Adaptive Identification and Predictive Control Using an Improved On-Line Sequential Extreme Learning Machine

Tiago Matias*, Francisco Souza*, Rui Araújo*, Saeid Rastegar*, and Jérôme Mendes*
*Institute of Systems and Robotics (ISR-UC), and
Department of Electrical and Computer Engineering (DEEC-UC),
University of Coimbra, Pólo II, PT-3030-290 Coimbra, Portugal
tmatias@isr.uc.pt, fasouza@isr.uc.pt, rui@isr.uc.pt, srastegar@isr.uc.pt, jermendes@isr.uc.pt

***Abstract*—This paper proposes a method for adaptive identification and predictive control using an online sequential extreme learning machine based on the recursive partial least-squares method (OS-ELM-RPLS). OL-ELM-RPLS is an improvement to the online sequential extreme learning machine based on recursive least-squares (OS-ELM-RLS) introduced in [1]. Like in the batch extreme learning machine (ELM), in OS-ELM-RLS the input weights of a single-hidden layer feedforward neural network (SLFN) are randomly generated, however the output weights are obtained by a recursive least-squares (RLS) solution. However, due to multicollinearities in the columns of the hidden-layer output matrix caused by the presence of redundant input variables or by a large number of hidden-layer neurons, the problem of estimation the output weights can become ill-conditioned. In order to circumvent or mitigate such ill-conditioning problem, it is proposed to replace the RLS method by the recursive partial least-squares (RPLS) method. The identification methodology is proposed for two application problems: (1) construction of a inferential model, and (2) the learning of a model for the Generalized predictive control (GPC) algorithm. The integration of the proposed adaptive identification method with the GPC results in an adaptive predictive control methodology. To validate and demonstrate the performance and effectiveness of the proposed methodologies, they are applied on modeling of two public regression data sets and on control of the flow through a simulated valve.**

I. Introduction

The design of an accurate model is one of the crucial steps in the construction of a inferential model and in model predictive control (MPC). In this process, multilayer feedforward neural networks (FFNN) have been used due to the good prediction capability. However, the training of FFNN using the back-propagation (BP) algorithm and its variants has been the bottleneck of the use of these networks in industrial applications [2] due to the large training time and the large amount of learning parameters (e.g. learning rate, number of learning epochs, stopping criteria, and other predefined parameters) that must be properly chosen to ensure convergence [3], being the linear models often preferred in comparison to multilayer FFNN. In order to overcome these problems in the construction of FFNN models, a new method called extreme learning machine (ELM) was proposed in [4]. These improvements provided by ELM make these models a valuable tool in industrial processes.

ELM is a batch learning algorithm for single hidden-layer FFNN (SLFN) where the input weights (weights of connections between the input variables and the neurons in the hidden-layer) and the bias of neurons in the hidden-layer are randomly assigned. The output weights (weights of connections between the neurons in the hidden-layer and the output neuron) are obtained using the Moore-Penrose (MP) generalized inverse, considering in the output neuron a linear activation function. However, in some applications a sequential learning should be preferred over the batch learning. One example of sequential learning application is in the on-line modeling of a process with a time-varying behavior. In this case, collecting a training data set that would be representative of all possible states and conditions of the process can be very difficult. These conditions include different intrinsic states in which the process can be operated, and also different states related to environmental changes, changes of the process input materials, etc. Due to this difficulty, an online adaptation tool should be used in order to construct a SLFN model that has the capability to self-adjust its parameters in order to provide a good estimation in each operation scenario. In [5], [1] an online sequential extreme learning machine based on the recursive least-squares (RLS) algorithm called OS-ELM-RLS was presented. However, the outputs of hidden neurons can have strong multicollinearities due to a large number of hidden nodes or due to redundancy in input variables. In such situations the output matrix of the hidden-layer, corresponding to a set of data samples, may have not full rank, which can result in an ill-conditioned problem to be solved by the least-squares solution, and in an unstable solution. In order to circumvent or mitigate the ill-condition problem, the RLS method can be replaced by the recursive partial least-squares (RPLS) method. In this paper, an improved online sequential extreme learning machine algorithm based on RPLS (OS-ELM-RPLS) is proposed. In the proposed methodology, the output weights of the SLFN are updated when a new data sample is available using a RPLS method and the number of latent variables is adapted using a leave-one-out validation: for each new sample, before updating the output weights vector, the preceding data sample is used to select the best number of latent variables. OS-ELM-RPLS was applied and compared with OS-ELM-RLS, RPLS, and OS-ELM-RPLSM over two real-world data sets. The OS-ELM-RPLSM is a modified version of OS-ELM-RPLS method, where the number of RPLS latent variables is not adapted online, but is selected by a 10-fold cross-validation procedure on the train data set. In all the experiments, the proposed OS-ELM-RPLS method always exhibits the best prediction performance. The identification methodology is proposed for two application problems: (1) construction of a inferential model, and (2) the learning of a model for the Generalized predictive control (GPC) algorithm. The integration of the proposed adaptive identification

method with the GPC results in an adaptive predictive control methodology. To validate and demonstrate the performance and effectiveness of the proposed methodologies, they are applied on modeling of two public regression data sets and on control of the flow through a simulated valve.

The paper is organized as follows. The SLFN architecture is overviewed in Section II. A review of the batch and sequential ELM is given in Section III. The proposed method is presented in Section IV. Section VI presents experimental results. Finally, concluding remarks are drawn in Section VII.

II. Single Hidden-Layer Feedforward Network Architecture

The neural network considered in this paper is a single hidden-layer feedforward neural network with n input variables, h hidden-layer neurons, and one neuron in the output layer. The output the SLFN at time-instant k is given by:

$$\hat{y}(k) = g\left(\mathbf{v}^T \mathbf{s}_k\right), \tag{1}$$

where $g(\cdot)$ represents the activation function of the output neuron, $\mathbf{v} = [b_O, v_1, \ldots, v_h]^T$ is the vector of output weights and bias, and $\mathbf{s}_k = [1, s_1(k), \ldots, s_h(k)]^T$ is the vector of the outputs of the h hidden-layer neurons (the first element is for the output bias) given by:

$$\mathbf{s}_k = \mathbf{\Gamma}\left(\mathbf{W}^T \mathbf{x}_k\right), \tag{2}$$

where

$$\mathbf{W} = \begin{bmatrix} b_1 & b_2 & \ldots & b_h \\ w_{11} & w_{12} & \ldots & w_{1h} \\ \vdots & \vdots & \vdots \ddots \vdots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{nh} \end{bmatrix} \tag{3}$$

is the matrix of first level weights and biases, b_j is the bias of hidden neuron j, and w_{ij} is the weight between the i-th input variable and the j-th hidden-layer neuron. $\mathbf{x}_k = [1, x_1(k), x_2(k), \ldots, x_n(k)]^T$ is the vector of inputs where the first element is for the bias of each hidden neuron, and $\mathbf{\Gamma}(\boldsymbol{\xi}) = [f(\xi_1), \ldots, f(\xi_h)]^T$ for $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_h]^T$, where $f(\cdot)$ represents the activation function of the neurons of the hidden layer.

Assuming the availability of N input-output data samples, usually a conventional gradient-based training algorithm is used to find the weights $\mathbf{W}$, and $\mathbf{v}$ that minimize the following cost function:

$$e = \sum_{k=1}^{N} \left(g\left(\mathbf{v}^T \mathbf{s}_k\right) - y(k)\right)^2, \tag{4}$$

where $y(k)$ is the desired output at time instant k.

Although this algorithm works well in many applications, it still has several issues, such as local minima, overfitting, slow convergence rate, etc.

In order to overcome these problems, the batch ELM was proposed in [4]. This method randomly generates the weights and bias of the first level of the SLFN and obtains the output weights using a MP generalized inverse, improving the generalization capability and decreasing both the number of parameters to be adjusted by the user and the computational costs, allowing the implementation in common automation equipment such as programmable logic controllers or microcontrollers. However, in processes with time-varying behaviors, sequential learning algorithms may be preferred over batch learning algorithms as they do an incremental learning, not requiring a complete batch retraining whenever new data is received.

III. Extreme Learning Machine

The batch ELM was proposed in [4]. In [6] it is proved that a SLFN with randomly chosen weights between the input layer and the hidden layer, and adequately chosen output weights are universal approximators for any bounded non-linear piecewise continuous function. In ELM, the input weights and bias matrix $\mathbf{W}$ is randomly assigned and, considering an output neuron with a linear activation function, the SLFN network can be regarded as a linear regression model between the output vector of the hidden layer and the output of the SLFN. Therefore, the output weights vector $\mathbf{v}$ can be estimated as:

$$\hat{\mathbf{v}} = \mathbf{S}_N^{\dagger} \mathbf{y}_N, \tag{5}$$

where $\mathbf{S}_N^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix

$$\mathbf{S}_N = [\mathbf{s}_1, \ldots, \mathbf{s}_N]^T, \tag{6}$$

and $\mathbf{y}_N = [y(1), \ldots, y(N)]^T$ is the vector of the target outputs.

Considering that $\mathbf{S}_N \in \mathbb{R}^{N \times h}$ with $N \geq h$ and $rank\left(\mathbf{S}_N\right) = h$ the Moore-Penrose generalized inverse of $\mathbf{S}_N$ can be given by:

$$\mathbf{S}_N^{\dagger} = \left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1} \mathbf{S}_N^T. \tag{7}$$

Substituting (7) into (5), the estimate $\hat{\mathbf{v}}$ of $\mathbf{v}$ can be obtained by the following least-squares solution:

$$\hat{\mathbf{v}} = \left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1} \mathbf{S}_N^T \mathbf{y}_N. \tag{8}$$

The sequential implementation of the ELM results in the application of recursive least-squares (RLS) to estimate the output weights vector $\hat{\mathbf{v}}$ [1].

Despite of the fast ELM training time, and regardless of using either batch or sequential/recursive ELM implementation, the solution obtained by least-squares may be not the most robust solution. The columns of the hidden-layer output matrix $\mathbf{S}_N$ (output of each hidden node) can have strong multicollinearities due to large number of hidden nodes or due to redundancy in the input variables, which can result in an ill-conditioned term $\left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1}$ and in an unstable least-squares solution [7]. In order to circumvent or mitigate the ill-condition of the hidden-layer output matrix, the partial least-squares (PLS) can be used to obtain the output weights.

IV. On-line Sequential ELM Based on Recursive Partial Least-Squares

If matrix $\left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1}$ is well-conditioned, the best estimate of $\mathbf{v}$ in a least squares sense is given by (8). However, if $\left(\mathbf{S}_N^T \mathbf{S}_N\right)^{-1}$ is ill-conditioned, an alternative to the Moore-Penrose generalized inverse should be used. Using a PLS method, the data are projected into a space spanned by a

Algorithm 1 Traditional batch-wise PLS algorithm with N available data samples.

Inputs: Hidden layer output matrix, $\mathbf{S}_N$; the vector of target outputs, $\mathbf{y}_N$; and the number of latent variables l;

1) Set $\mathbf{E}_0 = \mathbf{S}_N$, $\mathbf{f}_0 = \mathbf{y}_N$, and $k = 0$;
2) Let $k \leftarrow k + 1$ and $\mathbf{u}_k \leftarrow \mathbf{f}_{k-1}$;
3) Compute the latent scores and the loading factors:
   a) $\mathbf{i}_k = \mathbf{E}_{k-1}^T \mathbf{u}_k / \mathbf{u}_k^T \mathbf{u}_k$;
   b) $\mathbf{t}_k = \mathbf{E}_{k-1}\mathbf{i}_k / \|\mathbf{E}_{k-1}\mathbf{i}_k\|$;
   c) $q_k = \mathbf{f}_{k-1}^T \mathbf{t}_k / \|\mathbf{f}_{k-1}^T \mathbf{t}_k\|$;
   d) $\mathbf{u}_k = \mathbf{f}_{k-1} q_k$;
   e) $\mathbf{p}_k = \mathbf{E}_{k-1}^T \mathbf{t}_k$;
4) Compute the coefficient b_k:
   a) $b_k \leftarrow \mathbf{u}_k^T \mathbf{t}_k$;
5) Compute the residuals $\mathbf{E}_k$ and $\mathbf{f}_k$:
   a) $\mathbf{E}_k = \mathbf{E}_{k-1} - \mathbf{t}_k \mathbf{p}_k^T$,
   b) $\mathbf{f}_k = \mathbf{f}_{k-1} - b_k \mathbf{t}_k q_k$;
6) Repeat Steps 2) to 6) until all l principal factors are calculated.

number of latent variables (factors), and $\mathbf{S}_N$ and $\mathbf{y}_N$ are then represented by:

$$\mathbf{S}_N = \mathbf{T}\mathbf{P}^T + \mathbf{E}, \tag{9}$$

$$\mathbf{y}_N = \mathbf{U}\mathbf{q}^T + \mathbf{f}, \tag{10}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_l] \in \mathbb{R}^{N \times l}$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_l] \in \mathbb{R}^{N \times l}$ are the latent score matrices, and $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_l] \in \mathbb{R}^{h \times l}$ and $\mathbf{q} = [q_1, \ldots, q_l] \in \mathbb{R}^{1 \times l}$ are the loading matrices. $\mathbf{E} \in \mathbb{R}^{N \times h}$ and $\mathbf{f} \in \mathbb{R}^{N \times 1}$ are the input and output data residuals, and l is the number of latent variables used in the model.

The data matrices $\mathbf{S}_N$ and $\mathbf{y}_N$ can also be iteratively decomposed as follows. First, let:

$$\mathbf{S}_N = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{E}_1, \tag{11}$$

$$\mathbf{y}_N = \mathbf{u}_1 q_1 + \mathbf{f}_1, \tag{12}$$

where $\mathbf{t}_1$ and $\mathbf{u}_1$ are the first column of the latent score matrices $\mathbf{T}$ and $\mathbf{U}$, and $\mathbf{p}_1$ and q_1 are the first collumn of the loading matrices $\mathbf{P}$ and $\mathbf{q}$. $\mathbf{E}_1$ and $\mathbf{f}_1$ are the input and output data residuals in the first iteration. The latent score vectors are related by a linear inner model:

$$\mathbf{u}_1 = b_1 \mathbf{t}_1 + \mathbf{r}_1, \tag{13}$$

where b_1 is a coefficient which is determined by minimizing the residual $\mathbf{r}_1$. After going through the first latent score vectors calculation, the second vectors are calculated by decomposing the residuals $\mathbf{E}_1$ and $\mathbf{f}_1$ as follows:

$$\mathbf{E}_2 = \mathbf{E}_1 - \mathbf{t}_1 \mathbf{p}_1^T, \tag{14}$$

$$\mathbf{f}_2 = \mathbf{f}_1 - b_1 \mathbf{t}_1 q_1, \tag{15}$$

being the $\mathbf{t}_1$ score vector orthogonal to $\mathbf{E}_1$ and $\mathbf{f}_1$. This procedure is repeated until all the $\mathbf{T}$, $\mathbf{P}$, $\mathbf{U}$, $\mathbf{q}$ matrices are calculated. The overall PLS algorithm is summarized in Algorithm 1 [7]. This algorithm is derived with the assumption that the data $\mathbf{S}_N$ and $\mathbf{y}_N$ are scaled to zero mean and unit variance. However, if the levels of the signals at the outputs of the neurons are comparable, the scaling may be unnecessary [8].

Using this algorithm, the matrices $\mathbf{T}$, $\mathbf{P}$, $\mathbf{q}$, $\mathbf{I}$, and $\mathbf{B}$ can be constructed, where:

$$\mathbf{I} = [\mathbf{i}_1, \ldots, \mathbf{i}_l], \tag{16}$$

$$\mathbf{B} = diag\{b_1, \ldots, b_l\}. \tag{17}$$

Consider the following Lemma [7]:

Lemma 1: If $rank(\mathbf{S}_N) = l$ and $l \leq (h + 1)$, then

$$\mathbf{E}_l = \mathbf{E}_{l+1} = \ldots = \mathbf{E}_{h+1} = 0. \tag{18}$$

In matrix form, using Lemma 1 and equations (11)-(13), $\mathbf{S}_N$ and $\mathbf{y}_N$ can be decomposed as:

$$\mathbf{S}_N = \mathbf{T}\mathbf{P}^T + \mathbf{E}_l = \mathbf{T}\mathbf{P}^T, \tag{19}$$

$$\mathbf{y}_N = \mathbf{T}\mathbf{B}\mathbf{q}^T + \mathbf{f}_l. \tag{20}$$

Consider the following Lemma [7], which shows that $\mathbf{f}_l$ is orthogonal to the latent score vector $\mathbf{t}_l$:

Lemma 2: The output residual $\mathbf{f}_i$ is orthogonal to the previous latent score vector $\mathbf{t}_j$, i.e.

$$\mathbf{t}_j^T \mathbf{f}_i = 0, \text{ for } i \geq j. \tag{21}$$

As the latent score vectors are orthogonal and have unit length (Algorithm 1, Step 3b), all the columns of $\mathbf{T}$ are mutually orthonormal. So, the following relation can be derived using (19), (20) and Lemma 2,

$$\mathbf{S}_N^T \mathbf{S}_N = \mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{P}^T = \mathbf{P}\mathbf{P}^T, \tag{22}$$

$$\mathbf{S}_N^T \mathbf{y}_N = \mathbf{P}\mathbf{T}^T\mathbf{T}\mathbf{B}\mathbf{q}^T + \mathbf{P}\mathbf{T}^T\mathbf{f}_l = \mathbf{P}\mathbf{B}\mathbf{q}^T. \tag{23}$$

In order to minimize the square residuals $\|\mathbf{y}_N - \mathbf{S}_N\mathbf{v}\|^2$, using (22)-(23) the LS solution (8) can be transformed into the following PLS solution:

$$\hat{\mathbf{v}} = (\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{P}\mathbf{B}\mathbf{q}^T. \tag{24}$$

This solution is designed for the offline case (batch learning). However, when dealing with time-varying environments, and when the samples are delivered sequentially over the time, the solution is achieved by merging the old model, represented by matrices $\mathbf{P}, \mathbf{B}$, and $\mathbf{q}$, with the new sample. In the recursive PLS (RPLS), at each time instant k, the data is represented by:

$$\mathbf{S}_k^{PLS} = \begin{bmatrix} \lambda \mathbf{P}^T \\ \mathbf{s}_k^T \end{bmatrix}; \quad \mathbf{y}_k^{PLS} = \begin{bmatrix} \lambda \mathbf{B}\mathbf{q}^T \\ y(k) \end{bmatrix}, \tag{25}$$

where λ is the forgetting factor, which has a role similar to the role it has in the LS estimator. Then, $\mathbf{S}_k^{PLS}$ and $\mathbf{y}_k^{PLS}$ can be applied as inputs $\mathbf{S}_N$, $\mathbf{y}_N$ in Algorithm 1 to find the new PLS parameters.

The method proposed in this paper is based on the use of the recursive partial least-squares method to estimate the output weights of a SLFN (considering an output neuron with a linear activation function) and is called OS-ELM-RPLS. In OS-ELM-RPLS, to select the number of RPLS latent variables, a leave-one-out validation methodology is used: in each time instant k, before updating the output weights vector, the number of latent variables is selected based on the performance of the SLFN using the samples $\mathbf{S}_{k-1}^{PLS}$ and output weights vector $\hat{\mathbf{v}}_{k-1}$ obtained until time instant $(k-1)$. The proposed method is summarized in Algorithm 2.

Algorithm 2 On-line sequential ELM based on RPLS with adaptive number of latent variables.

1) Collect an initial training set $[\mathbf{X}_{N_0}, \mathbf{y}_{N_0}]$ for the learning algorithm, where $\mathbf{X}_{N_0} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N_0}]$. The number of samples N_0 can be equal to the number of neurons of the hidden layer h. However, in order to obtain a good initial prediction, a larger number of initial samples can be considered;
2) Assign an arbitrary value to the input weights and bias matrix $\mathbf{W}$;
3) Calculate the matrix of the outputs of the neurons of hidden layer $\mathbf{S}_{N_0}$ (6) using (2);
4) With Algorithm 1, obtain the matrices $\mathbf{T}$, $\mathbf{P}$, $\mathbf{q}$, $\mathbf{I}$, and $\mathbf{B}$, using the number of latent variables $l = rank(\mathbf{S}_{N_0})$;
5) Compute the initial output weights estimation $\hat{\mathbf{v}}_0$ using (24);
6) For each newly available data sample $[\mathbf{x}_k, y(k)]$, at instant k do:
   a) To prepare the selection of the number of latent variables, for $j = 1, \ldots, rank(\mathbf{S}_{k-1}^{PLS})$, using the previous matrix $\mathbf{S}_{k-1}^{PLS}$ and vector $\mathbf{y}_{k-1}^{PLS}$, do:
      i) Compute the matrices $\mathbf{T}$, $\mathbf{P}$, $\mathbf{q}$, $\mathbf{I}$, and $\mathbf{B}$, using Algorithm 1 with $l = j$ latent variables;
      ii) Cumpute the output weights estimation $\hat{\mathbf{v}}_{k-1}$ using (24);
      iii) Compute the estimated output $\hat{y}(k-1)$ using (1);
      iv) Obtain the error $e_j = \hat{y}(k-1) - y(k-1)$ between the estimated and real outputs;
   b) Select the number of latent variables as $l = \underset{j=1,\ldots,rank(\mathbf{S}_{k-1})}{\arg\min} (e_j)$;
   c) Obtain $\mathbf{S}_k^{PLS}$ and $\mathbf{y}_k^{PLS}$ using (2) and (25);
   d) Compute the matrices $\mathbf{T}$, $\mathbf{P}$, $\mathbf{q}$, $\mathbf{I}$, and $\mathbf{B}$ using Algorithm 1;
   e) Compute the output weights estimation $\hat{\mathbf{v}}_k$ using (24);
   f) Compute the estimated output $\hat{y}(k)$ using (1).

V. Generalized Predictive Control using OS-ELM-RPLS

The model learned by the OS-ELM-RPLS can be used to construct an adaptive model-based predictive controller.

Assuming that a deterministic model of the process can be constructed, the output of the model is given by:

$$\hat{y}(k) = \xi\left(\phi(k)\right), \tag{26}$$

with a regression vector

$$\phi(k) = [y(k-1), \ldots, y(k-n_a), u(k-d), \ldots, u(k-d-n_b)]^T, \tag{27}$$

being that $u(k)$ is the control signal at instant k, n_a and n_b indicate the number of output and input delays, respectively, and d is system delay.

At time instant $k = \tau$, an approximated linearized model can be obtained by [9]:

$$\hat{y}^l(k) = \psi(\tau) - \sum_{i=1}^{n_a} a_i y(k-i) + \sum_{i=o}^{n_b} b_i u(k-d-i), \tag{28}$$

where a_i and b_i are the linear model coefficients of the output and the input,

$$a_i = \left.\frac{\partial \xi\left(\phi(k)\right)}{\partial y(k-i)}\right|_{\phi(k)=\phi(\tau)}, \tag{29}$$

$$b_i = \left.\frac{\partial \xi\left(\phi(k)\right)}{\partial u(k-d-i)}\right|_{\phi(k)=\phi(\tau)}, \tag{30}$$

and $\psi(\tau)$ is a bias term given by:

$$\begin{aligned}\psi(\tau) = y(\tau) + a_1 y(\tau-1) + \ldots + a_{n_a} y(\tau-n_a) \\ -b_0 u(\tau-d) - \ldots - b_{n_b} u(\tau-d-n_b).\end{aligned} \tag{31}$$

The main advantage of the use of the instantaneous linearization of the neural network model is not having to use a controller design based on a nonlinear model. Such type of controller can have several problems related with the computational load of the required iterative minimization methods and the trapping in local minimum of the criterion [9]. Therefore, the proposed methodology used the instantaneous linearization with the generalized predictive control (GPC) methodology.

The GPC is one of the mostly used model predictive control (MPC) methods. The GPC method was proposed by Clarke [10] and has become one of the most popular MPC methods both in industry and academia. The basic idea of GPC is to calculate a sequence of future control signals in such a way that it minimizes a multistage cost function defined over a prediction horizon. So, in each iteration the objective is minimize a criterion of the following type:

$$J = \sum_{j=0}^{N_p} (y(k+j) - y^*(k+j))^2 + q(z^{-1}) \sum_{j=0}^{N_u-1} \Delta u(k+j)^2, \tag{32}$$

where $y^*(k)$ is the reference signal, q is a weighting factor for penalizing variations in the control input, $\Delta = 1 - z^{-1}$, and N_p and N_u denote the output and control horizons, respectively. In order to reduce the computational cost, $N_u = 1$ is chosen.

The solution for $\Delta u(k)$ can be obtained using the derivative of J with respect to the control inputs increment $\Delta u(k)$. Letting

$$\frac{\partial J}{\partial(\Delta(u(k)))} = 0, \tag{33}$$

and using the method in [11], $\Delta u(k)$ can by expressed by:

$$\Delta u(k) = (\mathbf{G}^T\mathbf{G} + q_0)^{-1}\mathbf{G}^T(\mathbf{\Phi}\mathbf{R} - \mathbf{F}y(k)), \tag{34}$$

where the matrices $\mathbf{G}$, $\mathbf{\Phi}$, $\mathbf{R}$, and $\mathbf{F}$ are defined in [11], and $q_0 > 0$.

In this control scheme, the proposed identification method OS-ELM-RPLS is used to online identify the model of the process and, in each iteration, the parameters of the linear model are obtained using (29) and (30), and the variation of the control signal is obtained using (34).

VI. Results

In this section simulation results are presented to demonstrate the feasibility, performance and effectiveness of the proposed neural network modeling technique in identification and in control. All the simulation experiments have been made in the Matlab environment running on a PC with 2.00 [GHz] CPU with 2 cores and 4GB RAM.

TABLE I
DATA SETS DESCRIPTION: TRAIN/TEST INDICATES THE NUMBER OF EXEMPLARS IN THE CORRESPONDING DATA SET; ARCHITECTURE INDICATES THE NUMBER OF INPUT, HIDDEN, AND OUTPUT NODES USED IN THE SLFN; λ INDICATES THE FORGETTING FACTOR USED IN THE RPLS AND IN THE RLS.

Data set	Train/Test	Architecture	λ
Debutanizer	599/1795	$7-15-1$	0.98
Polymerization	194/453	$12-20-1$	0.98

TABLE II
VARIABLES OF THE DEBUTANIZER DATA SET.

Variables	Description
x_1	Top temperature
x_2	Top pressure
x_3	Reflux flow
x_4	Flow to next process
x_5	6^{th} tray temperature
x_6	Bottom temperature
x_7	Bottom temperature
y	Butane (C4) concentration

A. Identification Results

This section presents experimental results in two real-world data sets. Table I describes the data sets, and parameters used in the experiments. For comparison purposes, the proposed OS-ELM-RPLS method is compared with (i) OS-ELM based on RLS proposed in [1] (OS-ELM-RLS), and (ii) RPLS method proposed in [7]. The OS-ELM-RPLSM is a modified version of OS-ELM-RPLS method, where the number of RPLS latent variables l is not adapted online. Thus, OS-ELM-RPLSM does not perform Steps 6a and 6b of Algorithm 2. In the OS-ELM-RPLSM and RPLS methods, the number of latent variables used was determined by a 10-fold cross-validation procedure applied on the training data set.

The data was sequentially divided into training set and testing set. All the input and output variables have been normalized to zero mean and unit variance. The train data was used to initialize/train the estimators and the approximation performance was evaluated on the test data. SLFNs with sigmoidal activation functions in the hidden-layer neurons, and a linear activation function in the output neuron were used. The optimal number of hidden neurons h and the forgetting factor λ used in all data sets were determined by means of experimentation. At each time instant k, the estimation of the output is performed before the estimator adaptation. The approximation performance of the estimators was evaluated using the mean square error (MSE) between the predicted and desired outputs.

1) Debutanizer Process: The first case of study consists in the prediction of butane (C4) concentration in the bottom flow of a debutanizer column. This case study was introduced in [12] and an associated data set is available for download in the book website. The data set of plant variables that is available for learning consists of 7 input variables, $\mathbf{x}_k = [x_1(k), \ldots, x_7(k)]^T$, and one target output variable to be estimated, $y(k)$. The variables correspond to temperatures, pressures, flows, and the output concentration. See Table II for further details.

From experiments made in the training data set it was concluded that the best number of latent variables in OS-ELM-RPLSM and RPLS is 1. The results of application of all methods are presented in Table III and Figure 1. The times that are shown in the table refer to the time taken by all methods to predict y and perform the SLFN model adaptation for all the samples of the test set. Figure 2 shows the number of latent variables in the test set chosen by the proposed method for all the samples.

TABLE III
PERFORMANCE RESULTS OF THE FOUR METHODS IN THE TEST DATA SET OF THE DEBUTANIZER PROCESS.

Method	MSE	Time [s]
OS-ELM-RPLS	**0.146**	13.68
OS-ELM-RPLSM	0.599	13.33
OS-ELM-RLS	0.677	7.51
RPLS	0.334	0.29

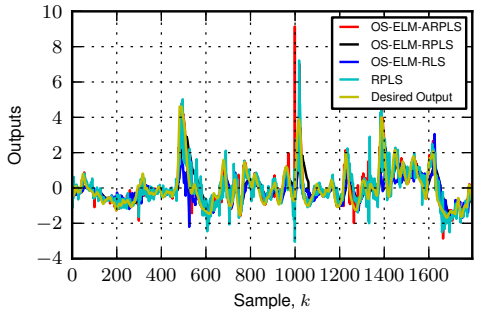

Fig. 1. Predicted and desired outputs using the four methods in the debutanizer data set.

Analyzing the results, it can be verified that the proposed OS-ELM-RPLS method is the method with the best prediction performance over the four methods. It can also be seen that the performance of the RPLS, which generates a linear model, is better than the performance of the OS-ELM-RLS and OS-ELM-RPLSM. Comparing the results of the OS-ELM-RPLS and the OS-ELM-RPLSM, it can be verified the importance of the adaptation of the number of latent variables performed by OS-ELM-RPLS during the estimation. With respect to the computational time, the RPLS is the fastest method. However, the time taken by OS-ELM-RPLS in each iteration is approximately $13.68/1795 \approx 8$ [milliseconds], which is a good time for real applications.

2) Polymerization Process: The polymerization data set is a benchmark for adaptive soft sensors introduced in [3], [13]. This data set describes a polymerization reactor and the objective is the prediction of the catalyst activity in the multitube. The data set covers 1 year of acquisition with 8687 available samples and is composed of 15 input variables.

This paper follows the same pre-processing procedure as was done in [13]: downsampling of the first 5800 samples by a factor of 10 to restrict the available information in the training set, the removal of variables 3, 4, 15, and removing all samples which have missing values. The preprocessing results into a data set with 194 training data samples and 453 test samples.

The results of application of all methods are presented in Table IV and Figure 3. From experiments made in the training data set is was concluded that the best number of latent variables in OS-ELM-RPLSM and RPLS is 3. Figure 4

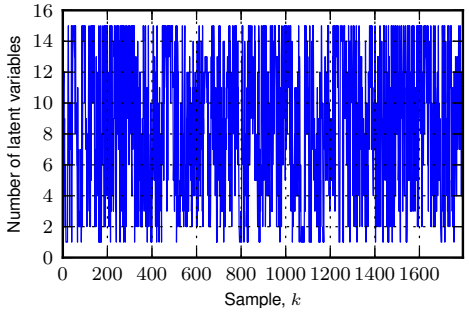

Fig. 2. Number of latent variables used by the OS-ELM-RPLS in the debutanizer test set.

TABLE IV
PERFORMANCE RESULTS OF THE FOUR METHODS IN THE TEST DATA SET OF THE POLYMERIZATION PROCESS.

Method	MSE	Time [s]
OS-ELM-RPLS	**0.031**	4.67
OS-ELM-RPLSM	0.056	4.25
OS-ELM-RLS	0.110	2.78
RPLS	0.047	0.07

shows the number of latent variables in the test data set chosen by the proposed OS-ELM-RPLS method. The obtained results are similar to the results of Experiment I: as in the previous experiment, the proposed method has the best prediction performance in test data set, followed by the RPLS. Once again, it is shown the importance of the online adaptation of the number of the latent variables.

B. Control Results

This case of study consists in the control of the fluid flow through a valve. The Wiener model that describes the valve is described in [14]. In this model, $u(k)$ is the pneumatic control signal applied to the stem, $j(k)$ is the stem position, and $y(k)$ is the flow through the valve which is the controlled variable. The model can be described as follows:

$$j(k) = \frac{0.0616z^{-1} + 0.0543z^{-2}}{1 - 1.5714z^{-1} + 0.6873z^{-2}}u(k), \quad (35)$$

$$y(k) = \frac{j(k)}{\sqrt{0.1 + 0.9j(k)^2}} + v(k), \quad (36)$$

where $v(k)$ is an external disturbance.

In the proposed control scheme, it was used a dataset for the initialization of the model. The dataset was constructed by injecting a control signal $u(k) = (sin(x(k))+1)/2$ for $x(k) = -\pi + 0.01k$ and $k = 1, \ldots, M$, where $M = \lfloor 2\pi/0.01 \rfloor$, and $\lfloor \alpha \rfloor$ is the greatest integer smaller than or equal to α.

In order to test the control system, the following reference signal is used:

$$y^*(k) = \begin{cases} 0.5, & \text{if } 0 < k < 149, \\ 1, & \text{if } 150 \leq k \leq 400, \end{cases} \quad (37)$$

and the applied load disturbances $v(k)$ are: $v(k) = 0.2$ for $k = 300$, and $v(k) = -0.2$ for $k = 350$. In the experiments, the sampling period adopted is 0.1s and the experiment duration time is 4 [s].

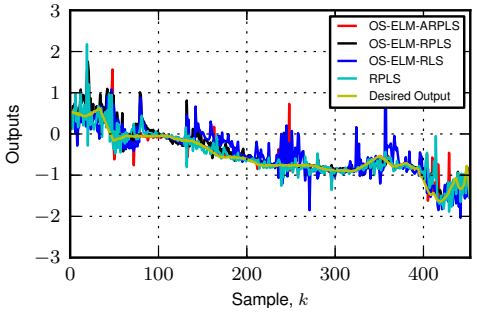

Fig. 3. Predicted and desired outputs using the four methods in the polymerization data set.

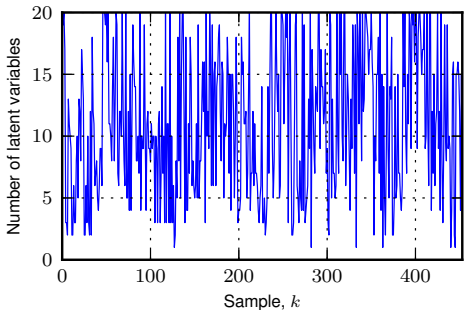

Fig. 4. Number of latent variables used by the OS-ELM-RPLS in the polymerization test set.

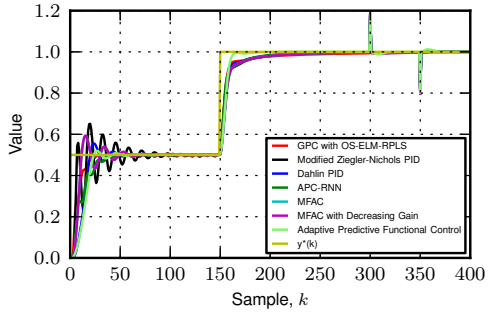

Fig. 5. Tracking performances of the seven controllers applied in the valve control.

The presented control scheme was compared with six self-tuning control approaches: PID controller based on the modified Ziegler-Nichols criterion [15], Dahlin PID controller [15], adaptive predictive control with recurrent neural network (APC-RNN) [16], Model-free Adaptive Controller (MFAC) [17], MFAC with decreasing gain [18], and adaptive predictive functional controller [19].

The tracking performances and the control signals of these seven self-tuning controllers are shown in Figures 5 and 6, respectively. The root mean square error for the output tracking measurement is shown is Table V.

From the experimental results it can be seen that all the control methodologies have a good tracking performance. The approach where the root mean square of the tracking error is

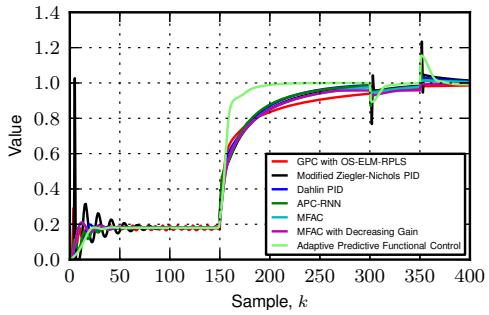

Fig. 6. Control signals of the seven controllers applied in the valve control.

TABLE V
ROOT MEAN SQUARE OF THE TRACKING ERROR IN THE VALVE CONTROL.

Controller	e_{RMS}
GPC using OS-ELM-RPLS	**0.0750**
Modified Ziegler-Nichols PID	0.0808
Dahlin PID	0.0972
Adaptive predictive control with RNN	0.0823
Model-free adaptive control	0.0838
MFAC with decreasing gain	0.0838
Adaptive predictive functional control	0.0978

smaller is the control scheme where the proposed OS-ELM-RPLS is used with the GPC controller.

VII. Conclusion

A novel learning algorithm for SLFNs called on-line sequential extreme learning machine based on recursive partial least-squares with adaptation of the number of latent variables (OL-ELM-RPLS) was presented. The proposed OL-ELM-RPLS method is an improvement of the OS-ELM-RLS method proposed in [1] where RLS is used to update the estimation of the output weights. Due to the multicollinearities that can exist in the hidden-layer output matrix columns caused by the excessive number of hidden neurons or redundant input variables, the estimation of the output weights by LS can result in an ill-conditioned problem and therefore in an unstable solution. Therefore, in the proposed methodology the RLS method was replaced by a RPLS method. Furthermore, an online adaptation of the number of RPLS latent variables was introduced in the proposed algorithm.

The identification methodology was proposed for two application problems: (1) system identification, and (2) the learning of a model for the generalized predictive control (GPC) algorithm. In the identification, the performance of the proposed method was better than the performance of OS-ELM-RPLSM, OS-ELM-RLS, and RPLS in both data sets. The results also show that the adaptation of the number of latent variables leads to an improvement of the prediction performance of the proposed method and that the time taken in each iteration of the method allows its online implementation in real-world applications. The results obtained in the control of a valve system, show that the control scheme using the proposed OS-ELM-RPLS identification methodology used jointly with the GPC controller was the method with best tracking performance.

Acknowledgment

This work was supported by Project SCIAD "Self-Learning Industrial Control Systems Through Process Data" (reference: SCIAD/2011/21531) co-financed by QREN, in the framework of the "Mais Centro - Regional Operational Program of the Centro", and by the European Union through the European Regional Development Fund (ERDF).

The authors acknowledge the support of FCT project PEst-C/EEI/UI0048/2014. Francisco Souza, Saeid Rastegar, and Jérôme Mendes have been supported by FCT under grants SFRH/BD/63454/2009, SFRH/BD/89186/2012, and SFRH/BD/63383/2009, respectively.

References

[1] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan, "On-line sequential extreme learning machine," in *IASTED International Conference on Computational Intelligence*, M. H. Hamza, Ed. IASTED/ACTA Press, July 2005, pp. 232–237.

[2] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "Op-elm: Optimally pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, 2010.

[3] 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.

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[5] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[6] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, July 2006.

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

[8] K. Helland, H. E. Berntsen, O. S. 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.

[9] J. Chen and T.-C. Huang, "Applying neural networks to on-line updated PID controllers for nonlinear process control," *Journal of Process Control*, vol. 14, no. 2, pp. 211–230, 2004.

[10] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control-part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[11] E. F. Camacho and C. Bordons, *Model Predictive Control*, ser. Advanced Textbooks in Control and Signal Processing Series. London: Springer-Verlag, July 2007.

[12] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer, 2007.

[13] P. Kadlec and B. Gabrys, "Local learning-based adaptive soft sensor for catalyst activation prediction," *AIChE Journal*, vol. 57, no. 5, pp. 1288–1301, 2011.

[14] H. Al-Duwaish and W. Naeem, "Nonlinear model predictive control of hammerstein and wiener models using genetic algorithms," in *Proceedings of the 2001 IEEE International Conference on Control Applications*, 2001, pp. 465–469.

[15] V. Bobál, J. Böhm, J. Fessl, and J. Machácek, *Digital Self-tuning Controllers: Algorithms, Implementation and Applications*. London, UK: Springer-Verlag, 2005.

[16] C.-H. Lu and C.-C. Tsai, "Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 3, pp. 1366–1375, March 2008.

[17] Z. Hou and W. Huang, "The model-free learning adaptive control of a class of siso nonlinear systems," in *Proceedings of American Control Conference*, vol. 1, Jun. 1997, pp. 343 –344 vol.1.

[18] X. Bu, Z. Hou, F. Yu, and F. Wang, "Robust model free adaptive control with measurement disturbance," *IET Control Theory Applications*, vol. 6, no. 9, pp. 1288 –1296, 14 2012.

[19] B. Zhang and W. Zhang, "Adaptive predictive functional control of a class of nonlinear systems," *ISA Transactions*, vol. 45, no. 2, pp. 175–183, 2006.