Nonlinear System Identification using a New Sliding-Window Kernel RLS Algorithm

Steven Van Vaerenbergh, Javier Vía and Ignacio Santamaría
Dept. of Communications Engineering, University of Cantabria, Spain
E-mail: {steven,jvia,nacho}@gtas.dicom.unican.es

***Abstract*— In this paper we discuss in detail a recently proposed kernel-based version of the recursive least-squares (RLS) algorithm for fast adaptive nonlinear filtering. Unlike other previous approaches, the studied method combines a sliding-window approach (to fix the dimensions of the kernel matrix) with conventional ridge regression (to improve generalization). The resulting kernel RLS algorithm is applied to several nonlinear system identification problems. Experiments show that the proposed algorithm is able to operate in a time-varying environment and to adjust to abrupt changes in either the linear filter or the nonlinearity.**

***Index Terms*— kernel methods, kernel RLS, sliding-window, system identification**

I. Introduction

An important number of kernel methods have been proposed in recent years, including support vector machines [1], kernel principal component analysis [2], kernel Fisher discriminant analysis [3] and kernel canonical correlation analysis [4], [5], with applications in classification and nonlinear regression problems. In their original forms, most of these algorithms cannot be used to operate online since a number of difficulties are introduced by the kernel methods, such as the time and memory complexities (because of the growing kernel matrix) and the need to avoid overfitting.

Recently a kernel RLS algorithm was proposed that dealt with both difficulties [6]: by applying a sparsification procedure the kernel matrix size was limited and the order of the problem was reduced. This kernel RLS algorithm allows for online training but cannot handle time-varying data. Other approaches to reduce the order of the kernel matrix have been also been proposed, including a recent kernel principal component analysis (PCA) algorithm [6] which is able to track time-varying data.

In this paper we discuss a different approach, applying conventional ridge regression instead of reducing the order of the feature space. A sliding-window technique is applied to fix the size of the kernel matrix in advance, allowing the algorithm to operate online in time-varying environments.

After describing the proposed sliding-window kernel RLS algorithm, we discuss its application to different nonlinear system identification problems. First, it is applied to identify a Wiener system, which is a simple nonlinear model that typically appears in satellite communications [7] or digital magnetic recording systems [8]. Then we apply the kernel RLS algorithm to a nonlinear regression problem. Apart from its applications in system identification, the proposed sliding-window kernel RLS algorithm can also be used as a building block for other, more complex adaptive algorithms. In particular, it has been applied to develop a sliding-window kernel canonical correlation analysis (CCA) algorithm [9].

The rest of this paper organized as follows: in Section II the least-squares criterion is explained briefly. Kernel methods are introduced in Section III and a detailed description of the sliding-window algorithm is given in Section IV. In Section V it is applied to two nonlinear system identification problems and Section VI summarizes the main conclusions of this work.

II. Regularized Least-Squares

A. Least-Squares Problem

The least-squares (LS) criterion [10] is a widely used method in signal processing. Given a vector $\mathbf{y} \in \mathbb{R}^{N\times 1}$ and a data matrix $\mathbf{X} \in \mathbb{R}^{N\times M}$ of N observations, it consists in seeking the optimal vector $\mathbf{h} \in \mathbb{R}^{M\times 1}$ that solves

$$J = \min_{\mathbf{h}} \|\mathbf{y} - \mathbf{X}\mathbf{h}\|^2. \tag{1}$$

When the number of observations N is at least equal to the number of unknowns M, the least-squares problem is *overdetermined* and has either one unique solution or an infinite number of solutions. If $\mathbf{X}^T\mathbf{X}$ is a non-singular matrix, the unique solution of the least-squares problem is given by

$$\hat{\mathbf{h}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2}$$

If $\mathbf{X}$ is rank deficient, $\mathbf{X}^T\mathbf{X}$ is a singular matrix corresponding to a LS problem with infinitely many solutions.

For full rank matrices $\mathbf{X}$ with $N \geq M$, the solution $\hat{\mathbf{h}}$ can be represented in the basis defined by the rows of $\mathbf{X}$. Hence it can also be written as $\mathbf{h} = \mathbf{X}^T\mathbf{a}$, making it a linear combination of the input patterns (this is sometimes denoted as the "dual representation").

This paper is based on "A Sliding-Window Kernel RLS Algorithm and its Application to Nonlinear Channel Identification," by S. Van Vaerenbergh, J. Vía, and I. Santamaría, which appeared in the Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 2006. © 2006 IEEE.

This work was supported by MEC (Ministerio de Educación y Ciencia) under grant TEC2004-06451-C05-02 and FPU grant AP2005-5366.

To improve generalization and increase the smoothness of the solution, a regularization term is often included in the standard least-squares formulation (1):

$$J = \min_{\mathbf{h}} \left[\|\mathbf{y} - \mathbf{X}\mathbf{h}\|^2 + c\mathbf{h}^T\mathbf{h} \right], \quad (3)$$

where c is a regularization constant. This problem is known as regularized least-squares, and its solution is given by

$$\hat{\mathbf{h}} = \left[\mathbf{X}^T\mathbf{X} + c\mathbf{I} \right]^{-1} \mathbf{X}^T\mathbf{y}, \quad (4)$$

where $\mathbf{I}$ is the identity matrix.

B. Recursive Least-Squares

In many problems not all data are known in advance and the solution has to be re-calculated as new observations become available. In case of linear problems, the well-known recursive least-squares (RLS) algorithm [10] can be used, which calculates the solution of the regularized LS problem (3) in a recursive manner, based on the Sherman-Morrison-Woodbury formula for matrix inversion [11].

The most commonly used form of RLS is the *exponentially weighted* RLS algorithm, which gives more weight to recent data and less weight to previous data. Given a positive scalar $\lambda < 1$ (the *forgetting factor*), this algorithm recursively calculates the solution to the problem

$$J = \min_{\mathbf{h}} \left[\sum_{j=0}^{N} \lambda^{N-j} |y(j) - \mathbf{x}_j^T\mathbf{h}\|^2 + c\mathbf{h}^T\mathbf{h} \right], \quad (5)$$

where $\mathbf{x}_j^T$ denotes the j-th row of $\mathbf{X}$, for a growing number of observations N. For a complete formulation of this algorithm, we refer to [10].

III. Kernel Methods

In recent years, the framework of reproducing kernel Hilbert spaces (RKHS) has led to the family of algorithms known as *kernel methods* [12]. These algorithms use Mercer kernels in order to produce nonlinear versions of conventional linear algorithms.

Kernel methods are based on the implicit nonlinear transformation of the data $\mathbf{x}_i$ from the input space to a high-dimensional *feature space* $\tilde{\mathbf{x}}_i = \Phi(\mathbf{x}_i)$. The key property of kernel methods is that scalar products in the feature space can be seen as nonlinear (kernel) functions of the data in the input space. Since this property avoids the explicit mapping to the feature space, it allows to perform any conventional scalar product based algorithm in the feature space by solely replacing the scalar products with the Mercer kernel function in the input space.

In this way, any positive definite kernel function satisfying Mercer's condition [1]: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ has an implicit mapping to some higher-dimensional feature space. This simple and elegant idea is known as the "kernel trick", and it is commonly applied by using a nonlinear kernel such as the Gaussian kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right),$$

which implies an infinite-dimensional feature space, or the polynomial kernel of order p

$$\kappa(\mathbf{x}_i, \mathbf{y}_j) = \left(1 + \mathbf{x}_i^T\mathbf{x}_j\right)^p.$$

A. Kernel LS

A nonlinear "kernel" version of the linear LS algorithm can be obtained by transforming the data into feature space. Using the transformed vector $\tilde{\mathbf{h}} \in \mathbb{R}^{M' \times 1}$ and the transformed data matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times M'}$, the LS problem (1) can be written in feature space as

$$J' = \min_{\tilde{\mathbf{h}}} \|\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{h}}\|^2. \quad (6)$$

The transformed solution $\tilde{\mathbf{h}}$ can now also be represented in the basis defined by the rows of the (transformed) data matrix $\tilde{\mathbf{X}}$, namely as

$$\tilde{\mathbf{h}} = \tilde{\mathbf{X}}^T\boldsymbol{\alpha}. \quad (7)$$

Moreover, introducing the *kernel matrix* $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ the LS problem in feature space (6) can be rewritten as

$$J' = \min_{\boldsymbol{\alpha}} \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 \quad (8)$$

in which the solution $\boldsymbol{\alpha}$ is an $N \times 1$ vector. The advantage of writing the nonlinear LS problem in the dual notation is that thanks to the "kernel trick", we only need to compute $\mathbf{K}$, which is done as

$$\mathbf{K}(i,j) = \kappa(\mathbf{x}_i, \mathbf{x}_j), \quad (9)$$

where $\mathbf{x}_i^T$ and $\mathbf{x}_j^T$ are the i-th and j-th rows of $\mathbf{X}$. As a consequence, the computational complexity of operating in this high-dimensional space is not necessarily larger than that of working in the original low-dimensional space.

B. Measures Against Overfitting

For most useful kernel functions, the dimension of the feature space, M', will be much higher than the number of available data points N (for instance, in case the Gaussian kernel is used, the feature space will have dimension $M' = \infty$). As mentioned in Section II, Eq. (8) could have an infinite number of solutions in these cases, due to the rank deficiency of $\tilde{\mathbf{X}}$. A number of techniques to handle this overfitting have been presented.

1) Kernel RLS with sequential sparsification: In [6] a sparsification process was proposed in which an input sample is only admitted into the kernel if its image in feature space cannot be sufficiently well approximated by combining the previously admitted samples. In this way the resulting kernel RLS algorithm reduces the order of the feature space [4], [5]. A related kernel LS algorithm was recently presented in [13].

2) Kernel RLS by kernel PCA: Another approach to reduce the order of the feature space is by applying principal component analysis (PCA) [14]. PCA is a technique based on singular value decomposition (SVD) to extract the dominant eigenvectors of a matrix.

3) Kernel ridge regression: A third method to regularize the solution of the kernel RLS problem is by applying conventional ridge regression. This type of regularization was chosen for the proposed algorithm because of its low computational complexity. In (kernel) ridge regression, the norm of the solution $\tilde{\mathbf{h}}$ is penalized as in (3) to obtain the following problem:

$$J'' = \min_{\tilde{\mathbf{h}}} \left[\|\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{h}}\|^2 + c\tilde{\mathbf{h}}^T\tilde{\mathbf{h}} \right]. \tag{10}$$

This problem can be written in dual notation as follows:

$$J'' = \min_{\boldsymbol{\alpha}} \left[\|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + c\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} \right] \tag{11}$$

whose solution is given by

$$\boldsymbol{\alpha} = \mathbf{K}_{reg}^{-1}\mathbf{y} \tag{12}$$

with $\mathbf{K}_{reg} = (\mathbf{K} + c\mathbf{I})$ and c is a regularization constant.

IV. The Online Algorithm

In a number of situations it is preferred to have an online, i.e. recursive, algorithm. In particular, if the data points $\mathbf{y}$ are the result of a time-varying process, an online algorithm can be designed, able to track these time variations. The key feature of an online algorithm is that the number of computations required per new sample must not increase as the number of samples increases.

A. A Sliding-Window Approach

In [15] we presented a regularized kernel version of the RLS algorithm, using a sliding-window approach. An online setup assumes we are given a stream of input-output pairs $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots\}$. The sliding-window approach considers only a window consisting of the last N pairs of this stream (see Fig. 1). For the n-th window, the observation vector $\mathbf{y}_n = [y_n, y_{n-1}, \dots, y_{n-N+1}]^T$ and observation matrix $\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, \dots, \mathbf{x}_{n-N+1}]^T$ are formed, and the corresponding regularized kernel matrix $\mathbf{K}_n = \tilde{\mathbf{X}}_n\tilde{\mathbf{X}}_n^T + c\mathbf{I}$ can be calculated.

Note that it is necessary to limit the number of data vectors $\mathbf{x}_n$, N, from which the kernel matrix is calculated. Unlike standard linear RLS, for which the correlation matrix has fixed size depending on the (fixed) *dimension of the input vectors* M, the size of the kernel matrix in an online scenario depends on the *number of observations* N.

In [6], a kernel RLS algorithm is designed that limits the matrix sizes by means of a sparsification procedure, which maps the samples to a (limited) dictionary. It allows both to reduce the order of the feature space (which prevents overfitting) and to keep the complexity of the algorithm bounded. In our approach these two measures are obtained by two different mechanisms. On one hand, the regularization against overfitting is done by penalizing the solutions, as in (12). On the other hand, the complexity of the algorithm is reduced by considering only the observations in a window with fixed length. The advantage of the latter approach is that it is able to track time variations without extra computational burden.

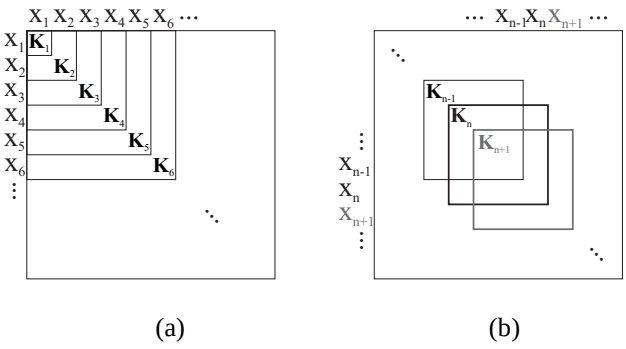

Figure 1. (a) Kernel matrices $\mathbf{K}_i$ of growing size. (b) Kernel matrices $\mathbf{K}_i$ of fixed size, obtained by only considering the data in windows of fixed size.

B. Updating the Inverse of the Kernel Matrix

The calculation of the updated solution $\boldsymbol{\alpha}_n$ requires the calculation of the $N \times N$ inverse matrix $\mathbf{K}_n^{-1}$ for each window. This is costly both computationally (requiring $O(N^3)$ operations) and memory-wise. Therefore an update algorithm is developed that can compute $\mathbf{K}_n^{-1}$ solely from knowledge of the data of the current window $\{\mathbf{X}_n, \mathbf{y}_n\}$ and the previous $\mathbf{K}_{n-1}^{-1}$. The updated solution $\boldsymbol{\alpha}_n$ can then be calculated in a straightforward way using Eq. (12).

Given the regularized kernel matrix $\mathbf{K}_{n-1}$, the new regularized kernel matrix $\mathbf{K}_n$ can be constructed in two steps. First, the first row and column of $\mathbf{K}_{n-1}$ are removed. This step is referred to as *downsizing* the kernel matrix, and the resulting matrix is denoted as $\hat{\mathbf{K}}_{n-1}$. In the second step, kernels of the new data are added as the last row and column to this matrix:

$$\mathbf{K}_n = \begin{bmatrix} \hat{\mathbf{K}}_{n-1} & \mathbf{k}_{n-1}(\mathbf{x}_n) \\ \mathbf{k}_{n-1}(\mathbf{x}_n)^T & k_{nn} + c \end{bmatrix}, \tag{13}$$

where $\mathbf{k}_{n-1}(\mathbf{x}_n) = [\kappa(\mathbf{x}_{n-N+1}, \mathbf{x}_n), \dots, \kappa(\mathbf{x}_{n-1}, \mathbf{x}_n)]^T$ and $k_{nn} = \kappa(\mathbf{x}_n, \mathbf{x}_n)$. This step is referred to as *upsizing* the kernel matrix.

Calculating the inverse kernel matrix $\mathbf{K}_n^{-1}$ is also done in two steps, using the two inversion formulas derived in appendices A and B. First, given the previous kernel matrix $\mathbf{K}_{n-1}$ and its inverse $\mathbf{K}_{n-1}^{-1}$, the inverse of the downsized $N-1 \times N-1$ matrix $\hat{\mathbf{K}}_{n-1}$ is calculated according to Eq. (15). Then, the matrix $\hat{\mathbf{K}}_{n-1}$ is upsized to obtain $\mathbf{K}_n$, and based on the knowledge of $\mathbf{K}_n$ and $\hat{\mathbf{K}}_{n-1}^{-1}$ the inversion formula from Eq. 16 is applied to obtain $\mathbf{K}_n^{-1}$.

Note that these formulas do not calculate the inverse matrices explicitly, but rather derive them from known matrices maintaining an overall time complexity of $O(N^2)$ of the algorithm, where N is the window length.

To initialize the algorithm, zero-padding of the data was applied to fill the sliding window. The initial observation matrix $\mathbf{X}_0$ consists entirely of zeros. The regularized kernel matrix $\mathbf{K}_0$ corresponding to these data is obtained and its inverse is calculated. For the Gaussian and polynomial kernel function, this matrix is $\mathbf{K}_0 = \mathbf{1} + c\mathbf{I}$, where $\mathbf{1}$ is

Algorithm 1 Summary of the proposed adaptive algorithm.

Initialize $\mathbf{K}_0$ and calculate $\mathbf{K}_0^{-1}$.
for $n = 1, 2, \ldots$ **do**
  Downsizing: Obtain $\hat{\mathbf{K}}_{n-1}$ out of $\mathbf{K}_{n-1}$.
  Calculate $\hat{\mathbf{K}}_{n-1}^{-1}$ according to Eq. (15).
  Upsizing: Obtain $\mathbf{K}_n$ according to Eq. (13).
  Calculate $\mathbf{K}_n^{-1}$ according to Eq. (16).
  Obtain the updated solution $\boldsymbol{\alpha}_n = \mathbf{K}_n^{-1}\mathbf{y}_n$.
end for

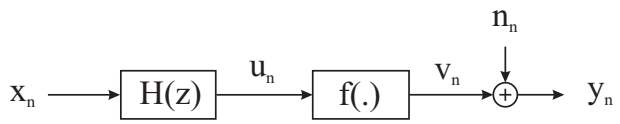

Figure 2. A nonlinear Wiener system.

an $N \times N$ matrix filled by ones and $\mathbf{I}$ is the unit matrix. The complete algorithm is summarized in Alg. (1).

V. Experiments

In this section we investigate the performance of the sliding-window kernel RLS algorithm to identify two nonlinear systems. In both cases the results are compared to online identification by a multi-layer perceptron (MLP), which is a standard approach for identifying nonlinear systems. Since kernel methods provide a natural nonlinear extension of linear regression methods [12], the proposed system is supposed to perform well compared to the MLP.

A. Identification of a Wiener System with an abrupt channel change

The Wiener system is a well-known and simple nonlinear system which consists of a series connection of a linear filter and a memoryless nonlinearity (see Fig. 2). Such a nonlinear channel can be encountered in digital satellite communications and in digital magnetic recording. Traditionally, the problem of blind nonlinear equalization or identification has been tackled by considering nonlinear structures such as MLPs [16], recurrent neural networks [17], or piecewise linear networks [18].

We consider a supervised identification problem, in which moreover at a given time instant the linear channel coefficients are changed abruptly to compare the tracking capabilities of both algorithms: During the first part of the simulation, the linear channel is $H_1(z) = 1 - 0.3668z^{-1} - 0.4764^{-2} + 0.8070^{-3}$ and after receiving 500 symbols it is changed into $H_2(z) = 1 - 0.8326z^{-1} + 0.6656z^{-2} + 0.7153^{-3}$. A binary signal ($x_n \in \{-1, +1\}$) is sent through this channel after which the signal is transformed nonlinearly according to the nonlinear function $v = \tanh(u)$, where v is the linear channel output. A scatter plot of u_n versus v_n can be found in Fig. 3. Finally, 20dB of additive white Gaussian noise (AWGN) is added. The Wiener system is treated as a black box of which only input and output are known. To fill the initial sliding-windows, zero-padding of the signals was applied. The MLP was trained in an online manner, i.e. in every iteration one new data point was used for updating the net weights.

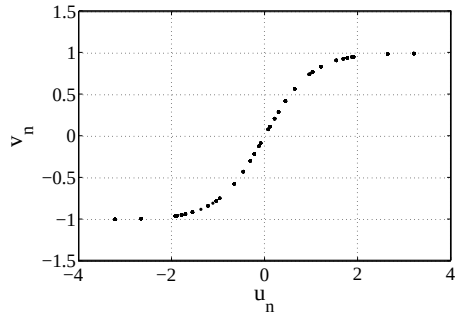

Figure 3. Signal in the middle of the Wiener system vs. output signal for binary input symbols and different indices n.

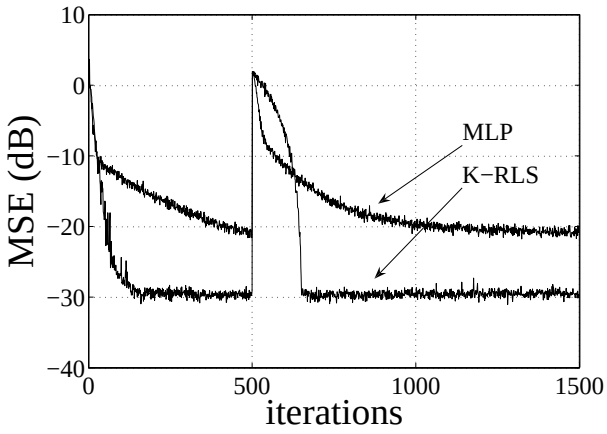

Figure 4. MSE of the identification of the nonlinear Wiener system of Fig. 2, for the standard method using an MLP and for the window-based kernel RLS algorithm with window length $N = 150$. A change in filter coefficients of the nonlinear Wiener system was introduced after sending 500 data points.

1) Comparison to MLP: System identification was first performed by an MLP with 8 neurons in its hidden layer and learning rate 0.1, and then by using the sliding-window kernel RLS with window size $N = 150$ and using a polynomial kernel of order $p = 3$. For both methods we applied time-embedding techniques in which the length L of the linear channel was known. More specifically, the used MLP was a time-delay MLP with L inputs, and the input vectors for the kernel RLS algorithm were time-delayed vectors of length L, $\mathbf{x}_n = [x_{n-L+1}, \ldots, x_n]^T$. The length of the used linear channels H_1 and H_2 was $L = 4$.

In iteration n the input-output pair $(\mathbf{x}_n, y_n)$ is fed into the identification algorithm. The performance is evaluated by estimating the next output sample $\hat{v}_{n+1}$, given the next input vector x_{n+1}, and comparing it to the actual output v_{n+1}. For both methods, the mean square error (MSE) was averaged out over 250 Monte-Carlo simulations.

The MSE for both approaches is shown in Fig. 4. In iteration 500 to 650 it can be seen that the algorithm needs

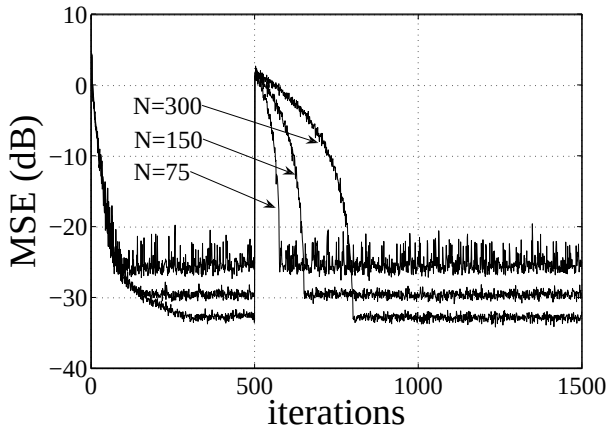

Figure 5. Comparison of the identification results of the kernel RLS algorithm for different window lengths for Wiener system identification. Larger windows obtain a better MSE. In general, the number of iterations needed for convergence is of the order of the window length.

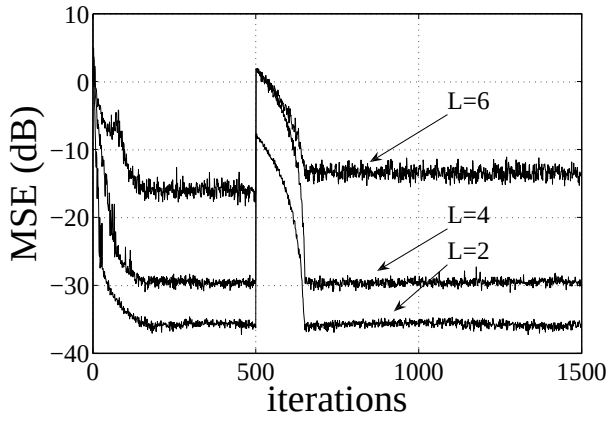

Figure 6. Comparison of the identification results of the kernel RLS algorithm for different Wiener system channel lengths. Since the same sliding-window length was used in the three situations, best results are obtained for the shortest channel. Note that the channel length was known while performing identification.

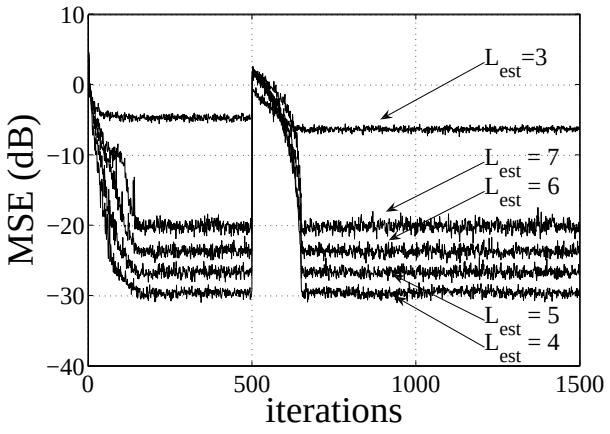

Figure 7. Identification results when the correct channel length $L = 4$ is not known. The curve $L_{est} = 3$ shows that underestimation of the channel leads to a worse performance than overestimation, which corresponds to $L_{est} > 4$.

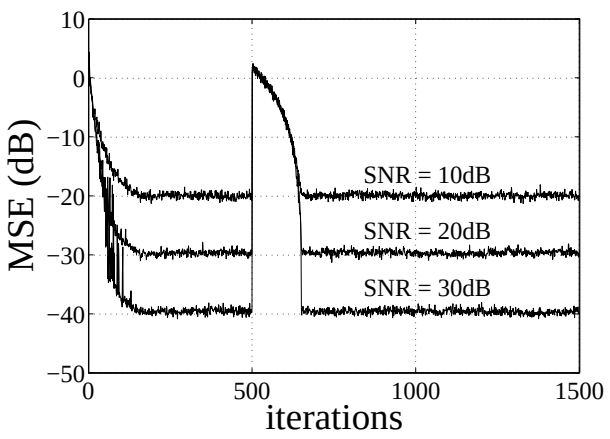

Figure 8. Identification results of the kernel RLS algorithm for different system SNR values in Wiener system identification.

N iterations to adjust completely to the channel change. For the chosen learning rate, the MSE of the MLP initially drops fast but it converges to a higher error.

In the following, the influence of some parameters is commented, compared to the basic setup with $N = 150$, $L = 4$ and 20dB SNR.

2) Influence of parameters:

a) Window length: Fig. 5 shows the performance of the kernel RLS algorithm for different sliding-window lengths N. First, it is observed that a larger window corresponds to slower convergence of the algorithm after the channel change. This occurs because N iterations are needed to replace the data in the kernel matrix with data corresponding to the new channel. Second, for small windows, peaks appear in the MSE. The kernel method generally combines the images of the data points within one window to represent the output of the nonlinear system. For small windows (such as $N = 75$) the data points within one window are often insufficient to model the complexity of the nonlinear system, thus increasing the MSE variance. This also explains why the algorithm converges to a higher MSE for larger windows.

b) Channel length: If a longer channel is used in a Wiener system, more information needs to be represented by the identification algorithm. If the same sliding-window length $N = 150$ is used in identification of Wiener systems with different channel lengths, performance drops for longer channels, as can be seen in Fig. 6. In general, the performance can be maintained for such channels if the window length is increased adequately.

If the correct channel length L is not known, an estimate L_{est} can be made. The simulations of Fig. 7 showed that the algorithm is very sensitive to underestimations of the channel length ($L_{est} < L$). In any case it is preferred to overestimate the channel length slightly ($L_{est} > L$). Note that computation time is hardly affected by an increase of the estimated channel length, since only the kernel function takes into account the channel length.

c) Noise level: Fig. 8 shows the MSE curves for different SNR values for the white Gaussian input noise, reflecting clearly the variance of the input noise.

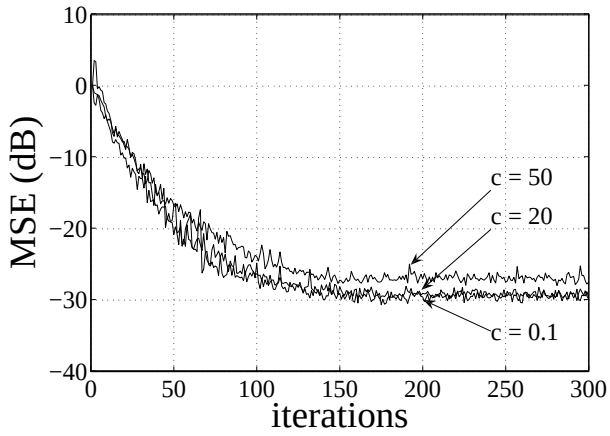

Figure 9. Influence of the regularization constant on the algorithm's performance. For $c \leq 20$ the obtained performance is very similar. For larger values, such as $c = 50$, regularization has a dominating effect on the performance.

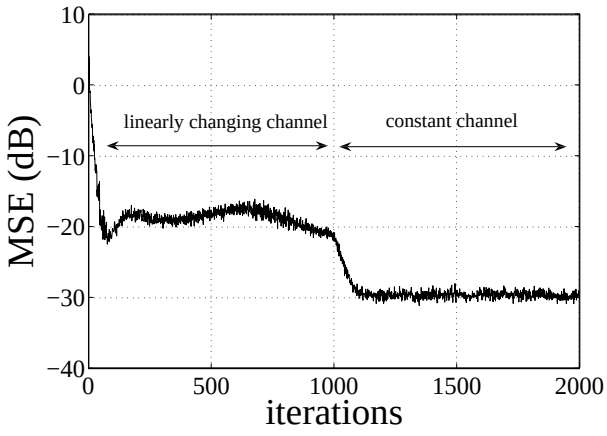

Figure 10. MSE for time-tracking of a Wiener system in which the channel varies linearly.

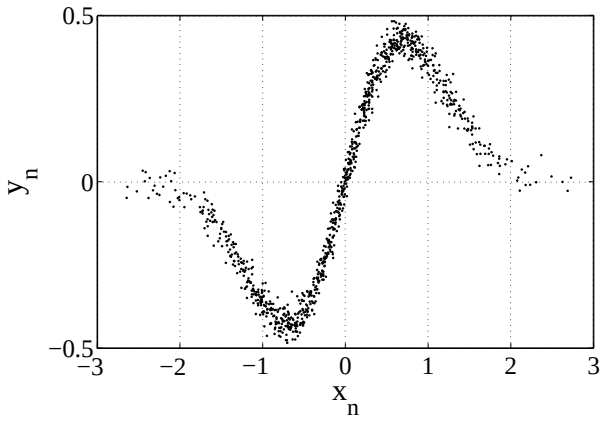

Figure 11. Input x_n versus output y_n of the system to identify, for different indices n. This system consists of a single nonlinearity to which noise is added at the output.

d) Regularization constant: The influence of the regularization constant on the algorithm performance is small, as can be seen in Fig. 9. For very large values ($c > 20$) the "smoothing" effect of the regularization constant on the solution can be noticed.

e) Kernel function: Given the polynomial shape of the observed nonlinearity, all previous tests were performed using a polynomial kernel function. The obtained MSE values are better than if for instance the Gaussian kernel function is used.

Finally, note that the nonlinear system has been treated as a black-box model. If the structure of the system is known, this information can be exploited to perform identification. For this purpose, the presented sliding-window kernel RLS algorithm can be extended and used as the basis of a more complex algorithm.

In [9] we proposed an online kernel canonical correlation analysis (CCA) algorithm by merging the sliding-window kernel RLS algorithm and a robust adaptive CCA algorithm. By applying a linear kernel on the input data and a nonlinear kernel on the output data, the CCA framework allows to treat the Wiener filter identification as two coupled LS problems, resulting in efficient identification and equalization. For details on this method we refer to [9].

B. Identification of a time-varying Wiener System

In *linear* systems that are varying in time, the RLS algorithm with forgetting factor (5) can be applied to identify the system and track the system changes. Although a similar exponentially weighted version of the kernel RLS algorithm for nonlinear system identification is considered a future research line, by using a sliding-window the presented algorithm is also capable of tracking time variations.

Fig. 10 shows the MSE curve for a Wiener system in which the channel varies linearly over the first 1000 iterations from $H_1(z)$ to $H_2(z)$ and is static over the next 1000 iterations. When the channel is varying in time, the sliding-window kernel RLS algorithm obtains a slightly higher MSE than for the static channel.

C. Nonlinear regression

In this section we perform system identification on a system consisting of a single nonlinearity. This system is equivalent to a Wiener system with a linear channel of length $L = 1$. For the experiment a nonlinear system corresponding to the equation

$$y_n = x_n \exp\left(-x_n^2\right) + n_n \tag{14}$$

was chosen where the input signal x_n was zero-mean white gaussian noise of unit variance, and n_n represents 20dB AWGN. A scatter plot of u_n versus v_n is shown in Fig. 11.

1) Comparison to MLP: As in the previous examples, the sliding-window kernel RLS method was used to identify the nonlinear system. The results are compared to the MSE for identification by an MLP, as shown in Fig. 12. After comparing results for a wide range of parameter values, a length of $N = 50$ was chosen for the sliding-window, and for the MLP 5 neurons in the hidden layer

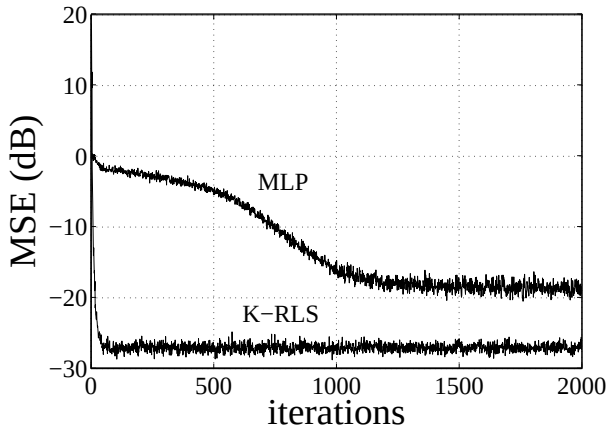

Figure 12. The performance of the kernel RLS algorithm and an MLP for nonlinear regression.

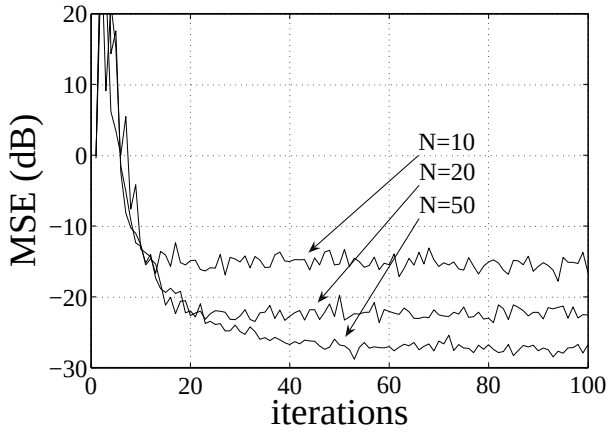

Figure 13. Comparison of the MSE results of the kernel RLS algorithm for nonlinearity identification with different window lengths.

and the learning rate 0.1 were used. These values were chosen. The Gaussian kernel was applied in the sliding-window algorithm.

In general a small window length is sufficient to identify the nonlinearity system, as can be seen in Fig. 13. The lowest MSE level was already obtained with a window of length $N = 50$.

VI. Conclusions

A detailed study of a recently proposed sliding-window-based RLS algorithm was presented. The main features of this algorithm are the introduction of regularization against overfitting (by penalizing the solutions) and the combination of a sliding-window approach and efficient matrix inversion formulas to keep the complexity of the problem bounded. Thanks to the use of a sliding-window the algorithm is able to provide tracking in a time-varying environment.

The results of this algorithm suggest it can be extended to deal with the nonlinear extensions of most problems that are classically solved by linear RLS. Future research lines also include an exponentially weighted version of this kernel RLS algorithm and a comparison to other new kernel methods.

References

[1] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York, USA: Springer-Verlag New York, Inc., 1995.

[2] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. NNSP'99*, Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. IEEE, Jan. 1999, pp. 41–48.

[4] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," *Journal of Machine Learning Research*, vol. 3, pp. 1–48, 2003.

[5] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis: An overview with application to learning methods," Royal Holloway University of London, Technical Report CSD-TR-03-02, 2003.

[6] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares-algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, Aug. 2004.

[7] G. Kechriotis, E. Zarvas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. on Neural Networks*, vol. 5, pp. 267–278, Mar 1994.

[8] N. P. Sands and J. M. Cioffi, "Nonlinear channel models for digital magnetic recording," *IEEE Trans. Magn.*, vol. 29, pp. 3996–3998, Nov 1993.

[9] S. Van Vaerenbergh, J. Vía, and I. Santamaría, "Online kernel canonical correlation analysis for supervised equalization of wiener systems," in *Proc. of the 2006 IJCNN International Joint Conference on Neural Networks (IJCNN)*, Vancouver, Canada, July 2006.

[10] A. Sayed, *Fundamentals of Adaptive Filtering*. New York, USA: Wiley, 2003.

[11] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The John Hopkins University Press, 1996.

[12] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: The MIT Press, 2002.

[13] E. Andelić, M. Schafföner, M. Katz, S. Krüger, and A. Wendemuth, "Kernel least-squares models using updates of the pseudoinverse," *Neural Computation*, vol. 18, no. 12, pp. 2928–2935, 2006.

[14] L. Hoegaerts, L. Lathauwer, I. Goethals, J. Suykens, J. Vandewalle, and B. Moor, "Efficiently updating and tracking the dominant kernel principal components." *Neural Networks*, vol. 20, no. 2, pp. 220–229, mar 2007.

[15] S. Van Vaerenbergh, J. Vía, and I. Santamaría, "A sliding window kernel rls algorithm and its application to nonlinear channel identification," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toulouse, France, May 2006.

[16] D. Erdogmus, D. Rende, J. C. Principe, and T. F. Wong, "Nonlinear channel equalization using multilayer perceptrons with information-theoric criterion," in *Proc. IEEE Workshop on Neural Networks and Signal Processing XI*, Falmouth, MA, Sept 2001, pp. 401–451.

[17] T. Adali and X. Liu, "Canonical piecewise linear network for nonlinear filtering and its application to blind equalization," *Signal Process.*, vol. 61, no. 2, pp. 145–155, Sept 1997.

[18] P. W. Holland and R. E. Welch, "Robust regresison using iterative reweighted least squares," *Commun. Statist. Theory Methods*, vol. A. 6, no. 9, pp. 813–827, 1997.

Steven Van Vaerenbergh received the degree of Electrical Engineer with specialization in Telecommunications from Ghent University, Belgium in 2003. Since 2003 he has been pursuing the Ph. D. degree at the Communications Engineering Department, University of Cantabria, Spain, under the supervision of I. Santamaría. His current research interests include kernel methods and their applications to identification, equalization and separation of signals.

Javier Vía received the degree of Telecommunication Engineer from the University of Cantabria, Spain in 2002. Since 2002 he has been pursuing the Ph. D. degree at the Communications Engineering Department, University of Cantabria, under the supervision of I. Santamaría. His current research interests include neural networks, kernel methods, and their application to digital communication problems.

Ignacio Santamaría received his Telecommunication Engineer Degree and his Ph.D. in Electrical Engineering from the Polytechnic University of Madrid, Spain in 1991 and 1995, respectively. In 1992 he joined the Department of Communications Engineering, University of Cantabria, Spain, where he is currently Professor. In 2000 and 2004, he spent visiting periods at the Computational NeuroEngineering Laboratory (CNEL), University of Florida.

Dr. Santamaría has more than 90 publications in refereed journals and international conference papers. His current research interests include nonlinear modeling techniques, machine learning theory and their application to digital communication systems, in particular to inverse problems (deconvolution, equalization and identification problems).

APPENDIX
MATRIX INVERSION FORMULAS

A. The inverse of a downsized matrix

By *downsizing* a matrix $\mathbf{K}$, we denote the operation of removing its first row and column. In the formulas below, downsizing $\mathbf{K}$ results in the matrix $\mathbf{D}$. The inverse matrix $\mathbf{D}^{-1}$ can easily be expressed in terms of the known elements of $\mathbf{K}^{-1}$ as follows:

$$\mathbf{K} = \begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{b} & \mathbf{D} \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} e & \mathbf{f}^T \\ \mathbf{f} & \mathbf{G} \end{bmatrix}$$

$$\Rightarrow \begin{cases} \mathbf{b}e + \mathbf{D}\mathbf{f} &= \mathbf{0} \\ \mathbf{b}\mathbf{f}^T + \mathbf{D}\mathbf{G} &= \mathbf{I} \end{cases}$$

$$\Rightarrow \mathbf{D}^{-1} = \mathbf{G} - \mathbf{f}\mathbf{f}^T/e. \tag{15}$$

B. The inverse of an upsized matrix

By *upsizing* a matrix we refer to adding one row and one column at the end. The matrix $\mathbf{A}$ shown below is upsized to the matrix $\mathbf{K}$. Given the inverse matrix $\mathbf{A}^{-1}$ and upsized matrix $\mathbf{K}$, the inverse matrix $\mathbf{K}^{-1}$ can be obtained as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & d \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{E} & \mathbf{f} \\ \mathbf{f}^T & g \end{bmatrix}$$

$$\Rightarrow \begin{cases} \mathbf{A}\mathbf{E} + \mathbf{b}\mathbf{f}^T &= \mathbf{I} \\ \mathbf{A}\mathbf{f} + \mathbf{b}g &= \mathbf{0} \\ \mathbf{b}^T\mathbf{f} + dg &= 1 \end{cases}$$

$$\Rightarrow \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{I} + \mathbf{b}\mathbf{b}^T\mathbf{A}^{-1H}g) & -\mathbf{A}^{-1}\mathbf{b}g \\ -(\mathbf{A}^{-1}\mathbf{b})^T g & g \end{bmatrix} \tag{16}$$

with $g = (d - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b})^{-1}$.