LOW COMPLEXITY APPROXIMATE CYCLIC ADAPTIVE MATCHING PURSUIT

Alexandru Onose, Bogdan Dumitrescu

Department of Signal Processing
Tampere University of Technology
PO BOX 553, 33101, Tampere, Finland
e-mail: alexandru.onose@tut.fi, bogdan.dumitrescu@tut.fi

ABSTRACT

Based on the iterated cyclic adaptive matching pursuit algorithm, we construct a low complexity approximate variant for finding sparse solutions to systems of linear equations. We employ a greedy neighbor permutation strategy coupled with an approximate scalar product matrix to ensure that the complexity of the algorithm remains low. The sparse solution is cyclically updated improving the performance while the sparsity level is estimated online using the predictive least squares criterion. The performance of the algorithm is similar to that of the non approximate variants while the complexity can be considerably lower.

Index Terms— matching pursuit, sparse filters, greedy algorithm, channel identification

1. INTRODUCTION

Sparse approximation problems have arisen in recent years from many practical application like compression, denoising or array processing. Several algorithms [1, 2, 3] have been proposed with applications to typical signal and image processing tasks. We aim to present in this paper a low complexity version of the iterated cyclic adaptive matching pursuit (ICAMP) [4] algorithm that employs a series of approximations and adaptation steps to provide a lower complexity without negatively influencing the estimation error.

Consider a typical FIR channel identification problem where, at time t, the input $u(t)$ and the output $d(t)$ are measured. The goal is to estimate the true coefficients h_j that minimize the estimation error

$$e(t) = d(t) - \sum_{j=0}^{N-1} h_j u(t-j), \tag{1}$$

where N is the filter length.

Work supported by Tekes FiDiPro – Finland Distinguished Professor Programme and by GETA – Graduate School in Electronics, Telecommunications and Automation. B. Dumitrescu is also with Department of Automatic Control and Computers, "Politehnica" University of Bucharest, Romania.

Minimizing the estimation error at each time instance can be transformed towards minimizing a least squares criterion

$$J(t) = \sum_{i=1}^{t} \lambda^{t-i} |e(i)|^2, \tag{2}$$

where $0 < \lambda \leq 1$ is a forgetting factor. Written in matrix form, the criterion is equivalent to minimizing the norm of the residual $||\mathbf{b} - \mathbf{A}\mathbf{x}||^2$; the solution vector $\mathbf{x}$ corresponds to the estimated coefficients h_j that result from the minimization of (2). We consider the solution $\mathbf{x}$ to be sparse, henceforth it has at most $M \ll N$ non zero coefficients. Moreover, the number and the positions of the coefficients are a priori unknown. The matrix $\mathbf{A} \in \mathbb{R}^{t \times N}$ is built with the input data, its i-th row being equal to $\lambda^{\frac{t-i}{2}} \boldsymbol{\alpha}^{(i)T}$ where

$$\boldsymbol{\alpha}^{(i)} = \left[u(i), u(i-1), \ldots, u(i-N+1)\right]^T. \tag{3}$$

The vector $\mathbf{b} \in \mathbb{R}^t$ contains the weighted output data $b_i = \lambda^{\frac{t-i}{2}} d(i)$. We denote hereafter $\beta^{(i)} = d(i)$ and, for simplicity, we drop the index (i) from $\boldsymbol{\alpha}^{(i)}$ and $\beta^{(i)}$ when they refer to the current available data at time t.

We develop an approximate variant of the ICAMP algorithm by selectively storing the information about the input data matrix $\mathbf{A}$. The algorithm updates and uses only the scalar products between the full matrix $\mathbf{A}$ and a set of columns from $\mathbf{A}$ that are associated with the current solution support. When a decision to modify the support is made, we delay the actual change by introducing a constant length buffer to temper any rapid support variation. The scalar products associated with the buffer are also updated and can contribute directly to changes in the support. We propose two algorithms that use a fixed respectively a variable upper sparsity bound and estimate the sparsity level with the predictive least squares (PLS) criterion.

The content of this paper is as follows: in section 2 we present the proposed low complexity approximate algorithms; a brief overview of the ICAMP algorithm and the PLS criterion in conjunction with the approximate algorithms is included in section 3; section 4 contains the results of our simulations proving the performance of the algorithms.

2. LOW COMPLEXITY APPROXIMATE ALGORITHMS

We begin by briefly reviewing the classic matching pursuit (MP) algorithm [5] and its adaptive counterpart, the adaptive matching pursuit (AMP) [6]. Starting from AMP, we introduce a series of approximations that provide lower complexity which, used in conjunction with the iterated cyclic adaptive matching pursuit algorithm [4] coupled with a buffer that mitigates fast changes in the solution support, form the basis of our proposed low complexity iterated adaptive matching pursuit algorithm.

The algorithm follows the MP column selection strategy selecting one by one columns (named active columns) from the matrix $\mathbf{A}$ based on their alignment with the residual. The selection starts by finding the best column $\mathbf{a}_{k_1}$ aligned with output data vector representing the first residual $\mathbf{b}_0 = \mathbf{b}$ (the solution $\mathbf{x}$ is null); the column $\mathbf{a}_{k_1}$ is included in the active column set $\mathcal{A}$. The projection of the current residual $\mathbf{b}_0$ on $\mathbf{a}_{k_1}$ is then removed from $\mathbf{b}_0$, producing the new residual $\mathbf{b}_1$ used in the selection of the next active column. After $i-1$ columns are selected, the search for the column $\mathbf{a}_{k_i}$, best aligned with the current residual $\mathbf{b}_{i-1}$, continues in the remaining column set $\mathcal{I}$ composed by columns not yet included in $\mathcal{A}$,

$$k_i = \arg\max_{l \in \mathcal{I}} \frac{|\mathbf{a}_l^T \mathbf{b}_{i-1}|^2}{||\mathbf{a}_l||^2}. \tag{4}$$

The new residual is computed by projecting $\mathbf{b}_{i-1}$ on $\mathbf{a}_{k_i}$,

$$\mathbf{b}_i = \mathbf{b}_{i-1} - x_{k_i} \mathbf{a}_{k_i}, \tag{5}$$

where the coefficient x_{k_i} represents the alignment of the column $\mathbf{a}_{k_i}$ with the residual $\mathbf{b}_{i-1}$,

$$x_{k_i} = \frac{\mathbf{a}_{k_i}^T \mathbf{b}_{i-1}}{||\mathbf{a}_{k_i}||^2}. \tag{6}$$

The selection continues until M columns are chosen.

The above equations are implemented using only the scalar products $\mathbf{\Psi} = \mathbf{A}^T\mathbf{b}$ between the input matrix and the output vector $\mathbf{b}$ and $\mathbf{\Phi} = \mathbf{A}^T\mathbf{A}$ between the columns of the matrix $\mathbf{A}$. The norms of the columns from the matrix $\mathbf{A}$ are stored separately in $\mathbf{\Theta}$ to simplify the presentation, otherwise they are present on the diagonal of $\mathbf{\Phi}$. Equation (5) can be easily expressed using the scalar products by multiplying it with $\mathbf{A}^T$ on the left.

Additionally, a neighbor search strategy is employed for the column selection. Relying on the slow variation of the channel, we reuse the selection order found at time $t-1$ when we perform the search at time t. Thus, starting from the old active set $\mathcal{A}$ we constrain the search for each new column position j between neighbor positions j and $j+1$ in the set. The resulting permutations are performed only between neighbor positions and produce the updated active set at current time t. Changes in the composition of the active set $\mathcal{A}$ are achieved by allowing all remaining inactive columns to compete for the last position M.

At each time instance t, when new input vector $\boldsymbol{\alpha}$ and output data β are available, all the scalar products are updated in place,

$$\begin{array}{lll} \mathbf{\Phi}_{1:N,1:N} & \leftarrow & \lambda \mathbf{\Phi}_{1:N,1:N} + \boldsymbol{\alpha}_{1:N} \boldsymbol{\alpha}_{1:N}^T \\ \mathbf{\Psi}_{1:N} & \leftarrow & \lambda \mathbf{\Psi}_{1:N} + \beta \boldsymbol{\alpha}_{1:N} \\ \mathbf{\Theta}_j & \leftarrow & \lambda \mathbf{\Theta}_j + \alpha_j^2 \ \text{ with } \ j = 1:N, \end{array} \tag{7}$$

and the solution and column selection are revised.

Consider the true sparse solution having $L_t \ll N$ non zero coefficients; by choosing $M \geq L_t$ small enough the main computational burden is due to the update of the scalar products $\mathbf{\Phi}$. They are however not all necessary when computing the solution at a given time. In a stationary regime, where the active set $\mathcal{A}$ composition does not change as new samples are received, the only scalar products required are the column norms $\mathbf{\Theta}$ and the scalar products between the active columns $\mathbf{a}_{k_{1:M}}$ and the full matrix $\mathbf{A}$. Thus, in the ideal scenario where no support changes are made, storing and updating only the partial scalar product matrix $\bar{\mathbf{\Phi}} = \mathbf{A}^T \mathbf{a}_{k_{1:M}}$ instead of the whole $\mathbf{\Phi} = \mathbf{A}^T\mathbf{A}$ ensures that all the required data are available.

When an inactive column $\mathbf{a}_{k_j}$ becomes active instead of the old column $\mathbf{a}_{k_M}$, the associated coefficient x'_{k_j} can still be computed precisely like in (6) using the stored column norms $\mathbf{\Theta}$; it produces an updated residual

$$\mathbf{b}'_M = \mathbf{b}_{M-1} - x'_{k_j} \mathbf{a}_{k_j}. \tag{8}$$

Because the channel is slow varying, there are usually no sudden changes, either in the support or in the coefficient values. Thus, when a coefficient becomes inactive it does so gradually until the associated column reaches the last position $\mathbf{a}_{k_M}$ in the active set $\mathcal{A}$ and is then replaced by the new column $\mathbf{a}_{k_j}$. If $M > L_t$, then we can assume that any coefficient x_{k_i} above the sparsity level L_t ($i > L_t$) is small since the true solution contains all the relevant coefficients. When the support changes, the coefficient x'_{k_j} has a negligible influence in decreasing the residual and the scalar products associated with (8) can be approximated by

$$\mathbf{A}^T \mathbf{b}'_M \approx \mathbf{A}^T \mathbf{b}_{M-1} \tag{9}$$

This does not require any information regarding the unknown scalar products $\mathbf{A}^T\mathbf{a}_{k_j}$. When the new coefficient gradually become significant, if the unknown scalar products $\mathbf{A}^T\mathbf{a}_{k_j}$ are far from their true values, the performance is negatively influenced even leading to instability. To circumvent this we propose to set $\mathbf{A}^T\mathbf{a}_{k_j}$ to zero and allow a number of updates to be performed before the column may be introduced into the active set. For this purpose we use a buffer $\mathcal{B}$ of length P (containing columns $\mathbf{a}_{k_{M+1:M+P}}$) to delay the introduction of any new column in $\mathcal{A}$. The scalar products are updated

for all the columns associated with $\mathcal{B}$, hence diminishing the approximation errors. Furthermore, since the scalar products with the columns from the active set are computed exactly, the errors introduced by the unknown scalar products after the column is included in $\mathcal{A}$ on position i do not affect significantly the coefficients $x_{k_{1:i}}$.

Thus, if the column $\mathbf{a}_{k_j}$, selected to be introduced in the active set $\mathcal{A}$ on position M, belongs to $\mathcal{I} \setminus \mathcal{B}$, it replaces the last column $\mathbf{a}_{k_{M+P}}$ in $\mathcal{B}$ instead of being introduced directly in the active set $\mathcal{A}$. The associated scalar products are set to zero and the solution is still computed with the old column $\mathbf{a}_{k_M}$. If $\mathbf{a}_{k_j}$ belongs to $\mathcal{B}$, it is promoted one position in the set, $k_{j-1} \leftrightarrow k_j$. It becomes active replacing $\mathbf{a}_{k_M}$ in the active set, $k_M \leftrightarrow k_{M+1}$, only when it is on the first position in $\mathcal{B}$. This ensures that a certain column is selected at least P times before becoming active which, coupled with the update of their associated scalar products as new samples are received, reduces the approximation errors.

We present in Alg. 1 the algorithm that constructs the set $\mathcal{B}$ and performs the neighbor permutations. We consider that the elements in $\mathbf{x}$, the columns in the matrix $\mathbf{A}$ and all the associated scalar products are permuted according to the column selection order implicitly present in the active set $\mathcal{A}$ and in the buffer $\mathcal{B}$. Additionally, the partial matrix $\bar{\mathbf{\Phi}}$ is stored in the full matrix $\mathbf{\Phi}$ to simplify the notations. Also, note that the scalar products between the active columns (and columns from $\mathcal{B}$) and all the other columns have exact values and are never set to zero (Alg. 1, step 1.3).

The complexity due to the scalar product update $\mathbf{\Phi}$ is reduced from $\frac{3}{2}N^2$ to $3N(M+P) - \frac{3}{2}(M+P)^2$. The scalar product update complexity is decreased by a factor proportional with $\frac{1}{2}\frac{N}{M+P}$.

3. APPROXIMATE ITERATED CYCLIC ADAPTIVE MATCHING PURSUIT

Selecting the column $\mathbf{a}_{k_i}$ and computing the solution coefficient x_{k_i} like in the AMP algorithm decreases the residual by the largest amount when the previously computed coefficients, $x_{k_{1:i-1}}$, are kept fixed. To further minimize the residual, a cyclical update [7] is performed by optimizing one coefficient at a time while keeping the rest $i-1$ coefficients fixed. The coefficient x_{k_j}, $j \in \{1, \ldots, i\}$ is updated by removing the corresponding column $\mathbf{a}_{k_j}$ from the active set, hence producing the residual

$$\mathbf{b}_i^{'} = \mathbf{b}_i + x_{k_j}\mathbf{a}_{k_j}, \tag{10}$$

and then reincluding it in the active set. The updated coefficient is

$$x_{k_j}^{'} = \frac{\mathbf{a}_{k_j}^T \mathbf{b}_i^{'}}{||\mathbf{a}_{k_j}||^2} = x_{k_j} + \gamma, \tag{11}$$

Alg. 1 (**Introduce the candidate column j in $\mathcal{B}$, perform the necessary neighbor permutations**).

1 if $j > M+P$ (the column is not in buffer $\mathcal{B}$)
- 1.1 Swap columns (and rows) $M+P$ and j in $\mathbf{\Phi}$
  Swap elements $M+P$ and j in $\tilde{\mathbf{\Psi}}$, $\mathbf{x}$ and $\mathbf{\Psi}$
  The information on row j in $\mathbf{\Phi}$ is discarded
- 1.2 Set to zero the unknown scalar products for column $M+P$
  $\Phi_{M+P+1:N,M+P} = 0$
- 1.3 Keep the known scalar products
  $\Phi_{M+P,M+P} = \Theta_{M+P}$
  $\Phi_{1:M+P-1,M+P} = \Phi_{M+P,1:M+P-1}^T$

2 if $1 < j \leq M+P$ (the column is in $\mathcal{B}$ or in $\mathcal{A}$)
- 2.1 Swap columns (and rows) $j-1$ and j in $\mathbf{\Phi}$
  Swap elements $j-1$ and j in $\tilde{\mathbf{\Psi}}$, $\mathbf{x}$ and $\mathbf{\Psi}$

where $\gamma = \frac{\mathbf{a}_{k_j}^T \mathbf{b}_i}{||\mathbf{a}_{k_j}||^2}$. This produces a residual update similar to (5),

$$\begin{gathered}\mathbf{b}_i \leftarrow \mathbf{b}_i^{'} - (x_{k_j} + \gamma)\mathbf{a}_{k_j} = \mathbf{b}_i - \gamma\mathbf{a}_{k_j} \\ x_{k_j} \leftarrow x_{k_j}^{'}.\end{gathered} \tag{12}$$

The cyclic update step, performed N_{it} times for all columns considered by the solution, further minimizes the residual. All the above updates can be expressed with the use of the scalar products $\mathbf{\Psi}$ and the partial scalar products $\bar{\mathbf{\Phi}}$ such that the approximate version of the ICAMP algorithm only requires the approximation steps presented in section 2. The column selection and coefficient estimation is presented in Alg. 2 while the cyclic update is included in Alg. 3.

The true sparsity level L_t of the solution $\mathbf{x}$ can be estimated with the use of information theoretic criteria [3, 4]. If L_t is unknown, we can apply the relations presented so far to find M, $M \geq L_t$, columns $\mathbf{a}_{k_{1:M}}$ that correspond to a M-sparse solution $\mathbf{x}$. Because the selection process introduces an inherent column order, we assume that after enough data are available the first selected positions correspond with high probability to the true support of the solution. The threshold M can be either fixed or can change such that it approaches $L_t + \Delta$, where Δ is a predefined constant ensuring that there are enough candidates for estimating the sparsity level [4]. For clarity we present only the fixed version of the algorithm, the variable one being rather straightforward. Note that for the variable version, due to the changes of the threshold M, the buffer $\mathcal{B}$ needs to be adjusted such that its size is always P. Thus, if M decreases, the last column from the active set $\mathcal{A}$ will be moved to $\mathcal{B}$ while the last column in $\mathcal{B}$ is discarded. If M increases, the first column from $\mathcal{B}$ moves to $\mathcal{A}$ and on the last position in $\mathcal{B}$ a new (random) column is promoted.

The estimate L of the true sparsity level L_t is computed as the point for which the predictive least squares (PLS) criterion

Alg. 2 (**Estimate the coefficient** i).

1 if $i < M$

1.1 Update scalar product for next column
$\tilde{\Psi}_{i+1} \leftarrow \tilde{\Psi}_{i+1} - \mathbf{\Phi}_{i+1,1:i-1}\mathbf{x}_{1:i-1}$

1.2 Find best candidate column between neighbors
$k_i = \arg\max_{l\in[i,i+1]} \frac{\tilde{\Psi}_l^2}{\Theta_l}$

2 if $i == M$

2.1 Update remaining scalar products
$\tilde{\mathbf{\Psi}}_{M+1:N} \leftarrow \tilde{\mathbf{\Psi}}_{M+1:N} - \mathbf{\Phi}_{M+1:N,1:M-1}\mathbf{x}_{1:M-1}$

2.2 Find best k_i candidate column between all remaining columns
$k_i = \arg\max_{l\in[M:N]} \frac{\tilde{\Psi}_l^2}{\Theta_l}$

3 Permute necessary columns (Alg. 1)

4 $x_i = \frac{\tilde{\Psi}_i}{\Phi_{i,i}}$ (evaluate coefficient value)

5 Update scalar product considering new residual
$\tilde{\mathbf{\Psi}}_{1:i+1} \leftarrow \tilde{\mathbf{\Psi}}_{1:i+1} - x_i\mathbf{\Phi}_{1:i+1,i}$

Alg. 3 (**Cyclic update of** j **coefficients**).

1 for $i = 1 : j$ (cyclically update each coefficient)

1.1 $\gamma = \frac{\tilde{\Psi}_i}{\Theta_i}$

1.2 $x_i \leftarrow x_i + \gamma$ (update the coefficient)

1.3 $\tilde{\mathbf{\Psi}}_{1:M_i+1} \leftarrow \tilde{\mathbf{\Psi}}_{1:M_i+1} - \gamma\mathbf{\Phi}_{1:M_i+1,i}$

[8] attains its minimum. At time instance t, the PLS criterion is

$$\mathbf{PLS}_j^{(t)} = \sum_{i=0}^{t} \lambda^{t-i} {e_j^{(i)}}^2, \tag{13}$$

where $e_j^{(i)} = \beta^{(i)} - \boldsymbol{\alpha}_{k_{1:j}}^{(i)T}\check{\mathbf{x}}_{j,k_{1:j}}$; the index (i) denotes the time at which the data are considered, $\check{\mathbf{x}}_{j,k_{1:j}}^{(i)}$ contains the coefficients of the j-sparse solution. The PLS criterion can be therefore computed recursively by

$$\mathbf{PLS}_j^{(t)} = \lambda \cdot \mathbf{PLS}_j^{(t-1)} + {e_j^{(t)}}^2. \tag{14}$$

The approximate ICAMP algorithm using the PLS criterion is summarized in Alg. 4. The number of operations required by the column selection and coefficient computation is $3N(M+P) - M^2 + NM$ while the cyclic update adds $\frac{1}{3}N_{\text{it}}M^3 + 4N_{\text{it}}M^2$ operation. The computation of the PLS criterion is fast requiring only M^2 operations.

4. SIMULATIONS

The performance of the algorithms was tested using a sparse FIR channel identification problem (1) with $N = 200$ and the true solutions having $L_t = 5$ non zero coefficients. The channel non zero coefficients are described by

$$h_j(t) = a_j \cos(2\pi f T_s t + \phi_j). \tag{15}$$

Alg. 4 (**Iterated cyclic adaptive matching pursuit with PLS column selection, approximate version**).

1 Update needed scalar products between columns of $\mathbf{A}$
$\mathbf{\Phi}_{1:N,1:M+P} \leftarrow \lambda\mathbf{\Phi}_{1:N,1:M+P} + \boldsymbol{\alpha}_{1:N}\boldsymbol{\alpha}_{1:M+P}^T$
Update scalar products $\mathbf{\Theta}$ and $\mathbf{\Psi}$ like in (7)
Save copy of scalar products $\tilde{\mathbf{\Psi}}_{1:N} = \mathbf{\Psi}_{1:N}$

2 for $i = 1 : M$ (select coefficients one by one)

2.1 Estimate coefficient i as in Alg. 2

2.2 for $l = 1 : N_{\text{it}}$ (improve the j-sparse solution)

2.2.1 Update $j = i$ coefficients as in Alg. 3

2.3 $\check{\mathbf{x}}_{i,1:i} = \mathbf{x}_{1:i}$ (store current coefficients)

3 Estimate the support size L using PLS

4 Use the stored values $\check{\mathbf{x}}_{L,1:L}$ for the coefficients

The coefficient positions j are chosen randomly, the amplitude a_j and the initial phase ϕ_j are uniformly distributed in $[0.05, 1]$ and $[0, 2\pi]$. The variation speed is given by the product fT_s. The filter is normed such that the average norm is $E\{||h_i||_2^2\} = 1$. The inputs $d(t)$ are normally distributed according to $\mathcal{N}(0, 1)$ and the outputs are corrupted by an additive Gaussian noise with $\sigma^2 = 0.01$. We measure the performance of the algorithms in terms of the coefficient mean square error

$$\mathbf{MSE}(t) = E\{||\mathbf{h} - \mathbf{x}||_2^2\}, \tag{16}$$

where $\mathbf{h}$ contains the actual values of the coefficients and $\mathbf{x}$ their estimates; 1000 test runs were used to estimate the MSE.

The tested algorithms are as follows: RLS, the standard algorithm using the full filter of length N; RLS-SP, the sparsity aware RLS algorithm with prior knowledge of the position and number of coefficients; GRLS and ICAMP algorithms from [3] respectively from [4] and ICAMP-A, the approximate algorithm presented herein, with prior knowledge of the support size, $M = L_t$; GRLS/ICAMP(-A)-F/V, the versions of the above algorithms that estimate online the sparsity level using the PLS criterion in conjunction with a fixed (-F) threshold $M = 20$ or a variable (-V) threshold with $\Delta = 5$; SPARLS, the algorithm presented in [2]; TNWL, the best of the algorithms from [1], using an optimized forgetting factor for the inner RLS loop.

The GRLS, SPARLS and TNWL algorithms are configured according to the recommendations from the corresponding articles. The parameter γ used in SPALRS and the forgetting factor f_{RLS} used in the inner RLS loop in TNWL were optimized using a grid search. For all the cyclic algorithms the number of optimization rounds was $N_{\text{it}} = 5$.

In Table 1 we present the average MSE (averaged over the last 100 samples) for all the studied algorithms. Fig. 1 contains the time evolution of our algorithm using a fixed M for $fT_s = 0.001$; at time $t = 500$, three of the $L_t = 5$ coefficients randomly change positions. In Fig. 2 we present the

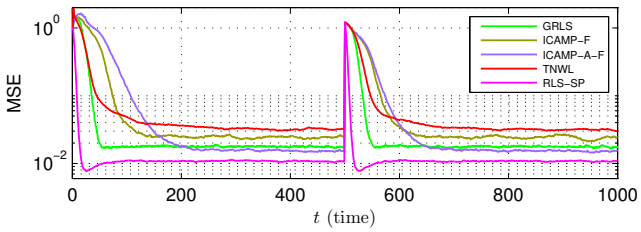

Fig. 1. MSE for $L_t = 5$ and $fT_s = 0.001$.

Table 1. Average MSE for the studied algorithms.

fT_s	0.002	0.001	0.0005	0.0002	0.0001
λ	0.90	0.92	0.94	0.96	0.98
RLS	5.3012	1.4565	0.36427	0.09774	0.04734
RLS-SP	0.0267	0.0110	0.00518	0.00246	0.00306
GRLS	0.0501	0.0178	0.00785	0.00343	0.00343
GRLS-F	0.0511	0.0189	0.00853	0.00356	0.00357
GRLS-V	0.0569	0.0187	0.00762	0.00330	0.00340
ICAMP	0.0534	0.0181	0.00790	0.00346	0.00343
ICAMP-F	0.0881	0.0249	0.00898	0.00343	0.00348
ICAMP-V	0.0683	0.0194	0.00762	0.00326	0.00340
ICAMP-A	0.0321	0.0139	0.00673	0.00310	0.00338
ICAMP-A-F	0.0409	0.0155	0.00706	0.00320	0.00342
ICAMP-A-V	0.0370	0.0146	0.00703	0.00323	0.00340
SPARLS	0.4417	0.1578	0.04225	0.01120	0.00767
γ	170	110	75	50	75
TNWL	0.1491	0.0311	0.01248	0.00471	0.00386
$\lambda_{\mathrm{RLS_{opt}}}$	0.9975	0.9825	0.9800	0.9850	0.9900

evolution of the MSE and the convergence time as a function of the buffer length P, also for $fT_s = 0.001$. For low values of the buffer length P the algorithms (ICAMP-A-F for $P = 2$, ICAMP-A-V for $P = 2, 3$) exhibit large errors for a small number of tests; we removed the associated test runs from the data presented in the figures. The buffer $\mathcal{B}$ mitigates any unwanted rapid changes in the support and allows the algorithms to achieve a lower MSE. The length P can be chosen as a trade off between the complexity and the MSE; it should be selected sufficiently large to reject the possible instabilities due to the approximations made; too large values have a negative impact on the convergence time (Fig. 2).

5. CONCLUSIONS

We propose an adaptive low complexity algorithm, employing a fixed length buffer and a series of approximations, able to outperform the non approximate counterpart and other competing adaptive algorithms. The decrease in complexity due to the update of the scalar products is proportional with

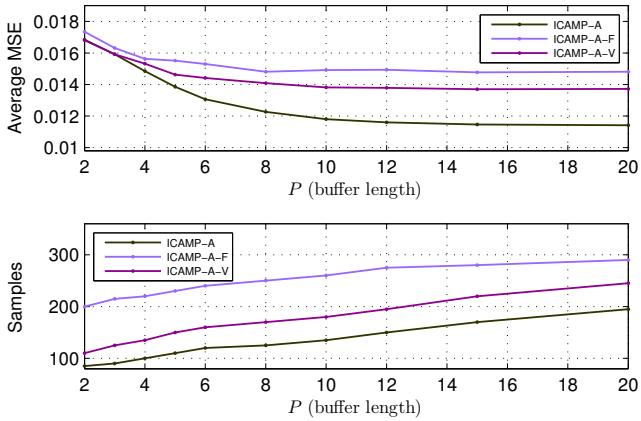

Fig. 2. Top: Average MSE as a function of the buffer length P. Bottom: Convergence time (number of samples until the MSE is within 5% of the average MSE) as a function of the buffer length P. The variation time is given by $fT_s = 0.001$.

$\frac{1}{2}\frac{N}{M+P}$ which, for low sparsity levels and small buffers, becomes substantial. The choice of buffer length P is a trade off between complexity and robustness.

6. REFERENCES

[1] D. Angelosante, J.A. Bazerque, and G.B. Giannakis, "Online Adaptive Estimation of Sparse Signals: Where RLS Meets the l_1-Norm," *IEEE Trans. Sign. Proc.*, vol. 58, no. 7, pp. 3436–3447, Jul. 2010.

[2] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The Sparse RLS Algorithm," *IEEE Trans. Sign. Proc.*, vol. 58, no. 8, 2010.

[3] B. Dumitrescu, A. Onose, P. Helin, and I. Tăbuş, "Greedy Sparse RLS," *IEEE Trans. Sign. Proc. (to appear)*, 2012.

[4] A. Onose and B. Dumitrescu, "Cyclic Adaptive Matching Pursuit," in *ICASSP, Kyoto, Japan*, Mar. 2012.

[5] S.G. Mallat and Z. Zhang, "Matching Pursuit with Time Frequency Dictionaries," *IEEE Trans. Sgn. Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.

[6] S.F. Cotter and B.D. Rao, "The Adaptive Matching Pursuit Algorithm for Estimation and Equalization of Sparse Time-Varying Channels," in *34th Asilomar Conf. Sign. Syst. Comp.*, 2000, vol. 2, pp. 1772–1776.

[7] M.G. Christensen and S.H. Jensen, "The Cyclic Matching Pursuit and its Application to Audio Modeling and Coding," in *41th Asilomar Conf. Sign. Syst. Comp.*, Nov. 2007, pp. 550–554.

[8] J. Rissanen, "Order Estimation by Accumulated Prediction Errors," *J. Appl. Probab.*, vol. 23, pp. 55–61, 1986.