR model	moderate (depends on window size)
Rec. NIPALS PLSR model	very low
Rec. kernel PLSR model	very low
JIT-PLSR model	high, constantly increasing as the new samples are acquired

As expected, JIT-PLSR model has very high computational load since new local PLSR model has to be built

[1] Intel Celeron M 1.4 GHz, 2 GB of RAM with Matlab 7.1 SP3

with each acquired sample. Additionally, JIT-PLS modeling approach requires storing all acquired samples which means that the process database is continuously increasing and local model building is slowing down since a bigger and bigger process database has to be searched (see Table 6). Moving window PLSR model and recursive kernel PLSR model has similar time execution since the same kernel PLS algorithm is implemented. However, moving window approach requires last W samples to be stored (where W is window size) while recursive kernel PLS algorithm requires only covariance matrix $M\mathrm{x}M$ to be stored. These characteristics of the adaptive methods should be taken into account when such estimators are implemented in process control equipment which can be quite computationally and memory limited.

5 CONCLUSION

Industrial processes exhibit some kind of time-varying behavior. Therefore, the estimation precision of estimators based on model with static parameters decreases during online operation. In order to avoid estimator precision degradation, an adaptation mechanism is usually implemented in online applications. This paper presents different adaptation methods which can be used for model adaptation. Adaptation methods are applied to the PLSR model and are evaluated on the data obtained by simulation. Results are compared and discussed in details. Guidelines for choosing the proper parameters of adaptation methods are also given. Apart from that, methods are analyzed in terms of computational load and memory requirements which are especially important when the estimator is implemented in the plant control equipment.

For the future work it would be interesting to see how the moving window and JITL method work with some other methods from machine learning field, such as Support Vector Regression. Additionally, the JITL method should be explored since its computational load can be decreased by storing already built local models.

REFERENCES

[1] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft sensors for monitoring and control of industrial processes*. Springer Verlag, 2007.

[2] T. McAvoy, "Intelligent "Control" Applications in the process industries," *Annual Reviews in Control*, vol. 26, no. 1, pp. 75–86, 2002.

[3] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Computers & Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[4] D. Slišković, R. Grbić, and Ž. Hocenski, "Methods for Plant Data-Based Process Modeling in Soft-Sensor Development," *Automatika*, vol. 52, no. 4, pp. 306–318, 2011.

[5] G. D. Gonzalez, "Soft sensors for processing plants," in *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials, IPMM'99.*, vol. 1, 1999.

[6] D. Slišković, R. Grbić, and E. K. Nyarko, "Data preprocessing in data based process modeling," in *Intelligent Control Systems and Signal Processing*, vol. 2, pp. 559–565, 2009.

[7] B. Lin, B. Recke, J. Knudsen, and S. Jorgensen, "A systematic approach for soft sensor development," *Computers & Chemical Engineering*, vol. 31, no. 5-6, pp. 419–425, 2007.

[8] D. Slišković, N. Perić, and I. Petrović, "Continuum Regression in Process Modeling Based on Plant Data," *Automatika*, vol. 46, no. 3–4, pp. 1–14, 2005.

[9] P. Kadlec, R. Grbić, and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors," *Computers & Chemical Engineering*, vol. 35, no. 1, pp. 1–24, 2011.

[10] D. Slišković, R. Grbić, and Ž. Hocenski, "Online data preprocessing in the adaptive process model building based on plant data," *Technical Gazette*, vol. 18, no. 1, pp. 41–50, 2011.

[11] K. Helland, H. Berntsen, O. Borgen, and H. Martens, "Recursive algorithm for partial least squares regression," *Chemometrics and Intelligent Laboratory Systems*, vol. 14, no. 1-3, pp. 129–137, 1992.

[12] S. Wold, "Exponentially weighted moving principal components analysis and projections to latent structures," *Chemometrics and Intelligent Laboratory Systems*, vol. 23, no. 1, pp. 149–161, 1994.

[13] B. S. Dayal and J. F. MacGregor, "Recursive exponentially weighted PLS and its applications to adaptive control and prediction," *Journal of Process Control*, vol. 7, no. 3, pp. 169–179, 1997.

[14] S. Joe Qin, "Recursive PLS algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998.

[15] W. Li, H. Yue, S. Valle-Cervantes, and S. Qin, "Recursive PCA for adaptive process monitoring," *Journal of Process Control*, vol. 10, no. 5, pp. 471–486, 2000.

[16] X. Wang, U. Kruger, and B. Lennox, "Recursive partial least squares algorithms for monitoring complex industrial processes," *Control Engineering Practice*, vol. 11, no. 6, pp. 613–632, 2003.

[17] D. Slišković, *Difficult-to-measure process variables estimation based on plant data*. Ph.d. thesis, Faculty of Electrical Engineering and Computing, Zagreb, 2005.

[18] B. Bakshi, "A common framework for the unification of neural, chemometric and statistical modeling methods," *Analytica Chimica Acta*, vol. 384, no. 3, pp. 227–247, 1999.

[19] I. Miletic, S. Quinn, M. Dudzic, V. Vaculik, and M. Champagne, "An industrial perspective on implementing on-line applications of multivariate statistics," *Journal of Process Control*, vol. 14, no. 8, pp. 821–836, 2004.

[20] J. Liu, R. Srinivasan, and P. N. SelvaGuru, "Practical challenges in developing data-driven soft sensors for quality prediction," in *18th European Symposium on Computer Aided Process Engineering*, vol. 25, pp. 961–966, 2008.

[21] L. Fortuna, S. Graziani, and M. Xibilia, "Soft sensors for product quality monitoring in debutanizer distillation columns," *Control Engineering Practice*, vol. 13, no. 4, pp. 499–508, 2005.

[22] T. Komulainen, M. Sourander, and S.-L. Jämsä-Jounela, "An online application of dynamic PLS to a dearomatization process," *Computers & Chemical Engineering*, vol. 28, no. 12, pp. 2611–2619, 2004.

[23] R. Sharmin, U. Sundararaj, S. Shah, L. Vande Griend, and Y.-J. Sun, "Inferential sensors for estimation of polymer quality parameters: Industrial application of a PLS-based soft sensor for a LDPE plant," *Chemical Engineering Science*, vol. 61, no. 19, pp. 6372–6384, 2006.

[24] S. Mu, Y. Zeng, R. Liu, P. Wu, H. Su, and J. Chu, "Online dual updating with recursive PLS model and its application in predicting crystal size of purified terephthalic acid (PTA) process," *Journal of Process Control*, vol. 16, no. 6, pp. 557–566, 2006.

[25] D. Wang and R. Srinivasan, "Eliminating the effect of multivariate outliers in PLS-based models for inferring process quality," in *19th European Symposium on Computer Aided Process Engineering*, vol. 26, pp. 755–760, 2009.

[26] H. Martens and T. Naes, *Multivariate calibration*. John Wiley & Sons Inc, 1992.

[27] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, no. 1, pp. 1–17, 1986.

[28] F. Lindgren, P. Geladi, and S. Wold, "The kernel algorithm for PLS," *Journal of Chemometrics*, vol. 7, no. 1, pp. 45–59, 1993.

[29] B. S. Dayal and J. F. MacGregor, "Improved PLS algorithms," *Journal of Chemometrics*, vol. 11, no. 1, pp. 73–85, 1997.

[30] C. Cheng and M. S. Chiu, "A new data-based methodology for nonlinear process modeling," *Chemical Engineering Science*, vol. 59, no. 13, pp. 2801–2810, 2004.

[31] Z. Ge and Z. Song, "Online monitoring of nonlinear multiple mode processes based on adaptive local model approach," *Control Engineering Practice*, vol. 16, no. 12, pp. 1427–1437, 2008.

[32] K. Fujiwara, M. Kano, S. Hasebe, and A. Takinami, "Soft-sensor development using correlation-based just-in-time modeling," *AIChE Journal*, vol. 55, no. 7, pp. 1754–1765, 2009.

Dražen Slišković was born in Osijek, Croatia, in 1962. He received his B.Sc., M.Sc. and PhD degrees in 1988, 1998 and 2005 from Faculty of Electrical Engineering and Computing, University of Zagreb. After graduation, he was employed in Elektroosijek, in the field of measurement, supervision and control. Currently he is with Faculty of Electrical Engineering, University of Osijek, participating in education in the field of Automatic Control. His research interests are process control, process modeling, estimation of difficult-to-measure process variables, diagnostics and fault detection.

Ratko Grbić was born in 1983 in Virovitica, Croatia. In 2006 he graduated from the Faculty of Electrical Engineering, University of Osijek, Croatia. He is currently a Ph.D. student at the same Faculty, where he works as a assistant at the Department of Industrial Plants and Automation. Teaching activities include undergraduate and graduate courses such as process control and process identification. His research interests include process modeling and estimation of difficult-to-measure process variables. He is a member of KoREMA and IEEE.

Željko Hocenski (1952) is working as a scientist and researcher in the area of industrial electronics, computer engineering, automation and process control. He received B.Sc. (1976), M.Sc. (1984) degree in Electrical Engineering and Ph.D. degree (1996) in Computing from the Faculty of Electrical Engineering and Computing (FER Zagreb), University of Zagreb, Croatia. He was working at the Institute of Electrical Engineering of holding Končar in Zagreb (1977-1984) in the fields industrial electronics and automation, microprocessor control and industrial communications. From 1984 is at J.J. Strossmayer University of Osijek, Faculty of Electrical Engineering, position full professor (2006). He was Vice-dean (1997-2003) and Dean (2003-2005). Now is the head of the Computer Engineering Department. Teaching activities include undergraduate and graduate courses. Scientific activities are in design, diagnosis, verification and validation of embedded computer systems as well as fault-tolerant computer systems for process control and automation. His research results are published in more than 80 scientific articles and conference papers, books, course-books, 20 studies/reports and projects related to production. Prof. Hocenski has particularly excelled in the leadership of national and international research projects. Awarded as author of three technical improvements and innovations. He is a member of KoREMA (Managing Board), IEEE (vice president of the IEEE Computer Society, Croatian section), ACM and SICE.

AUTHORS' ADDRESSES

Prof. Dražen Slišković, Ph.D.
Ratko Grbić, B.Sc.
Prof. Željko Hocenski, Ph.D.
Faculty of Electrical Engineering,
University of Osijek,
Kneza Trpimira 2b, HR-31000, Osijek, Croatia
email: {drazen.sliskovic, ratko.grbic,
zeljko.hocenski}@etfos.hr

Received: 2011-12-20
Accepted: 2012-12-14