Sliding-window RLS low-cost implementation of proportionate affine projection algorithms

Yuriy Zakharov[†], *Senior Member*, *IEEE*, and Vítor H. Nascimento[††], *Senior Member*, *IEEE*

Abstract

This paper addresses adaptive filtering for sparse identification. Proportionate affine projection algorithms (PAPAs) are known to be efficient techniques for this purpose. We show that the PAPA performance may improve with an increase in the projection order M (for example, such as $M = 512$), which, however, also results in an increased complexity; the complexity is in general $\mathcal{O}(M^2N)$ or at least $\mathcal{O}(MN)$ operations per sample, where N is the filter length. We show that PAPAs are equivalent to specific sliding-window recursive least squares (SRLS) adaptive algorithms with time-varying and tap-varying diagonal loading (SRLS-VDLs). We then propose an approximation to the SRLS-VDLs based on dichotomous coordinate descent (DCD) iterations with a complexity of $\mathcal{O}(N_uN)$, which does not depend on M; it depends on the number of DCD iterations N_u, which as we show can be significantly smaller than M, thus allowing a low-complexity implementation of PAPA adaptive filters.

Index Terms

Adaptive filter, affine projection, diagonal loading, dichotomous coordinate descent, DCD, PAPA, RLS, sliding window, sparse identification

I. Introduction

Compared to the classical least squares (LS) identification algorithms, the algorithms exploiting system sparsity can achieve a better performance. Various techniques have been proposed in the literature for adaptive sparse identification, especially in application to echo cancelation [1]–[6]. The proportionate affine projection algorithms (PAPAs) [7] are considered as good candidates for this application [4]. In the PAPA family, significant attention is paid to the improved PAPA (IPAPA) and its multiple modifications (see [8]–[15] and references therein). In general, the complexity of sparse adaptive algorithms is higher than that of their LS counterparts. Therefore, reduction of the complexity is an important issue.

 [†]Department of Electronics, Univ. of York, UK, yz1@ohm.york.ac.uk. [††]Dept. of Electronic Systems Eng. Univ. of São Paulo, Brazil, vitor@lps.usp.br. The work of Y. V. Zakharov was partly supported by FAPESP grant 2011/06994-2 and the York-FAPESP grant 2012/50565-1. The work of V. H. Nascimento was partly supported by CNPq and FAPESP grants 303921/2010-2 and 2012/50565-1.

The complexity of PAPA adaptive filters of length N and projection order M is in general $\mathcal{O}(M^2N)$, but it can be reduced down to $\mathcal{O}(MN)$ operations per sample for specific weight matrices [4]. Typically, the value M is suggested to be chosen from 2 to 8 to keep the complexity low. Further increase in M results in significant increase of the algorithm complexity. The contributions of this paper are:

- we show through examples that increasing the PAPA projection order M to higher values (e.g., such as $M = 512$) than usually proposed in the literature may result in significantly improved identification performance;
- we show that PAPA adaptive filters are equivalent to specific sliding-window recursive least squares (SRLS) adaptive filters with time-varying and tap-varying diagonal loading, which we call SRLS-VDL algorithms;
- we show that the SRLS-VDL algorithms can be approximately implemented with a complexity of $\mathcal{O}(N_uN)$, where $N_u \ll M$, i.e., we propose a low-cost approximate implementation of PAPA adaptive filters with a complexity independent of the projection order; we call corresponding techniques SRLS-VDL-DCD algorithms.

The parameter N_u is the number of dichotomous coordinate descent (DCD) iterations used at each time instant for the approximation. The DCD algorithm is a low-complexity iterative technique for solving systems of linear equations without multiplications or divisions [16], [17]. Thus, the proposed implementation of PAPA adaptive filters makes it possible to take advantage of the improved performance achieved with higher projection order M, and at the same time reduces the complexity compared even to that of standard PAPA implementations using a low order M.

DCD iterations have previously been applied to the PAPA family of algorithms to implement the *memory improved* PAPA (MIPAPA) adaptive filter [9], [10], [12], which in particular has allowed an efficient FPGA implementation of the adaptive filter [18]. The MIPAPA adaptive filter achieves a significant reduction in complexity compared to the *improved* PAPA (IPAPA) adaptive filter (e.g., see Fig. 9 and Fig. 11) via a specific choice of the weight matrix [8]. Further reduction in complexity is achieved by using DCD iterations to solve an $M \times M$ system of linear equations (instead of a direct matrix inversion) at each time step. However, these linear systems are different at each time instant, and there is no simple method for using the solution at one instant to provide a good initial condition (*warm* start) for the next. Since every linear system must be solved from scratch, a relatively large number of DCD iterations (of the order of M) is necessary in the MIPAPA (or other PAPA) adaptive filters. The approach proposed here obtains a large reduction in complexity in comparison with the above approach, because the system of linear equations that must be solved at each time instant in the proposed algorithm, although being a large $N \times N$ system, can be cast in an iterative form with a good *warm* start, resulting in good performance with a relatively small number of DCD iterations N_u, e.g., $N_u = 16$ or even smaller.

Note that there were several attempts in the literature to give new interpretations to PAPA adaptive filters, resulting in novel useful adaptive filters. In [19], it is shown a relation between PAPAs and Kalman filters. In [20], a relation

is shown between proportionate adaptive filters and the basis pursuit. Here, we show another useful relation, now between PAPA and SRLS recursions, that also results in new efficient adaptive filters.

The paper is organized as follows. In Section II, we introduce the signal model, PAPA and SRLS adaptive filters, and analyse their complexities. Section III describes a relation between PAPA and SRLS adaptive filters and introduces the SRLS-VDL adaptive algorithm. Section IV presents the SRLS-VDL-DCD adaptive algorithm, which is a low-cost implementation of the SRLS-VDL algorithm. Section V presents a comparison of the performance and complexity of the proposed and known adaptive algorithms. Finally, Section VI draws conclusions. The Appendix contains the derivation of the SRLS-VDL-DCD algorithm.

Notation: We use capital and small bold fonts to denote matrices and vectors, respectively; e.g., $\mathbf{X}$ is a matrix and $\mathbf{x}$ is a vector. Elements of a vector $\mathbf{q}$ are denoted as q_i; $(\cdot)^T$ denotes the transpose of a matrix; $\mathbf{I}_N$ is an $N \times N$ identity matrix; $\mathbf{0}_{N,K}$ denotes an $N \times K$ matrix with all-zero entries.

II. Signal model, PAPA and SRLS adaptive filters

The signal model corresponding to the identification scenario is described by the relationship

$$d(n) = \mathbf{h}^T(n)\mathbf{x}(n) + \nu(n), \tag{1}$$

where $d(n)$ is the noisy output of the system to be identified, $\nu(n)$ is a zero-mean Gaussian random noise with variance σ^2, and the vector $\mathbf{x}(n) = [x(n)\ x(n-1)\ \ldots\ x(n-N+1)]^T$ represents the input to the system with unknown impulse response $\mathbf{h}(n)$ of length N. We are interested in scenarios where the vector $\mathbf{h}(n)$ to be identified is sparse.

A. PAPA adaptive filters

We denote $\mathbf{X}(n) = [\mathbf{x}(n), \ldots, \mathbf{x}(n-M+1)]^T$ and $\mathbf{d}(n) = [d(n), \ldots, d(n-M+1)]^T$ as the $M \times N$ matrix of the regressor data and $M \times 1$ vector of the desired signal, respectively, where M is the length of a sliding window. Consider the generalized PAPA recursion [4]:

$$\mathbf{R}_Q(n) = \delta\mathbf{I}_M + \mathbf{X}(n)\mathbf{Q}(n)\mathbf{X}^T(n), \tag{2}$$

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{X}(n)\hat{\mathbf{h}}(n-1), \tag{3}$$

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu\mathbf{Q}(n)\mathbf{X}^T(n)\mathbf{R}_Q^{-1}(n)\mathbf{e}(n), \tag{4}$$

where $\mu > 0$ is a step-size parameter, $\delta > 0$ is a regularization parameter, and $n > 0$ is the time index. The PAPA can be implemented replacing the matrix inversion $\mathbf{R}_Q^{-1}(n)$ and further multiplication of it by the error vector $\mathbf{e}(n)$ in (4), with the solution of the $M \times M$ system of equations $\mathbf{R}_Q(n)\boldsymbol{\varepsilon}(n) = \mathbf{e}(n)$ [21], [22] as shown in Table I.

TABLE I
GENERAL STRUCTURE OF PAPA ADAPTIVE FILTERS

	Initialization: $\hat{\mathbf{h}}(0) = \mathbf{0}_{N,1}$
	for $n = 1, 2, \ldots$
1.	Update $\mathbf{Q}(n)$ using $\hat{\mathbf{h}}(n-1)$
2.	$\mathbf{R}_Q(n) = \delta \mathbf{I}_M + \mathbf{X}(n)\mathbf{Q}(n)\mathbf{X}^T(n)$
3.	$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{X}(n)\hat{\mathbf{h}}(n-1)$
4.	Solve $\mathbf{R}_Q(n)\boldsymbol{\varepsilon}(n) = \mathbf{e}(n)$ w.r.t. $\boldsymbol{\varepsilon}(n)$
5.	$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu\mathbf{Q}(n)\mathbf{X}^T(n)\boldsymbol{\varepsilon}(n)$

The $N \times N$ matrix $\mathbf{Q}(n)$ is typically chosen as a diagonal matrix $\mathbf{Q}(n) = \text{diag}\{\mathbf{q}(n)\}$ and the vector $\mathbf{q}(n)$ of its diagonal elements relates to the impulse response estimate $\hat{\mathbf{h}}(n-1)$ obtained at time instant $(n-1)$; this relationship is different for different PAPAs. The key idea for the choice of the vector $\mathbf{q}(n)$ is that its elements $q_i(n)$ should be large if the corresponding $|h_i(n-1)|$ are large; e.g., in the IPAPA adaptive filter, the elements $q_i(n)$ $(i = 0, \ldots, N-1)$ are given by [4], [23]

$$q_i(n) = \frac{1-\kappa}{2N} + \frac{1+\kappa}{2}\frac{|\hat{h}_i(n-1)|}{||\hat{\mathbf{h}}(n-1)||_1 + \epsilon}, \tag{5}$$

where $\epsilon > 0$ is a small number to avoid division by zero, $-1 \leq \kappa < 1$, and

$$||\hat{\mathbf{h}}(n-1)||_1 = \sum_{i=0}^{N-1} |\hat{h}_i(n-1)|$$

is the ℓ_1-norm of the vector $\hat{\mathbf{h}}(n-1)$.

If $\mathbf{Q}(n)$ is an identity matrix, i.e., $\mathbf{Q}(n) = \mathbf{I}_N$, we arrive at the classical regularized affine projection algorithm (APA) [24].

B. Classical SRLS adaptive filter

The classical SRLS algorithm at every time instant n finds an estimate $\tilde{\mathbf{h}}(n)$ of the impulse response $\mathbf{h}(n)$ as a solution to the $N \times N$ system of normal equations [25], [26]

$$\mathbf{R}(n)\tilde{\mathbf{h}}(n) = \mathbf{b}(n), \tag{6}$$

where $\mathbf{R}(n) = \mathbf{X}^T(n)\mathbf{X}(n)$ and $\mathbf{b}(n) = \mathbf{X}^T(n)\mathbf{d}(n)$. The matrix $\mathbf{R}(n)$ can be rank-deficient and then solving the equation in (6) can be ill-posed. To stabilize the SRLS solution, diagonal loading (regularization) is used, so instead of (6), a modified system is solved,

$$\mathbf{R}_W(n)\tilde{\mathbf{h}}(n) = \mathbf{b}(n), \tag{7}$$

where $\mathbf{R}_W(n) = \mathbf{R}(n) + \mathbf{W}(n)$, $\mathbf{W}(n) = \delta\mathbf{I}_N$, and δ is the regularization parameter.

Further improvement can be achieved if the final estimate $\hat{\mathbf{h}}(n)$ of $\mathbf{h}(n)$ is obtained using the recursion

$$\hat{\mathbf{h}}(n) = (1-\mu)\hat{\mathbf{h}}(n-1) + \mu\tilde{\mathbf{h}}(n), \tag{8}$$

where μ is a step-size parameter ($0 < \mu \leq 1$). When $\mu = 1$, we have $\hat{\mathbf{h}}(n) = \tilde{\mathbf{h}}(n)$ and we arrive at the classical SRLS algorithm as defined by (6). For $\mu < 1$, the estimate $\hat{\mathbf{h}}(n)$ is the result of averaging in time by linearly combining the current estimate $\tilde{\mathbf{h}}(n)$ with the weight μ and the previous estimate $\hat{\mathbf{h}}(n-1)$ with the weight $1-\mu$. The closer μ is to zero, the larger the memory of the algorithm. See also [26] on using such a step-size parameter in recursive least squares (RLS) adaptive filters.

C. Relation between the regularized APA and SRLS adaptive filters

In [24], a relation has been shown between the regularized APA and the SRLS algorithm. More specifically, it is shown that the regularized APA recursion (2)–(4) with $\mathbf{Q}(n) = \mathbf{I}_N$ can be implemented as the following SRLS recursion

$$\begin{aligned}
\mathbf{R}_W(n) &= \delta\mathbf{I}_N + \mathbf{X}^T(n)\mathbf{X}(n), && (9)\\
\mathbf{e}(n) &= \mathbf{d}(n) - \mathbf{X}(n)\hat{\mathbf{h}}(n-1), && (10)\\
\hat{\mathbf{h}}(n) &= \hat{\mathbf{h}}(n-1) + \mu\mathbf{R}_W^{-1}(n)\mathbf{X}^T(n)\mathbf{e}(n). && (11)
\end{aligned}$$

Substituting (10) in (11), we can represent this recursion in another form, which will be useful for our further consideration,

$$\begin{aligned}
\hat{\mathbf{h}}(n) &= (1-\mu)\hat{\mathbf{h}}(n-1)\\
&+ \mu\mathbf{R}_W^{-1}(n)\left[\mathbf{b}(n) + \delta\hat{\mathbf{h}}(n-1)\right], && (12)
\end{aligned}$$

or

$$\hat{\mathbf{h}}(n) = (1-\mu)\hat{\mathbf{h}}(n-1) + \mu\tilde{\mathbf{h}}(n), \tag{13}$$

where again $\mathbf{b}(n) = \mathbf{X}^T(n)\mathbf{d}(n)$, and $\tilde{\mathbf{h}}(n)$ is a solution to the system of equations

$$\mathbf{R}_W(n)\tilde{\mathbf{h}}(n) = \mathbf{b}(n) + \delta\hat{\mathbf{h}}(n-1). \tag{14}$$

Thus, the APA can be implemented using either the usual APA recursion (2)–(4) with $\mathbf{Q}(n) = \mathbf{I}_N$ or the regularized SRLS recursion (9)–(11).

Note that the regularized SRLS recursion (8) with the system (7) is similar to the recursion (13) with the

system (14), except that the system (14) contains an extra term $\delta\hat{\mathbf{h}}(n-1)$. This term plays a stabilizing role when $M < N$.

D. Complexity of PAPA and SRLS adaptive filters

Let us consider the complexity of these two implementations. For PAPA, the computation of matrix $\mathbf{R}_Q(n)$ requires in general $\mathcal{O}(M^2N)$ operations per time instant; this however can be reduced to $\mathcal{O}(MN)$ using a specific choice of the matrix $\mathbf{Q}(n)$ as done for the *memory* PAPA [4]. The other steps for small M at most require $\mathcal{O}(MN)$ operations. Therefore, for small M, the PAPA complexity can be thought of as at least $\mathcal{O}(MN)$ operations per time instant. For large M, the matrix inversion $\mathbf{R}_Q^{-1}(n)$ may require up to $\mathcal{O}(M^3)$ operations per time instant and it will dominate the complexity; however, approximate techniques can be used for this purpose to reduce the complexity [9], [18] (see below). We show in Section V that large values of M however are useful for improving the PAPA identification performance (see Fig. 2 and Fig. 5).

More specifically, for the IPAPA adaptive filter, the complexity of computing the matrix $\mathbf{R}_Q(n)$ (see step 2 in Table I) is about $(M^2N + MN)/2$ multiplications and $(M^2N + MN - M^2)/2$ additions [18]; this is the most significant contribution to the IPAPA complexity. For the MIPAPA, computing the matrix $\mathbf{R}_Q(n)$ requires about $(2MN - N)$ multiplications and $2MN - 2M - N$ additions [18]. Solving the system of equations at step 4 in both the adaptive filters can be efficiently done using the DCD algorithm with $N_u \approx 5M$ DCD iterations, thus requiring about $10M^2$ additions [9], [18]. The other steps of the adaptive filters involve: computation of the error vector $\mathbf{e}(n)$ at step 3, requiring MN multiplications and MN additions; the filter update at step 5, requiring MN multiplications and MN additions; and updating $\mathbf{Q}(n)$ at step 1, requiring N multiplications and $2N$ additions. Summarizing this analysis, we can conclude that the IPAPA complexity is approximately $0.5M^2N + 2.5MN$ multiplications and $0.5M^2N + 2.5MN + 9.5M^2$ additions. For the MIPAPA adaptive filter, the complexity is approximately $4MN$ multiplications and $4MN + 10M^2$ additions.

Regarding the SRLS recursion, due to the matrix inversion in (11) or, alternatively, solving the system in (14), the complexity is in general $\mathcal{O}(N^3)$. Using the matrix inversion lemma [27], this can be reduced down to $\mathcal{O}(N^2)$ operations per time instant, which is still too high in practice. However, the approximate and numerically stable implementation of the regularized SRLS algorithm based on DCD iterations and proposed in [22] has a complexity of $\mathcal{O}(N_uN)$, while retaining the SRLS performance even with a small number of DCD iterations N_u. Therefore, if M is high compared to the parameter N_u, it can be less costly to use the SRLS recursion than the APA recursion, while avoiding possible numerical instabilities inherent to fast implementations of the exact algorithms.

In the next section, we extend the relation between APA and SRLS recursions to a more general relation between the PAPA and specific SRLS recursions. We then exploit this result to obtain a reduced-complexity implementation of the PAPA adaptive filters.

III. Relation between PAPA and SRLS recursions

We now show that the PAPA recursions can be transformed into specific SRLS recursions. We first apply the matrix inversion lemma [27] to the matrix $\mathbf{R}_Q(n)$ in (2) and obtain

$$\mathbf{R}_Q^{-1}(n) = \frac{1}{\delta}\mathbf{I}_M - \frac{1}{\delta}\mathbf{X}(n)[\delta\mathbf{Q}^{-1}(n) + \mathbf{R}(n)]^{-1}\mathbf{X}^T(n). \tag{15}$$

Multiplying $\mathbf{R}_Q^{-1}(n)$ by $\mathbf{X}^T(n)$ and denoting $\mathbf{W}(n) = \delta\mathbf{Q}^{-1}(n)$ and $\mathbf{R}_W = \mathbf{R}(n) + \mathbf{W}(n)$, we obtain

$$\mathbf{X}^T(n)\mathbf{R}_Q^{-1}(n) = \frac{1}{\delta}\mathbf{X}^T(n) - \frac{1}{\delta}\mathbf{R}(n)\mathbf{R}_W^{-1}(n)\mathbf{X}^T(n). \tag{16}$$

Multiplying both sides in (16) by vector $\mathbf{e}(n)$ from the right and denoting $\mathbf{z}(n) = \mathbf{X}^T(n)\mathbf{e}(n)$, after some algebra we arrive at

$$\begin{aligned}\mathbf{X}^T(n)\mathbf{R}_Q^{-1}(n)\mathbf{e}(n) &= \frac{1}{\delta}\mathbf{z}(n) - \frac{1}{\delta}\mathbf{R}(n)\mathbf{R}_W^{-1}(n)\mathbf{z}(n) \\ &= \frac{1}{\delta}\mathbf{W}(n)\mathbf{R}_W^{-1}(n)\mathbf{z}(n). \end{aligned}\tag{17}$$

Substituting (17) in (4) and taking into account that $\mathbf{W}(n) = \delta\mathbf{Q}^{-1}(n)$, we arrive at an equivalent form of the PAPA recursion:

$$\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n-1) + \mu\mathbf{R}_W^{-1}(n)\mathbf{z}(n). \tag{18}$$

We now take into account that

$$\mathbf{z}(n) = \mathbf{X}^T(n)[\mathbf{d}(n) - \mathbf{X}^T(n)\hat{\mathbf{h}}(n-1)] \tag{19}$$

$$= \mathbf{X}^T(n)\mathbf{d}(n) - \mathbf{R}(n)\hat{\mathbf{h}}(n-1) \tag{20}$$

to modify the recursion in (18) as

$$\begin{aligned}\hat{\mathbf{h}}(n) &= (1-\mu)\hat{\mathbf{h}}(n-1) \\ &+ \mu\mathbf{R}_W^{-1}(n)\left[\mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1)\right]. \end{aligned}\tag{21}$$

Denoting

$$\tilde{\mathbf{h}}(n) = \mathbf{R}_W^{-1}(n)\left[\mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1)\right], \tag{22}$$

the final estimate can again be represented in the form

$$\hat{\mathbf{h}}(n) = (1-\mu)\hat{\mathbf{h}}(n-1) + \mu\tilde{\mathbf{h}}(n). \tag{23}$$

TABLE II
SRLS-VDL ADAPTIVE FILTER

	Initialization: $\hat{\mathbf{h}}(0) = \mathbf{0}_{N,1}$ $\mathbf{R}(0) = \mathbf{0}_{N,N}$
	for $n = 1, 2, \ldots$
1.	$\mathbf{R}(n) = \mathbf{X}^T(n)\mathbf{X}(n)$
2.	Update the matrix $\mathbf{W}(n)$
3.	$\mathbf{R}_W(n) = \mathbf{R}(n) + \mathbf{W}(n)$
4.	$\mathbf{b}(n) = \mathbf{X}^T(n)\mathbf{d}(n)$
5.	Solve $\mathbf{R}_W(n)\tilde{\mathbf{h}}(n) = \mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1)$
6.	$\hat{\mathbf{h}}(n) = (1-\mu)\hat{\mathbf{h}}(n-1) + \mu\tilde{\mathbf{h}}(n)$

If $\mathbf{Q}(n) = \mathbf{I}_N$ and, consequently, $\mathbf{W}(n) = \delta\mathbf{I}_N$, from (23) we obtain, as a particular case, the SRLS recursion (14) corresponding to the classical APA.

The sequence of steps described above can be summarized into the adaptive algorithm shown in Table II. We call it an SRLS adaptive algorithm with time-variant and tap-variant diagonal loading (SRLS-VDL), since the diagonal weighting matrix $\mathbf{W}(n)$ varies with time n and its diagonal elements are nonuniform. At every time instant n, the SRLS-VDL algorithm generates an estimate $\hat{\mathbf{h}}(n)$ which is exactly the same as that generated by the PAPA, provided that $\mathbf{W}(n) = \delta\mathbf{Q}^{-1}(n)$. This is just a different set of operations (different implementation) leading to the same result. Note that the matrix $\mathbf{Q}(n)$ defines a specific PAPA. If $\mathbf{W}(n)$ does not satisfy this equality, the estimate will be equal to that produced by another PAPA, whose matrix $\mathbf{Q}(n)$ satisfies the equality $\mathbf{Q}(n) = \delta^{-1}\mathbf{W}^{-1}(n)$.

In general, this implementation is not simpler than the implementation using the PAPA recursion given by (2)–(4). However, to the algorithm in Table II, we can apply the approach described in [22] and further exploited in [6], [28], [29] to arrive at a low-complexity approximation of the SRLS-VDL adaptive filter, as shown in the next section.

IV. LOW-COMPLEXITY IMPLEMENTATION OF THE SRLS-VDL ALGORITHM: SRLS-VDL-DCD ALGORITHM

In Appendix I, we show that the SRLS-VDL algorithm presented in Table II can be approximately implemented using the SRLS-VDL-DCD adaptive filter presented in Table III. Here, $\tilde{y}(n)$ is the output of the adaptive filter and $\tilde{y}_M(n)$ is a delayed output; consequently, $\tilde{e}(n)$ is the *a priori* error and $\tilde{e}_M(n)$ is a delayed *a priori* error. The vector $\mathbf{r}(n)$ is the residual vector that takes into account that the system of equations $\mathbf{R}_W(n)\tilde{\mathbf{h}}(n) = \mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1)$ is solved with $\hat{\tilde{\mathbf{h}}}(n-1)$ as a *warm-start*. Note that if the system at step 8 is solved exactly or the number of DCD iterations is not limited then the SRLS-VDL-DCD adaptive filter will produce exactly the same results as the SRLS-VDL or PAPA adaptive filter. However, such an approach would not bring any reduction in the complexity. Since we can use the previous solution as an initial condition, a small number N_u of DCD iterations is enough to achieve a performance close to that of the PAPA adaptive filter, thus reducing the complexity.

The most suitable version of the DCD algorithm for adaptive filtering is the leading DCD algorithm [22], [30].

TABLE III
SRLS-VDL-DCD ADAPTIVE FILTER

	Initialization: $\hat{\mathbf{h}}(0)=\mathbf{0}_{N,1}$, $\hat{\mathbf{h}}(-1)=\mathbf{0}_{N,1}$, $\hat{\tilde{\mathbf{h}}}(0)=\mathbf{0}_{N,1}$, $\mathbf{r}(0)=\mathbf{0}_{N,1}$, $\mathbf{R}(0)=\mathbf{0}_{N,N}$	$\times$	$\pm$
	for $n=1,2,\ldots$		
1.	$\mathbf{R}(n)=\mathbf{X}^T(n)\mathbf{X}(n)$	$2N$	$2N$
2.	Update the matrix $\mathbf{W}(n)$	$P_{w,m}$	$P_{w,a}$
3.	$\tilde{y}(n)=\hat{\tilde{\mathbf{h}}}^T(n-1)\mathbf{x}(n)$	N	N
4.	$\tilde{y}_M(n)=\hat{\tilde{\mathbf{h}}}^T(n-1)\mathbf{x}(n-M)$	N	N
5.	$\tilde{e}(n)=d(n)-\tilde{y}(n)$	$-$	1
6.	$\tilde{e}_M(n)=d(n-M)-\tilde{y}_M(n)$	$-$	1
7.	$\mathbf{r}(n)=\mathbf{r}(n-1)+\tilde{e}(n)\mathbf{x}(n)-\tilde{e}_M(n)\mathbf{x}(n-M)$ $+\mathbf{W}(n)\hat{\mathbf{h}}(n-1)-\mathbf{W}(n-1)\hat{\mathbf{h}}(n-2)$ $-\Delta\mathbf{W}(n)\hat{\tilde{\mathbf{h}}}(n-1)$	$5N$	$5N$
8.	Solve $[\mathbf{R}(n)+\mathbf{W}(n)]\Delta\tilde{\mathbf{h}}(n)=\mathbf{r}(n)$ using DCD iterations to obtain $\Delta\hat{\tilde{\mathbf{h}}}(n)$ and update $\mathbf{r}(n)$ and $\hat{\tilde{\mathbf{h}}}(n)=\hat{\tilde{\mathbf{h}}}(n-1)+\Delta\hat{\tilde{\mathbf{h}}}(n)$	$-$	$2N_uN$
9.	$\hat{\mathbf{h}}(n)=(1-\mu)\hat{\mathbf{h}}(n-1)+\mu\hat{\tilde{\mathbf{h}}}(n)$	$2N$	N

It requires about $2N_uN$ additions and no multiplications or divisions. This algorithm is based on a fixed-point representation of the solution in an amplitude interval $[-H,+H]$ with M_b bits; see more details on the DCD algorithm, e.g., in [17], [22].

We now estimate the complexity of the SRLS-VDL-DCD algorithm in Table III. The matrix update at step 1 requires $2N$ multiplications and $2N$ additions (see [22] for details). The filtering steps 3 and 4 each require N multiplications and N additions. Computation of the residual vector $\mathbf{r}(n)$ at step 7 requires $5N$ multiplications and $5N$ additions. Step 8 involves computing $\mathbf{R}(n)+\mathbf{W}(n)$ and applying the DCD algorithm, including updating $\mathbf{r}(n)$ and $\hat{\tilde{\mathbf{h}}}(n)$, that requires $2N_uN$ additions (see [22] and [17] for more details). The update at step 9 involves $2N$ multiplications and N additions.

Computation of the diagonal elements of the weight matrix $\mathbf{W}(n)$ at step 2 is specific for each specific PAPA adaptive filter. E.g., if we use the SRLS-VLD-DCD algorithm in order to approximately implement the IPAPA adaptive filter, then the weights are defined by elements q_i given by equation (5). Direct computation according to (5) would require $\mathcal{O}(N)$ operations. However, at each time instant, only a few (N_u) filter taps are updated. Consequently, only N_u elements of $\mathbf{W}(n)$ need to be updated with a complexity of $\mathcal{O}(N_u)$. Since $N_u \ll N$, this complexity, i.e., the number of multiplications $P_{w,m}$ and number of additions $P_{w,a}$, can be neglected.

By summarizing this complexity analysis, we see that the SRLS-VDL-DCD algorithm can be implemented with $11N$ multiplications and $(10+2N_u)N$ additions. It is important to notice that the number of multiplications is a fixed value that only depends on the number of filter taps N and does not depend on the number of DCD iterations

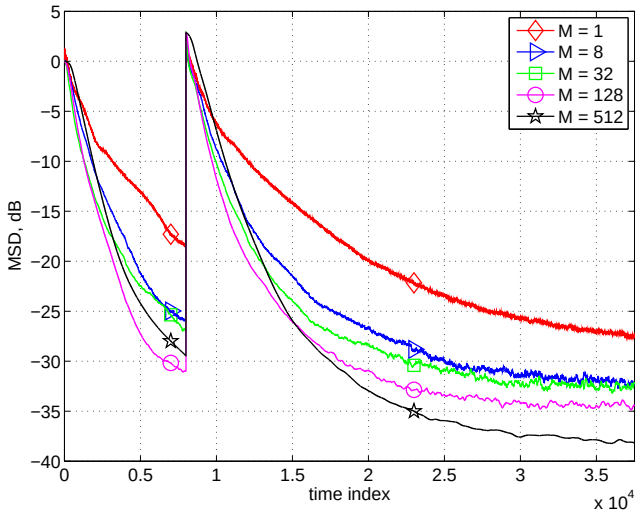

Fig. 1. MSD performance of the IPAPA adaptive filter for different values of the projection order M averaged over 10 realizations of AR(1) input signal of unit variance with AR parameter 0.8. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 30 dB. The other IPAPA parameters are: $\kappa = 0.99$, $\epsilon = 10^{-4}$, $\delta = 1$, $\mu = 0.5/M$.

N_u. Only the number of additions increases proportionally to the number of DCD iterations.

V. Numerical results

In this section, we present results of computer simulation of the proposed SRLS-VDL-DCD adaptive filter. We compare the Mean Square Deviation (MSD) performance of this adaptive filter against the IPAPA adaptive filter. As the input regressors $\mathbf{x}(n)$, we use a real speech signal sampled at a rate of 8000 Hz, each sample represented with 14 bits. The impulse response of the unknown system $\mathbf{h}(n)$ is of length $N = 512$; only $K = 100$ of its elements are non-zero. They are kept constant during the first 8000 instants (equivalent to 1 s) and then changed. The change is in both positions and values of the non-zero taps. Positions of the K non-zero elements in $\mathbf{h}(n)$ are chosen randomly. These elements are generated as independent Gaussian zero-mean random numbers of unit variance and then $\mathbf{h}(n)$ is normalized to have the unity norm. The MSD is calculated as

$$\text{MSD}(n) = \frac{||\mathbf{h}(n) - \hat{\mathbf{h}}(n)||_2^2}{||\mathbf{h}(n)||_2^2} \tag{24}$$

and plotted against the time index n; here, $||\cdot||_2$ denotes the ℓ_2–norm.

Before presenting simulation results for speech signals, we first consider a case of the input to the adaptive filter being an autoregressive random process of the first order, AR(1). Fig. 1 shows the MSD performance of the IPAPA adaptive filter for such a case with the AR parameter 0.8. It can be seen that for the chosen step-size $\mu = 0.5/M$ the convergence rate for all values of the projection order M is approximately the same; we chose the inverse dependence of μ on M to keep the convergence rate about the same for all M. However, the steady-state MSD performance improves as M increases. This indicates that it is beneficial to use the IPAPA with large values of M.

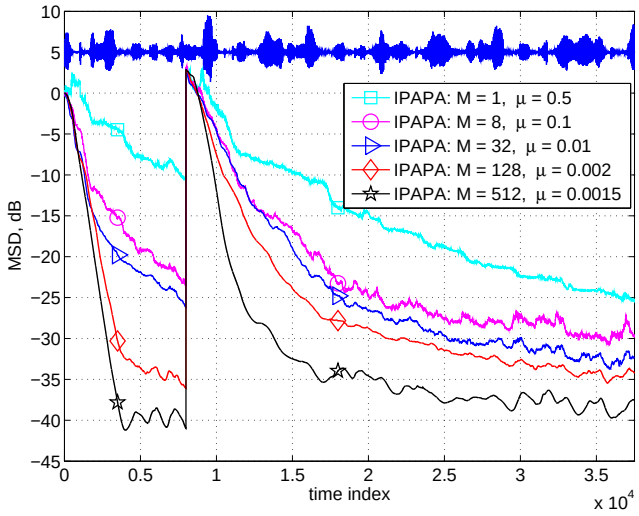

Fig. 2. MSD performance of the IPAPA adaptive filter for different values of the projection order M for one realization of the input speech signal. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other IPAPA parameters are: $\kappa = 0.99$ and $\delta = 10^4$, $\epsilon = 10^{-4}$. The step size μ is adjusted for every M to achieve the best performance. The top blue curve in the plot shows the input speech signal used for the identification.

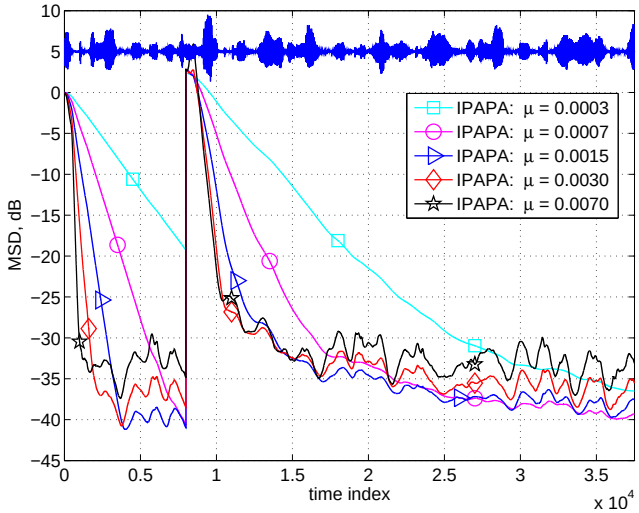

Fig. 3. MSD performance of the IPAPA adaptive filter for different values of the step-size parameter μ for one realization of the input speech signal. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other IPAPA parameters are: $M = 512$, $\kappa = 0.99$ and $\delta = 10^4$, $\epsilon = 10^{-4}$.

Fig. 2 demonstrates the improvement of the IPAPA performance with increased M for a real speech signal. Note that the speech samples are represented as integer numbers with 14 bits; e.g., the maximum magnitude of the speech signal is equal to 16260 at time instant 9933. Therefore the regularization parameter δ is set to a large value, $\delta = 10^4$. For every M, the step-size parameter μ is adjusted to guarantee the best performance. These values of μ are slightly different from those in Fig. 1. In the case of $M = 1$, we arrive at the improved proportionate NLMS (IPNLMS) algorithm [4], [23]; this algorithm shows an inferior performance compared to the IPAPA performance for higher values of M.

Fig. 3 shows the dependence of the MSD performance on the step-size μ to illustrate how, in our simulations,

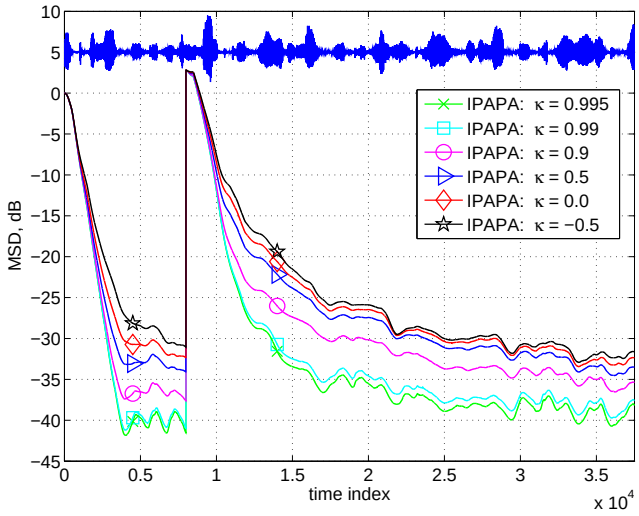

Fig. 4. MSD performance of the IPAPA adaptive filter for different values of the parameter κ for one realization of the input speech signal. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other IPAPA parameters are: $M = 512$, $\mu = 1.5 \cdot 10^{-3}$, $\delta = 10^4$, $\epsilon = 10^{-4}$.

the step-size parameter is adjusted to achieve 'the best performance'. Note that μ usually compromises between the convergence rate and steady-state MSD, and thus 'the best performance' does not exist. In our case, we try to achieve a low steady-state MSD without loosing too much in the convergence rate. In Fig. 3, it can be seen that, after the change of the impulse response, the convergence rate for $\mu = 0.0015$ is close to that for higher μ, whereas for $\mu < 0.0015$, the convergence rate decreases considerably. Therefore, we choose $\mu = 0.0015$ as a compromise step-size value.

Fig. 4 shows the dependence of the MSD performance on the parameter κ defining the matrix $\mathbf{Q}(n)$ in the IPAPA. In the IPNLMS algorithm (IPAPA with $M = 1$), good results are achieved for $\kappa = 0$ and $\kappa = -0.5$ [23]. However, for the high projection order ($M = 512$), in this scenario, the parameter κ needs to be close to unity. Therefore, for our simulation, we chose $\kappa = 0.99$.

Fig. 5 confirms this trend of improvement of the IPAPA performance with an increase in the projection order M. In this figure, we plot the average MSD obtained for different projection orders (from $M = 1$ to $M = N = 512$). The ensemble-averaged MSD is obtained using $L = 10$ different extracts (about 5-second long each) of the same speech signal as was used to generate results in Fig. 2. A large improvement in the performance is achieved when moving from $M = 1$ to 8, 32, 128 and 512, with differences of more than 5 dB between the cases $M = 8$ and $M = 512$.

Fig. 6 presents simulation results for the SRLS-VDL-DCD algorithm. It shows that, for the case of $M = 512$, that when the number of DCD iterations increases, the performance approaches that of the IPAPA with the same value of M. A small number of DCD iterations ($N_u = 8$ in this case) are enough to approach closely the IPAPA performance. However, the complexity of the SRLS-VDL-DCD adaptive filter is significantly lower than the IPAPA

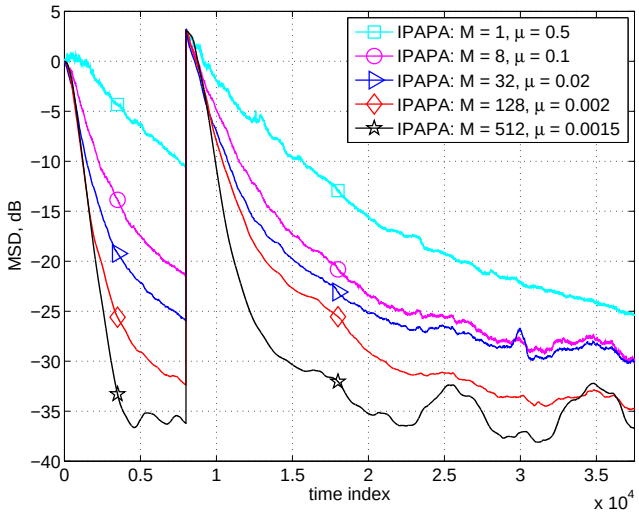

Fig. 5. MSD performance of the IPAPA adaptive filter for different values of the projection order M, averaged over 10 input speech signals. Parameters of the scenario are: $N = 512$, $K = 100$, SNR = 45 dB. The other IPAPA parameters are: $\kappa = 0.99$ and $\delta = 10^4$, $\epsilon = 10^{-4}$. The step size μ is adjusted for every M to achieve the best performance.

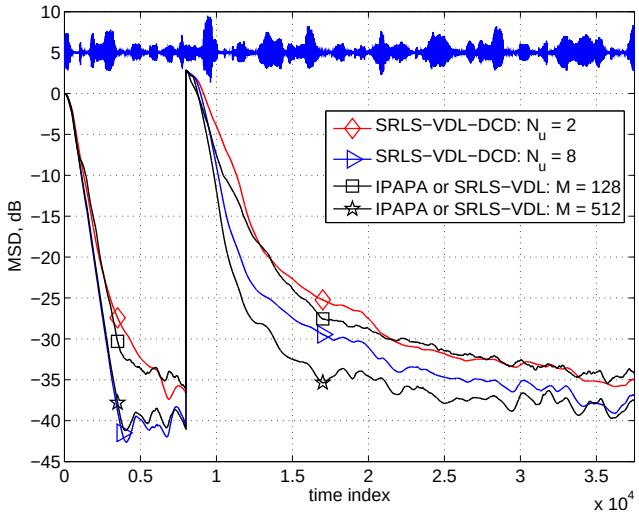

Fig. 6. MSD performance of the SRLS-VDL-DCD algorithm with different numbers of DCD updates N_u for one input speech realization. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other parameters are: $M = 512$ (apart from the IPAPA with $M = 128$), $\kappa = 0.99$ and $\delta = 10^4$, $\epsilon = 10^{-4}$, $\mu = 1.5 \cdot 10^{-3}$ for $M = 512$ and $\mu = 2.5 \cdot 10^{-3}$ for the IPAPA with $M = 128$.

complexity. In this figure, we show the performance of both the IPAPA and also the ideal implementation of the SRLS-VDL adaptive filter; indeed, the performance of both algorithms is exactly the same.

Fig. 7 presents similar results for a lower SNR: SNR = 25 dB. It can be seen that, again a small number of DCD iterations ($N_u = 8$) are enough to approach closely the IPAPA performance.

While Fig. 6 shows results for a single speech realization, Fig. 8 shows the MSD obtained by averaging the MSD performance for 10 input speech realizations, as in Fig. 5. The ensemble-averaged curves show the same behavior as in Fig. 6, confirming that a small number, $N_u = 16$, of DCD iterations is sufficient to approximate the performance of the exact SRLS-VDL adaptive filter, and $N_u = 8$ already achieves a good performance, compared to that of the IPAPA with $M = 128$.

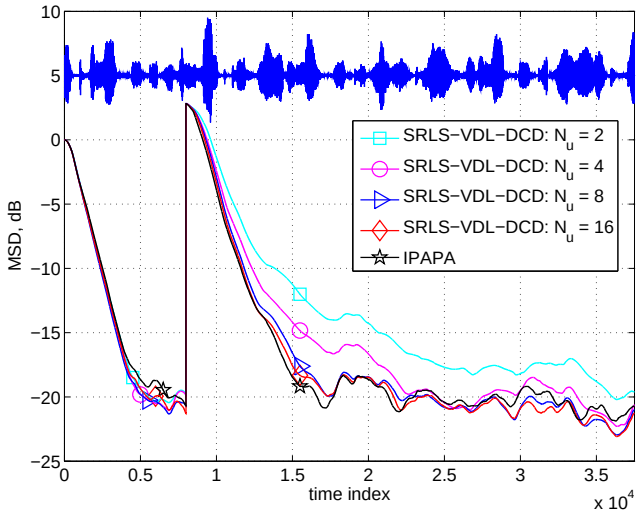

Fig. 7. MSD performance of the SRLS-VDL-DCD algorithm with different numbers of DCD updates N_u for one input speech realization. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 25 dB. The other parameters are: $M = 512$, $\kappa = 0.99$, $\delta = 10^4$, $\epsilon = 10^{-4}$, and $\mu = 6 \cdot 10^{-4}$.

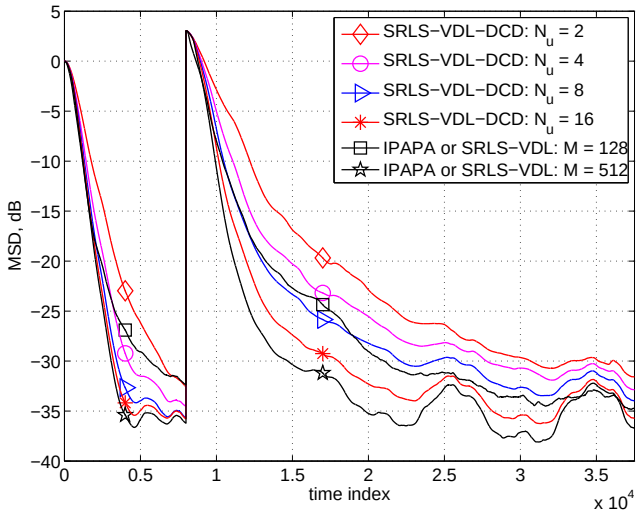

Fig. 8. MSD performance of the SRLS-VDL-DCD adaptive filter with different numbers of DCD updates N_u, averaged over 10 input speech signals. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other parameters are: $M = 512$ (apart from the IPAPA with $M = 128$), $\kappa = 0.99$, $\delta = 10^4$, $\epsilon = 10^{-4}$, $\mu = 1.5 \cdot 10^{-3}$ for $M = 512$ and $\mu = 2.5 \cdot 10^{-3}$ for the IPAPA with $M = 128$.

Fig. 9 shows the complexity (number of multiplications and additions) of the IPAPA, MIPAPA, and SRLS-VDL-DCD adaptive filters for the case of $N = 512$ and $N_u = 16$. It is seen that for the projection order $M = 2$, the number of multiplications of the SRLS implementation is close to that of the PAPA implementations. For $M \geq 3$, the PAPA implementations require more multiplications than the SRLS implementation. With large values of M, the complexity of the SRLS implementation is orders of magnitude less than that of PAPA.

Fig. 10 compares the performance of SRLS-VDL-DCD and IPAPA, choosing parameters so that their complexities will be similar (in fact, SRLS-VDL-DCD has slightly lower complexity, see Fig. 11). In this situation, Fig. 10 shows that with just two ($N_u = 2$) DCD iterations the SRLS-VDL-DCD adaptive filter with $M = 512$ already performs similarly or better than IPAPA with $M = 4$.

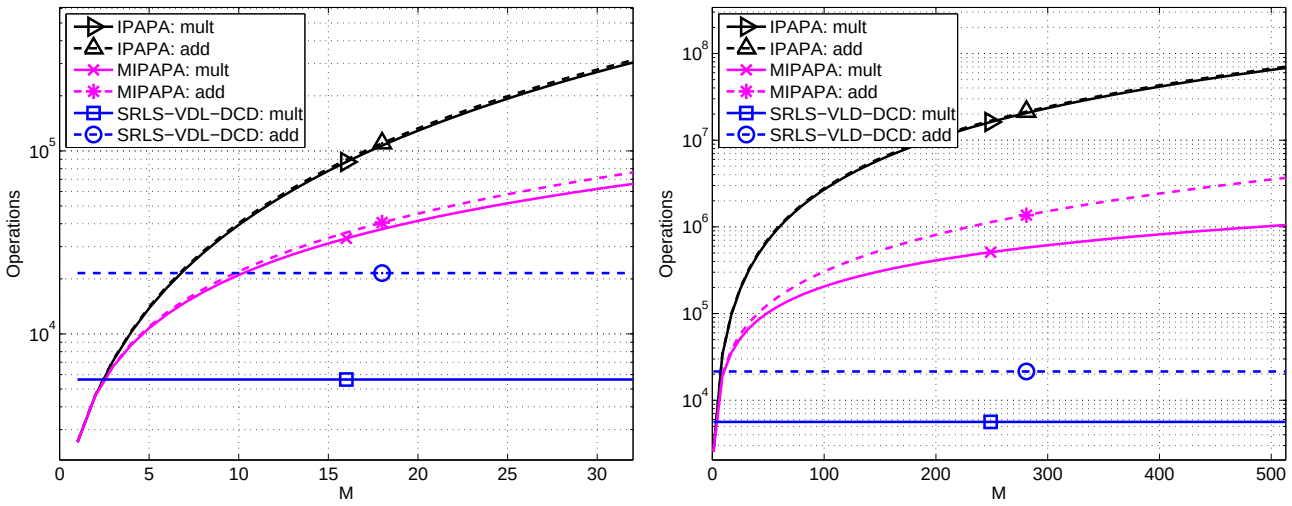

Fig. 9. Comparison of complexity (number of multiplications and additions) of the IPAPA, MIPAPA and SRLS-VDL-DCD adaptive filters against the projection order M; $N = 512$, $N_u = 16$.

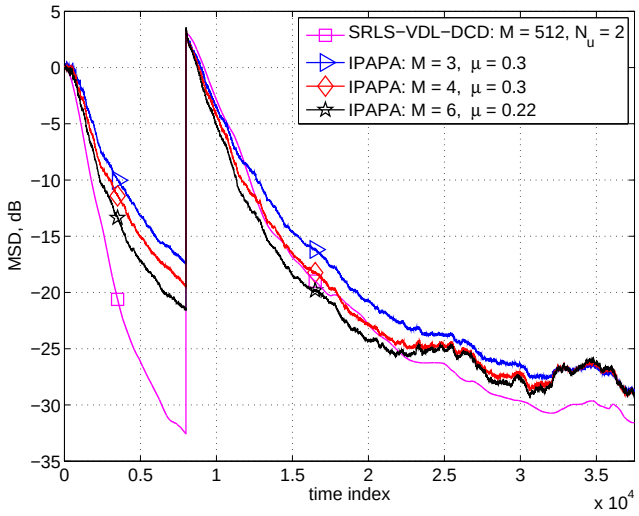

Fig. 10. MSD performance of the SRLS-VDL-DCD adaptive filter ($M = 512$, $N_u = 2$, $\mu = 1.5 \cdot 10^{-3}$) against the IPAPA adaptive filter with small values of M, averaged over 10 input speech signals. Parameters of the scenario: $N = 512$, $K = 100$, SNR = 45 dB. The other parameters are: $\kappa = 0.99$, $\delta = 10^4$, and $\epsilon = 10^{-4}$.

Therefore, the new algorithm provides more flexibility in designing adaptive filters. As our examples have shown, SRLS-VDL-DCD with a large value of M can be designed (by choosing N_u) to have the same performance, but with a smaller complexity even than IPAPA with small M. On the other hand, by increasing N_u, the performance of SRLS-VDL-DCD approaches that of IPAPA with large M, but now with a significant reduction in complexity.

VI. Conclusion

In this paper we studied the influence of the affine projection order in proportionate affine projection adaptive filters for sparse system identification, showing that, contrary to common practice, large projection orders (comparable to the length of the impulse response to be estimated) may result in better identification performance.

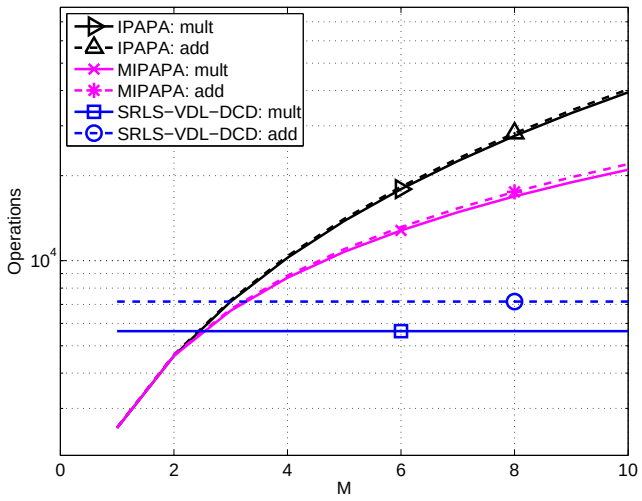

Fig. 11. Comparison of complexity (number of multiplications and additions) of the IPAPA, MIPAPA and SRLS-VDL-DCD adaptive filters against the projection order M; $N = 512$, $N_u = 2$.

We proceeded to propose new implementations for PAPAs that allow the use of large projection orders, while keeping the complexity low (in fact, even for small projection orders, the new algorithms can be designed to have similar performance and lower complexity compared to previous low-complexity PAPA implementations). The key to the low-cost algorithms is the connection we make between PAPA and sliding-window RLS with specific time- and tap-varying diagonal loading. Viewing PAPA as a sliding-window RLS allows one to find at each time instant a good initial condition to the low-cost DCD algorithm for solving a system of equations, so that only a small number of iterations is necessary to compute the filter estimates.

One drawback of using high projection order is the increased memory size for saving the input autocorrelation matrix. For standard PAPA implementations, the memory size is $\mathcal{O}(M^2)$, whereas in our case it is $\mathcal{O}(N^2)$. If memory is an issue, one would prefer to use the standard PAPA implementation with small M. However, if performance is more important and/or complexity is the limiting factor, the new implementation would be preferable.

APPENDIX: SRLS-VDL-DCD ADAPTIVE FILTER

Based on the SRLS-VDL recursion, we now derive a recursion for the SRLS-VDL-DCD adaptive filter, which can be approximately implemented with a complexity of $\mathcal{O}(N_u N)$ operations per time instant using DCD iterations.

We notice that, for every sample n, the current estimate $\tilde{\mathbf{h}}(n)$ can be found as a solution to the system of equations:

$$\mathbf{R}_W(n)\tilde{\mathbf{h}}(n) = \mathbf{g}(n), \tag{25}$$

where $\mathbf{g}(n) = \mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1)$. Assume that an approximate solution $\hat{\tilde{\mathbf{h}}}(n-1)$ to the system

$$\mathbf{R}_W(n-1)\tilde{\mathbf{h}}(n-1) = \mathbf{b}(n-1) + \mathbf{W}(n-1)\hat{\mathbf{h}}(n-2) \tag{26}$$

was found for the time instant $(n-1)$. We now want to use $\hat{\tilde{\mathbf{h}}}(n-1)$ as a *warm start* (initialization) for finding an approximate solution $\hat{\tilde{\mathbf{h}}}(n)$ for time instant n as

$$\hat{\tilde{\mathbf{h}}}(n) = \hat{\tilde{\mathbf{h}}}(n-1) + \Delta\tilde{\mathbf{h}}(n). \tag{27}$$

To find the vector $\Delta\tilde{\mathbf{h}}(n)$, we need to solve a system of equations

$$\mathbf{R}_W(n)\Delta\tilde{\mathbf{h}}(n) = \mathbf{r}(n), \tag{28}$$

where $\mathbf{r}(n) = \mathbf{g}(n) - \mathbf{R}_W(n)\hat{\tilde{\mathbf{h}}}(n-1)$. Assuming that there exist $\Delta\mathbf{g}(n)$ and $\Delta\mathbf{R}_W(n)$ so that

$$\mathbf{g}(n) = \mathbf{g}(n-1) + \Delta\mathbf{g}(n), \tag{29}$$

$$\mathbf{R}_W(n) = \mathbf{R}_W(n-1) + \Delta\mathbf{R}_W(n), \tag{30}$$

we can represent the vector $\mathbf{r}(n)$ as

$$\begin{aligned}\mathbf{r}(n) &= \mathbf{g}(n-1) + \Delta\mathbf{g}(n) \\ &- \mathbf{R}_W(n-1)\hat{\tilde{\mathbf{h}}}(n-1) - \Delta\mathbf{R}_W(n)\hat{\tilde{\mathbf{h}}}(n-1) \\ &= \mathbf{r}_r(n-1) + \Delta\mathbf{g}(n) - \Delta\mathbf{R}_W(n)\hat{\tilde{\mathbf{h}}}(n-1),\end{aligned} \tag{31}$$

where $\mathbf{r}_r(n-1)$ denotes a residual vector given by

$$\mathbf{r}_r(n-1) = \mathbf{g}(n-1) - \mathbf{R}_W(n-1)\hat{\tilde{\mathbf{h}}}(n-1). \tag{32}$$

The idea of a low-complexity implementation of the adaptive algorithm is in finding an approximate solution to the system in (28). This can be done using a few (N_u) DCD iterations, each of complexity $\mathcal{O}(N)$. If the other steps in the algorithm can also be done with a complexity of $\mathcal{O}(N)$, then the whole algorithm complexity would be of $\mathcal{O}(N)$. However, obtaining the vector $\mathbf{r}(n)$ in (31) involves a matrix-vector multiplication, which in general is of complexity $\mathcal{O}(N^2)$, and computation of other vectors that if implemented directly would also involve significant complexity. Below we derive recursions which allow reducing the complexity.

We first notice that the vector $\mathbf{r}_r(n-1)$ is a by-product calculated in the DCD algorithm, and can be considered already available.

We now find a recursion for computation of $\Delta\mathbf{g}(n)$. Denoting $\Delta\mathbf{b}(n) = \mathbf{b}(n) - \mathbf{b}(n-1)$, we can write

$$\Delta\mathbf{g}(n) = \Delta\mathbf{b}(n) + \mathbf{W}(n)\hat{\mathbf{h}}(n-1) - \mathbf{W}(n-1)\hat{\mathbf{h}}(n-2). \tag{33}$$

From $\mathbf{b}(n) = \mathbf{X}(n)\mathbf{d}(n)$ we obtain that

$$\Delta\mathbf{b}(n) = d(n)\mathbf{x}(n) - d(n-M)\mathbf{x}(n-M). \tag{34}$$

Taking into account that $\mathbf{W}(n)$ is a diagonal matrix, we can see that computation of

$$\begin{aligned} \Delta\mathbf{g}(n) &= d(n)\mathbf{x}(n) - d(n-M)\mathbf{x}(n-M) \\ &+ \mathbf{W}(n)\hat{\mathbf{h}}(n-1) - \mathbf{W}(n-1)\hat{\mathbf{h}}(n-2) \end{aligned} \tag{35}$$

requires a complexity $\mathcal{O}(N)$.

The term $\Delta\mathbf{R}_W(n)\hat{\tilde{\mathbf{h}}}(n-1)$ in (31) can be written as

$$\Delta\mathbf{R}_W(n)\hat{\tilde{\mathbf{h}}}(n-1) = \left[\Delta\mathbf{R}(n) + \Delta\mathbf{W}(n)\right]\hat{\tilde{\mathbf{h}}}(n-1), \tag{36}$$

where $\Delta\mathbf{W}(n) = \mathbf{W}(n) - \mathbf{W}(n-1)$. We notice that

$$\Delta\mathbf{R}(n) = \mathbf{x}(n)\mathbf{x}^T(n) - \mathbf{x}(n-M)\mathbf{x}^T(n-M) \tag{37}$$

and

$$\Delta\mathbf{R}(n)\hat{\tilde{\mathbf{h}}}(n-1) = \tilde{y}(n)\mathbf{x}(n) - \tilde{y}_M(n)\mathbf{x}(n-M), \tag{38}$$

where

$$\tilde{y}(n) = \mathbf{x}^T(n)\hat{\tilde{\mathbf{h}}}(n-1), \tag{39}$$

$$\tilde{y}_M(n) = \mathbf{x}^T(n-M)\hat{\tilde{\mathbf{h}}}(n-1). \tag{40}$$

We further denote

$$\tilde{e}(n) = d(n) - \tilde{y}(n), \tag{41}$$

$$\tilde{e}_M(n) = d(n-M) - \tilde{y}_M(n), \tag{42}$$

and finally obtain the following recursion for computation of the right-hand part of the system of equations in (28):

$$\begin{aligned} \mathbf{r}(n) &= \mathbf{r}_r(n-1) + \tilde{e}(n)\mathbf{x}(n) - \tilde{e}_M\mathbf{x}(n-M) \\ &+ \mathbf{W}(n)\hat{\mathbf{h}}(n-1) - \mathbf{W}(n-1)\hat{\mathbf{h}}(n-2) \\ &- \Delta\mathbf{W}(n)\hat{\tilde{\mathbf{h}}}(n-1). \end{aligned} \tag{43}$$

Summarizing the above recursions, we arrive at the SRLS-VDL-DCD adaptive filter that allows a low-complexity implementation of the SRLS-VDL algorithms and, consequently, the low-complexity PAPA implementation. The structure of the SRLS-VDL-DCD adaptive filter is presented in Table III.

REFERENCES

[1] B. D. Rao and B. Song, "Adaptive filtering algorithms for promoting sparsity," in *Proceedings IEEE Int. Conf. Acoustics, Speech and Signal Processing, ICASSP'2003*, 2003, pp. VI–361–364.

[2] B. Babadi, N. Kalouptsidis, and V. Tarokh, "SPARLS: The sparse RLS algorithm," *IEEE Trans. Signal Processing*, vol. 58, no. 8, pp. 4013–4025, 2010.

[3] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada, "A sparse adaptive filtering using time-varying soft-thresholding techniques," in *Proceedings IEEE Int. Conf. Acoustic, Speech and Signal Processing, ICASSP'2010*, 2010, pp. 3734–3737.

[4] C. Paleologu, J. Benesty, and S. Ciochina, "Sparse adaptive filters for echo cancellation," *Synthesis Lectures on Speech and Audio Processing*, vol. 6, no. 1, pp. 1–124, 2010.

[5] E. M. Eksioglu and A. K. Tanc, "RLS algorithm with convex regularization," *IEEE Signal Processing Letters*, vol. 18, no. 8, pp. 470–473, 2011.

[6] Y. V. Zakharov and V. H. Nascimento, "DCD-RLS adaptive filters with penalties for sparse identification," *IEEE Transactions on Signal Processing*, vol. 61, no. 12, pp. 3198–3213, June 2013.

[7] T. Gansler, J. Benesty, S. L. Gay, and M. Sondhi, "A robust proportionate affine projection algorithm for network echo cancellation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Istanbul, Turkey*, 2000, vol. 2, pp. 793–796.

[8] C. Paleologu, S. Ciochina, and J. Benesty, "An efficient proportionate affine projection algorithm for echo cancellation," *IEEE Signal Processing Letters*, vol. 17, no. 2, pp. 165–168, 2010.

[9] C. Stanciu, C. Anghel, C. Paleologu, J. Benesty, F. Albu, and S. Ciochina, "A proportionate affine projection algorithm using dichotomous coordinate descent iterations," in *10th International Symposium on Signals, Circuits and Systems (ISSCS)*, 2011, pp. 1–4.

[10] F. Albu, "A proportionate affine projection algorithm using fast recursive filtering and dichotomous coordinate descent iterations," *Proc. of SPAMEC 2011*, pp. 93–96, 2011.

[11] J. Yang and G. E. Sobelman, "Efficient μ-law improved proportionate affine projection algorithm for echo cancellation," *Electronics Letters*, vol. 47, no. 2, pp. 73–74, 2011.

[12] F. Albu, "New proportionate affine projection algorithm," in *ASME 2012 Noise Control and Acoustics Division Conference at InterNoise 2012*. American Society of Mechanical Engineers, 2012, pp. 35–41.

[13] L. Xiao, Y. Wang, P. Zhang, M. Wu, and J. Yang, "Variable regularisation efficient μ-law improved proportionate affine projection algorithm for sparse system identification," *Electronics Letters*, vol. 48, no. 3, pp. 182–184, 2012.

[14] F. Albu and H. K. Kwan, "Memory improved proportionate affine projection sign algorithm," *Electronics Letters*, vol. 48, no. 20, pp. 1279–1281, 2012.

[15] H. Zhao, Y. Yu, S. Gao, X. Zeng, and Z. He, "Memory proportionate APA with individual activation factors for acoustic echo cancellation," *IEEE Transactions on Audio, Speech, and Language Processing*, pp. 1–9, April 2014, online, early access.

[16] Y. V. Zakharov and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem," *Electonics Letters*, vol. 40, no. 9, pp. 567–569, 2004.

[17] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 56, no. 11, pp. 2425–2438, 2009.

[18] C. Stanciu, C. Anghel, C. Paleologu, J. Benesty, F. Albu, and S. Ciochina, "FPGA implementation of an efficient proportionate affine projection algorithm for echo cancellation," in *19th European Signal Processing Conference (EUSIPCO 2011), Barcelona, Spain*, 29 Aug. – 2 Sept., 2011, pp. 1284–1288.

[19] C. Paleologu, J. Benesty, and S. Ciochina, "Study of the general Kalman filter for echo cancellation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 8, pp. 1539–1549, Aug. 2013.

[20] J. Benesty, C. Paleologu, and S. Ciochina, "Proportionate adaptive filters from a basis pursuit perspective," *IEEE Signal Processing Letters*, vol. 17, no. 12, pp. 985–988, 2010.

[21] Y. Zakharov and F. Albu, "Coordinate descent iterations in fast affine projection algorithm," *IEEE Signal Processing Letters*, vol. 12, no. 5, pp. 353–356, May 2005.

[22] Y. Zakharov, G. White, and J. Liu, "Low complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 3150–3161, July 2008.

[23] J. Benesty and S. L. Gay, "An improved PNLMS algorithm," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando, FL, USA*, 2002, vol. 2, pp. 1881–1884.

[24] A. H. Sayed, *Fundamentals of Adaptive Filtering*, Hoboken, N. J.: Wiley, 2003.

[25] M. Montazeri and P. Duhamel, "A set of algorithms linking NLMS and block RLS algorithms," *IEEE Transactions on Signal Processing*, vol. 43, no. 2, pp. 444–453, 1995.

[26] J. H. Husoy and M. S. E. Abadi, "Unified approach to adaptive filters and their performance," *IET Signal Processing*, vol. 2, no. 2, pp. 97–109, 2008.

[27] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[28] J. Liu and Y. Zakharov, "Low complexity dynamically regularised RLS algorithm," *Electronics Letters*, vol. 44, no. 14, pp. 886–885, 2008.

[29] Y. V. Zakharov and V. H. Nascimento, "Sparse RLS adaptive filter with diagonal loading," in *46th Asilomar Conference on Signals, Systems and Computers*, 2012.

[30] Vítor H. Nascimento and Magno T. M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing:*, Rama Chellappa and Sergios Theodoridis, Eds., vol. 1, Signal Processing Theory and Machine Learning, pp. 619—761. Chennai: Academic Press, 2014.

PLACE
PHOTO
HERE

Yuriy V. Zakharov (M'01) received the M.Sc. and Ph.D. degrees in electrical engineering from the Moscow Power Engineering Institute, Moscow, Russia, in 1977 and 1983, respectively. From 1977 to 1983, he was an Engineer with the Special Design Agency, Moscow Power Engineering Institute. From 1983 to 1999, he was the Head of Laboratory at the N. N. Andreev Acoustics Institute, Moscow. From 1994 to 1999 he was with Nortel as a DSP Group Leader. Since 1999, he has been with the Communications Research Group, University of York, York, U.K., where he is currently a Reader. His interests include signal processing and communications.

Vítor H. Nascimento (M'01) was born in São Paulo, Brazil. He obtained the B.S. and M.S. degrees in Electrical Engineering from the University of São Paulo in 1989 and 1992, respectively, and the Ph.D. degree from the University of California, Los Angeles, in 1999. From 1990 to 1994 he was a Lecturer at the Univ. of São Paulo, and in 1999 he joined the faculty at the same school, where he is now an Associate Professor. One of his papers received the 2002 IEEE SPS Best Paper Award. He served as an Associate Editor for the IEEE Signal Processing Letters from 2003 to 2005, for the IEEE Transactions on Signal Processing from 2005 to 2008 and for the EURASIP Journal on Advances in Signal Processing from 2006 to 2009, and was a member of the IEEE-SPS Signal Processing Theory and Methods Technical Committee from 2007 to 2012. Since 2010 he is the chair of the São Paulo SPS Chapter. His research interests include signal processing theory and applications, robust and nonlinear estimation, and applied linear algebra.