A LOW-COMPLEXITY RLS-DCD ALGORITHM FOR VOLTERRA SYSTEM IDENTIFICATION

*Raffaello Claser and Vítor H. Nascimento**
Univ. of São Paulo, Brazil

Yuriy V. Zakharov
University of York, UK

ABSTRACT

Adaptive filters for Volterra system identification must deal with two difficulties: large filter length M (resulting in high computational complexity and low convergence rate) and high correlation in the input sequence. The second problem is minimized by using the recursive least-squares algorithm (RLS), however, its large computation complexity ($\mathcal{O}(M^2)$) might be prohibitive in some applications. We propose here a low-complexity RLS algorithm, based on the dichotomous coordinate descent algorithm (DCD), showing that in some situations the computational complexity is reduced to $\mathcal{O}(M)$. The new algorithm is compared to the standard RLS, normalized least-mean squares (NLMS) and affine projections (AP) algorithms.

I. INTRODUCTION

Linear models and methods have played a key role in engineering and signal processing because of their inherent simplicity. However, there are numerous practical situations in which nonlinear processing is needed, either because the nonlinearities in the system under study are too important to be disregarded, or because the desired behavior cannot be achieved with a linear system [1]. In the case of system identification, an important class of nonlinear models are linear-in-the-parameters nonlinear models, in which the input-output relation is nonlinear, but the estimation problem is essentially linear. Popular examples are polynomial filters, and in particular Volterra filters [1]. The Volterra system model is similar to a Taylor series, but with the ability to capture "memory" effects. In [1], the LMS second-order adaptive Volterra filter has been introduced using truncated Volterra series expansions.

Truncated Volterra series models have become popular in nonlinear adaptive filtering applications, such as echo cancellation [2], [3], channel equalization [4], system identification, detection and estimation and physiological system modeling [5]. Several adaptive filters using Volterra models have been proposed, based on the least-mean squares (LMS), recursive least-squares (RLS), affine projections (AP) algorithms, among others [1], [3]–[5].

A problem related to Volterra system identification is the large number of parameters to be estimated, which results in large computational complexity. In fact, a full second-order Volterra kernel with memory depth of N samples contains $N(N+1)/2$ terms [4], see Section II. This means that adaptive filters for identification of Volterra models should have low computational complexity, preferably linear on the filter length M (the number of parameters to be estimated). This is the case of the LMS and normalized LMS (NLMS) algorithms, but their convergence rate is slow, due to the correlation in the Volterra kernel [4]. The RLS algorithm can solve this problem, but at the cost of a computational complexity that grows quadratically with M. Fast $\mathcal{O}(M)$ versions of the RLS algorithm, such as lattice RLS [6] cannot be used in this case, since they require time-shifted regressors, which is not the case for Volterra models. Fast multichannel QRD-RLS filters are good options, but require a large number of divisions and square-roots [7], [8]. Another option is the AP algorithm (APA) [4], [6], whose computational complexity grows linearly with M, with a convergence rate between those of the LMS and RLS algorithms.

In this paper we describe a new alternative, a low-complexity version of the RLS algorithm suitable for Volterra system identification, extending the RLS-DCD algorithm, a low-complexity version of RLS, based on the dichotomous coordinate descent (DCD) algorithm. RLS-DCD was originally proposed in [9], and later was generalized to deal with widely-linear models in [10]. The RLS-DCD algorithm of [9], [10] has complexity $\mathcal{O}(M)$ when the regressor has a time-shift structure. The algorithm in [10] extends this result to widely-linear estimation, in which the regressor is composed of two separate delay lines. We show how a generalized version of [10] can be applied to lower the complexity of Volterra system identification, organizing the Volterra kernel in a number of separate delay lines. The reduction in complexity results from exploring some block-matrix symmetries in the regressor autocorrelation matrix $\mathbf{R}(i)$. For a full second-order Volterra kernel, the resulting complexity is $\mathcal{O}(M^{3/2})$, but if the number of cross-terms is limited, as suggested in [4], the complexity becomes linear in the filter length M. For simplicity, this paper concentrates on the second-order Volterra kernel, but the idea can easily be extended to higher-order kernels.

This paper is organized as follows: in Section II we give a brief description of Volterra series. Section III describes the DCD and RLS-DCD algorithms and how to reduce the computational cost of updating the matrix $\mathbf{R}(i)$ in

*The work of Y.V. Zakharov and V.H. Nascimento was partly supported by York-FAPESP grant no. 2014/50765-6. In addition, V.H. Nascimento also received support from FAPESP grant 2014/04256-2 and CNPq grant 306268/2014-0. Contact: rclaser1@gmail.com, vitor@lps.usp.br.

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

RLS-DCD for second-order Volterra kernels. Section IV exemplifies the performance of our new technique under different conditions, and compares the new algorithm with standard RLS, NLMS and affine projection algorithms. Lastly, Section V concludes the paper.

II. THE TRUNCATED VOLTERRA SERIES

For simplicity of presentation, we restrict our discussion to truncated second-order Volterra system models. However, the concepts discussed in this section are valid for higher-order Volterra systems with finite memory [11]. As we

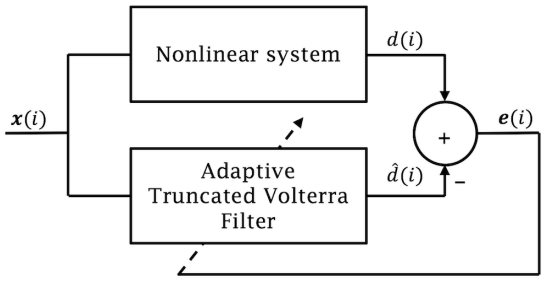


Fig. 1. Block-diagram of an Adaptive Truncated Volterra Filter.

can see in (1) and Figure 1, the adaptive filter tries to approximate the desired signal $d(i)$ using a first- and second-order truncated Volterra series expansion in the N most recent samples, as

$$\begin{aligned}\hat{d}(i) = &\sum_{m_1=0}^{N-1} w_1(m_1;i)x(i-m_1) \\ &+ \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} w_2(m_1,m_2;i)x(i-m_1)x(i-m_2),\end{aligned} \tag{1}$$

where $w_1(m_1;i)$ and $w_2(m_1,m_2;i)$ are linear and quadratic coefficients, respectively, of the adaptive filter at time i (a bias term could also be included). The adaptive filter iteratively updates its coefficients at each timestep so as to minimize the mean squared error $\mathrm{E}\{e^2(i)\}$, where $\mathrm{E}\{\cdot\}$ is the expectation operator and $e(i)$ is the error signal

$$e(i) = d(i) - \hat{d}(i). \tag{2}$$

In this paper, the regressor vector $\mathbf{s}(i)$ of the Volterra filter is organized in sets of delay lines defined as: $\mathbf{s}_L(i)$ (Linear delay line of length N), $\mathbf{s}_Q(i)$ (Quadratic delay line of length N) and $\mathbf{s}_{C_k}(i)$ (delay-k Cross-Product delay line of length $N-k$) i.e.,

$$\mathbf{s}(i) = \begin{bmatrix}\mathbf{s}_L(i) & \mathbf{s}_Q(i) & \mathbf{s}_{C_1}(i) & \dots & \mathbf{s}_{C_K}(i)\end{bmatrix}^T, \tag{3}$$

where $1 \le K \le N-1$ denotes the maximum delay included in the cross-product terms. The tap-delay lines are given by

$$\mathbf{s}_L(i) = \begin{bmatrix}x(i) & x(i-1) & \dots & x_L(i-N+1)\end{bmatrix}^T \tag{4}$$

$$\mathbf{s}_Q(i) = \begin{bmatrix}x^2(i) & x^2(i-1) & \dots & x^2(i-N+1)\end{bmatrix}^T \tag{5}$$

$$\mathbf{s}_{C_k}(i) = \begin{bmatrix}x(i)x(i-k) & x(i-1)x(i-1-k) & \dots\end{bmatrix}^T, \tag{6}$$

where the length of the last delay line is $N-k$. This particular ordering of the regressor vector $\boldsymbol{s}(i)$ is used in the next section to derive a low-complexity RLS-DCD algorithm for Volterra filters.

III. THE RLS-DCD ALGORITHM

The RLS algorithm computes a weight vector $\boldsymbol{w}(i)$ by iteratively solving the normal equations [6], [12]

$$\mathbf{R}(i)\mathbf{w}(i) = \boldsymbol{p}(i) \tag{7}$$

where

$$\mathbf{R}(i) = \sum_{i=0}^{j} \lambda^{i-j}\mathbf{s}(j)\mathbf{s}^T(j) = \lambda\mathbf{R}(i-1) + \mathbf{s}(i)\mathbf{s}^T(i) \tag{8}$$

is the $M \times M$ autocorrelation matrix, and λ is the forgetting factor. The cross-correlation vector $\boldsymbol{p}(i)$ is

$$\boldsymbol{p}(i) = \sum_{j=0}^{i} \lambda^{i-j} d(j)\mathbf{s}(j) = \lambda\boldsymbol{p}(i-1) + d(i)\mathbf{s}(i) \tag{9}$$

For a regressor vector such as in (3), the standard RLS algorithm uses the matrix inversion lemma to find a recursion to $\boldsymbol{R}^{-1}(i)$, and thus solves (7). This requires $\mathcal{O}(M^2)$ multiplications and additions at each time instant [6], [12]. Low-complexity ($\mathcal{O}(M)$) versions of RLS, such as the fast transversal filter (FTF) [6], lattice RLS [6], [13] and RLS-DCD [9], [12] require that the regressor vector be composed of a single delay line and thus cannot be applied. This constraint was eased in [10], in which a modified version of the RLS-DCD algorithm was developed for the case of widely-linear estimation, in which the regressor is composed of two delay lines. Here we generalize the result of [10] to the case of multiple delay lines, and show how this can be used to obtain a low-complexity RLS-DCD algorithm suitable for Volterra system identification.

Contrary to standard RLS, the RLS-DCD algorithm uses a low-cost iterative algorithm to find an approximate solution $\boldsymbol{\Delta w}(i)$ to the modified problem

$$\boldsymbol{R}(i)\boldsymbol{\Delta w}(i) = \boldsymbol{\beta}(i), \tag{10}$$

where $\boldsymbol{\beta}(i)$ is computed through the recursion

$$\begin{aligned}&\hat{d}(i) = \mathbf{s}^T(i)\hat{\mathbf{w}}(i-1), && \boldsymbol{\beta}(i) = \lambda\mathbf{r}(i-1) + e(i)\mathbf{s}(i), \\ &\hat{\mathbf{w}}(i) = \hat{\mathbf{w}}(i-1) + \boldsymbol{\Delta}\hat{\mathbf{w}}(i) && \boldsymbol{r}(i) = \boldsymbol{\beta}(i) - \boldsymbol{R}(i)\boldsymbol{\Delta w}(i),\end{aligned}$$

and $\mathbf{r}(i)$ is the residue at time i. Since in an adaptive filter the weight update $\boldsymbol{\Delta w}(i)$ is expected to be small, a sufficiently good approximation to (10) can be obtained through an iterative algorithm, using just a few iterations. We follow [9], [14] and choose the DCD algorithm to solve (10), obtaining an $\mathcal{O}(M)$ algorithm. DCD is an iterative method for solving least-squares problems, designed to avoid multiplications and divisions (which are replaced by bit-shifts) [12], [15].

We use here the "leading" DCD algorithm, summarized in Table I. M_b represents the precision in the solution (it corresponds to the number of bits in the solution if

the algorithm is implemented using fixed-point arithmetic). The constant H should be chosen as a power of two, in which case all multiplications and divisions reduce to bit shifts, thus allowing simple implementations in hardware [14]. N_u is the maximum number of vector operations in the algorithm, which for adaptive filtering applications can be chosen as $N_u \ll M$. The complexity of the leading DCD algorithm is upper limited by $(2M+1)N_u + M_b$ additions. This corresponds to a worst case scenario when the algorithm makes use of all N_u updates and the condition at step 3 in Table I is never satisfied [9].

Table I. Leading DCD algorithm

step	Equation
	Initialization: $\boldsymbol{\Delta}\mathbf{w}(i)=0$, $\mathbf{r}=\boldsymbol{\beta}$, $h=H$, $m=1$
	for $i=1,\ldots,N_u$
1	$n = arg\ max_{p=1,\ldots,M}\{\lvert r_p\rvert\}$
2	$m=m+1$, $h=h/2$
3	if $m > M_b$, algorithm stops
4	if $\lvert r_n\rvert \le (h/2)\mathbf{R}_{n,n}$, then go to step 2
5	$\boldsymbol{\Delta}\mathbf{w}(i) = \boldsymbol{\Delta}\mathbf{w}(i) + \text{sign}(r_n)h$
6	$\mathbf{r}(i) = \mathbf{r}(i) - \text{sign}(r_n)h\mathbf{R}^{(1:M,1)}$

The DCD algorithm thus allows us to solve (10) in $\mathcal{O}(M)$ operations. The difficulty remains the update of $\boldsymbol{R}(i)$: if we use (8) directly, the number of operations required is $\mathcal{O}(M^2)$. When the regressor vector $\boldsymbol{s}(i)$ consists of a single delay line, [9] was able to use the structure in $\boldsymbol{R}(i)$ to reduce the number of operations to $\mathcal{O}(M)$; later [10] extended this result to the case of $\boldsymbol{s}(i)$ consisting of two delay lines.

In the case of Volterra system identification, $\boldsymbol{s}(i)$ in (3) consists of $K+2$ delay lines, where $K \le N-1$ is a limit to the number of cross-terms considered. With a memory depth of N, we have N elements for each $\mathbf{s}_L(i)$ and $\mathbf{s}_Q(i)$, and $N-k$ elements for $\mathbf{s}_{C_k}(i)$, so that the total number of coefficients M is

$$M = 2N + \frac{(2N-1-K)K}{2}, \quad 1 \le K \le N-1. \quad (11)$$

Taking into account the structure of $\boldsymbol{s}(i)$, $\mathbf{R}(i)$ can be partitioned as

$$\mathbf{R}(i) = \begin{bmatrix} \mathbf{R}_L(i) & \mathbf{R}_{LQ}(i) & \ldots & \boldsymbol{R}_{LC_K}(i) \\ \mathbf{R}_{QL}(i) & \mathbf{R}_Q(i) & \ldots & \boldsymbol{R}_{QC_K}(i) \\ \mathbf{R}_{C_1L}(i) & \mathbf{R}_{C_1Q}(i) & \ldots & \boldsymbol{R}_{C_1C_K}(i) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{C_KL}(i) & \mathbf{R}_{C_KQ}(i) & \ldots & \boldsymbol{R}_{C_K}(i) \end{bmatrix} \quad (12)$$

where for each block $\boldsymbol{R}_{ab}(i)$, $a,b \in \{L,Q,C_1,\ldots,C_K\}$ we have a recursion (note that we write $\boldsymbol{R}_{aa}(ii)$ as $\boldsymbol{R}_a(i)$ to simplify notation)

$$\boldsymbol{R}_{ab}(i) = \lambda \boldsymbol{R}_{ab}(i-1) + \boldsymbol{s}_a(i)\boldsymbol{s}_b^T(i). \quad (13)$$

For each block on the main diagonal, instead of (13), we can use the low-cost update proposed in [9],

$$\mathbf{R}_a(i) = \begin{bmatrix} r_a(i) & \rho_a^T(i) \\ \rho_a(i) & \mathbf{R}_a^{(1:\alpha-1,1:\alpha-1)}(i-1) \end{bmatrix}, \quad (14)$$

where the notation $\mathbf{R}_a^{(1:\alpha-1,1:\alpha-1)}(i-1)$ stands for a matrix with the elements of $\mathbf{R}_a(i-1)$ from rows 1 through $\alpha-1$ and from columns 1 through $\alpha-1$, and α is an integer equal to the dimension of $\boldsymbol{s}_a(i)$. $r_a(i)$ is a real number and $\rho_a(i)$ is an $(\alpha-1)\times 1$ vector. From (14), we note that we only need to update the first column of $\mathbf{R}_a(i)$, since we already have $\mathbf{R}_a^{(1:\alpha-1,1:\alpha-1)}(i-1)$ from the previous iteration and $\mathbf{R}_a(i)$ is symmetric. This is done as follows

$$\mathbf{R}_a^{(1:\alpha,1)}(i) = \begin{bmatrix} r_a(i) \\ \rho_a(i) \end{bmatrix} = \lambda \mathbf{R}_a^{(1:\alpha,1)}(i-1) + \boldsymbol{s}_a^{(1)}(i)\mathbf{s}_a(i), \quad (15)$$

where $\boldsymbol{s}_a^{(1)}(i)$ represents the first element of vector $\boldsymbol{s}_a(i)$. Similarly, for the off-diagonal blocks we can use the recursion [10]

$$\mathbf{R}_{ab}(i) = \begin{bmatrix} r_{ab}(i) & \rho_{ba}^T(i) \\ \rho_{ab}(i) & \mathbf{R}_{ab}^{(1:\alpha_1-1,1:\alpha_2-1)}(i-1) \end{bmatrix}, \quad (16)$$

where $a \neq b$, and α_1 and α_2 are the lengths of $\mathbf{s}_a(i)$ and $\mathbf{s}_b(i)$, respectively. In general, $\mathbf{R}_{ab}$ is not symmetric, so $\rho_{ab} \neq \rho_{ba}$:

$$\mathbf{R}_{ab}^{(1:\alpha_2,1)}(i) = \begin{bmatrix} r_{ab}(i) \\ \rho_{ab}(i) \end{bmatrix} = \lambda \mathbf{R}_{ab}^{(1:\alpha_2,1)}(i-1) + \boldsymbol{s}_a^{(1)}(i)\mathbf{s}_b(i), \quad (17)$$

$$\rho_{ba}(i) = \lambda \rho_{ba}(i-1) + \boldsymbol{s}_b^{(1)}(i)\mathbf{s}_a^{(2:\alpha_1,1)}(i). \quad (18)$$

These recursions reduce the total number of operations in the update of the $\boldsymbol{R}(i)$ matrix, so that the total computational cost for the RLS-DCD algorithm applied to Volterra system identification reduces to the values given in Table II (we present the number of multiplications for two cases, when $0<\lambda<1$ is any value, or when $\lambda = 1-2^{-m}$ for an integer $m>0$). Table III summarizes the algorithm. The costs for RLS, NLMS and APA (order γ) are from [6].

Table II. Computational cost (RLS, NLMS, APA from [6])

Algorithm	+	×	÷
RLS	M^2+3M	M^2+5M+1	1
NLMS	$3M$	$3M+1$	1
APA of order γ	$(\gamma^2+2\gamma)M$ $+\gamma^3+\gamma^2$	$(\gamma^2+2\gamma)M$ $+\gamma^3+\gamma$	γ
RLS-DCD (M from (11))	$(N+1)K^2-K^3/2$ $-K/2+4M+$ $(2M+1)N_u+M_b$	$(2N+2)K^2$ $-K^3$ $-K+4M$	-
RLS-DCD ($\lambda=1-2^{-m}$)	$(2N+2)K^2-K^3$ $-K+5M+$ $(2M+1)N_u+M_b$	$(N+1)K^2$ $-K^3/2$ $-1/2K+3M$	-

Note from Table II and (11) that if we increase M by increasing the value of N, but keeping fixed the value of K, then the computational cost of DCD-RLS grows linearly with M. This situation is important, since in practice the value of K is kept small to avoid having a filter length that is too large, as explained in [4].

On the other hand, for a full second-order Volterra kernel, such that $K=N-1$, (11) implies that $M \approx N^2/2$ for large N, and from Table II we conclude that the computational complexity of RLS-DCD is $\mathcal{O}(M^{3/2})$, still much lower than the $\mathcal{O}(M^2)$ of standard RLS.

Table III. RLS-DCD algorithm for Volterra system identification

step	Equation
	Initialization:
	$\Delta\hat{\mathbf{w}}(0) = 0_{M\times 1}$, $\mathbf{r}(0) = 0_{M\times 1}$,
	Each block-matrix of main diagonal of $\mathbf{R}(0) = \delta\mathbf{I}$,
	Each block-matrix of upper diagonal of $\mathbf{R}(0) = 0$
	for $i = 1, 2, \ldots$
	for $a = L, Q, C_1, \ \ldots$
1	$\mathbf{R}_a^{(1:\alpha,1)}(i) = \lambda\mathbf{R}_a^{(1:\alpha,1)}(i-1) + s_a(i)\mathbf{s}_a(i)$
	end
	for $ab = LQ, LC_1, LC_2, \ \ldots$ (upper diagonal of $\mathbf{R}(i)$)
2	$\mathbf{R}_{ab}^{(1:\alpha_2,1)}(i) = \lambda\mathbf{R}_{ab}^{(1:\alpha_2,1)}(i-1) + s_a(i)\mathbf{s}_b(i)$
3	$\mathbf{R}_{ab}^{(2:\alpha_1,1)}(i) = \lambda\mathbf{R}_{ab}^{(2:\alpha_1,1)}(i-1) + s_b(i)\mathbf{s}_a^{(2:\alpha_1,1)}(i)$
	end
4	$\mathbf{R}_L, \mathbf{R}_Q, \mathbf{R}_{C_1}, \ldots \Rightarrow \ \mathbf{R}(i)$
5	$y(i) = \mathbf{s}^T(i)\hat{\mathbf{w}}(i-1)$
6	$e(i) = d(i) - y(i)$
7	$\boldsymbol{\beta}(i) = \lambda\mathbf{r}(i-1) + e(i)\mathbf{s}(i)$
8	$\mathbf{R}(i)\Delta\mathbf{w}(i) = \boldsymbol{\beta}(i) \Rightarrow \ \Delta\hat{\mathbf{w}}(i), \mathbf{r}(i)$
9	$\hat{\mathbf{w}}(i) = \hat{\mathbf{w}}(i-1) + \Delta\hat{\mathbf{w}}(i)$

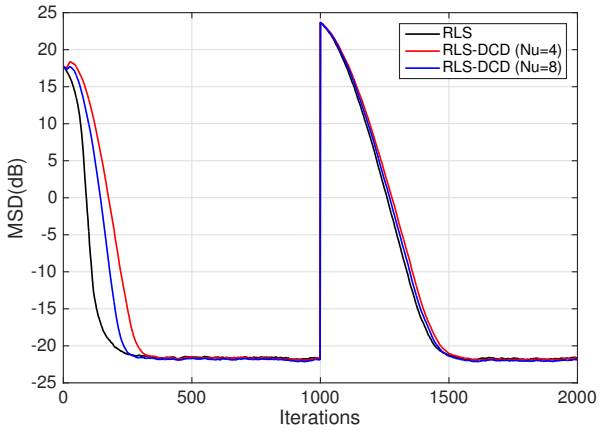


Fig. 2. MSD and tracking performance comparison between RLS and RLS-DCD when estimating a Volterra model, using $M_b = 16$, $N = 20$, $K = 4$ and $M = 110$.

IV. RESULTS

In this section, we compare the new RLS-DCD algorithm with RLS, NLMS and APA for the identification of a Volterra model. The true plant in our simulation has a linear part plus a 2nd-order term, $N = 20$ and $K = 4$. The input signal $\mathbf{x}(i)$ is white Gaussian noise with unit variance, independent of the measurement noise $v(i)$. The desired signal is given by

$$d(i) = \mathbf{w}_{\text{opt}}^T\mathbf{s}(i) + v(i) \tag{19}$$

where $\mathbf{w}_{\text{opt}}$ is the optimum coefficient vector (chosen randomly from a standard Gaussian distribution), and $v(i)$ is white Gaussian noise with variance σ_0^2. The forgetting factor is $\lambda = 1 - 2^{-6}$ and $\mathbf{R}(0) = (1/\lambda)\cdot\mathbf{I}$, where $\mathbf{I}$ is the identity matrix of appropriate size.

For the RLS-DCD algorithm, we use the same value of λ, $H = 64$ and $M_b = 16$. The simulation in Fig. 2 compares the mean square deviation (MSD $= E\{\|\mathbf{w} - \mathbf{w}_{\text{opt}}\|^2\}$) of the RLS and RLS-DCD with the same values of $N = 20$ and $K = 4$, so from (11), $M = 110$. We choose N_u to be 4 or 8, and include a change in the system coefficients at the middle of the simulation to compare the tracking performance of the algorithms. We use 100 simulations to obtain the ensemble-average learning curves.

We see from Fig. 2 that the performance of the RLS-DCD algorithm is already close to that of the standard RLS, even when using N_u as small as 4.

Fig. 3(a) and 3(b) compare the number of additions necessary for NLMS, RLS, APA with $\gamma = 4$ and $\gamma = 35$ and RLS-DCD. For this comparison we use $N_u = 4$, $M_b = 16$ as in Fig. 2, but vary the memory depth N for fixed $K = 4$ in Fig. 3(a), while in Fig. 3(b) we fix $N = 20$ and vary the maximum cross-term delay K. As we can see, the number of additions is in most cases smaller than that obtained with APA with $\gamma = 4$, and much smaller than the number of additions needed by RLS and APA with $\gamma = 35$. The number of multiplications behaves similarly.

Another simulation was performed by comparing the performance between RLS, RLS-DCD, NLMS and APA. For this simulation we used the same parameters as in Fig. 2 for RLS and RLS-DCD, and a step-size $\mu = 0.8$ for NLMS, $\mu = 0.31$ and $\mu = 0.023$ for APA with projection orders $\gamma = 4$ and $\gamma = 35$, respectively. As we can see in Fig. 4(a) (APA, $\gamma = 4$) and Fig. 4(b) (APA, $\gamma = 35$), both NLMS and APA have slower convergence when compared to the RLS and RLS-DCD algorithms.

V. CONCLUSION

In this paper, we propose a DCD-based Volterra RLS algorithm with reduced computational complexity, exemplifying the method using second-order Volterra kernels. When the maximum delay K in the cross-terms of the Volterra kernel is kept small, the computational complexity of the new algorithm grows linearly with the filter length and is comparable with the complexity of a low order APA.

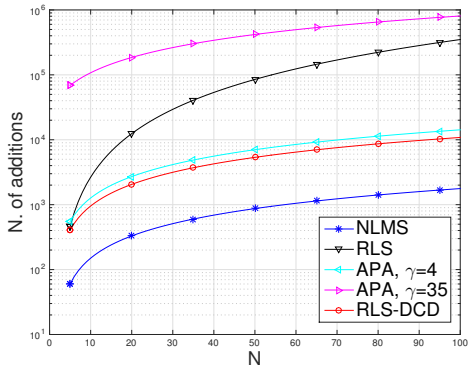
We compared the new algorithm with standard RLS, NLMS and APA, showing that the same steady-state MSD from RLS can be achieved using the RLS-DCD algorithm, but with fastest convergence when compared to NLMS and APA (even for an APA order as high as $\gamma = 35$).

VI. REFERENCES

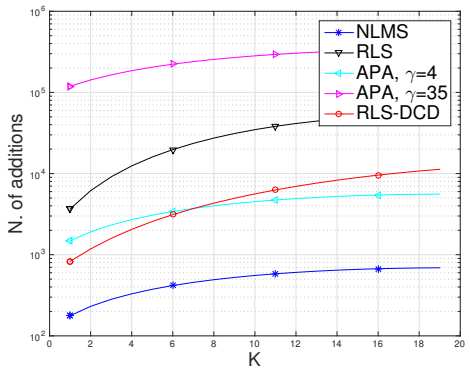
[1] V. J. Mathews, "Adaptive polynomial filters," *IEEE Signal Process. Mag.*, vol. 8, pp. 10–26, 1991.

[2] A. Stenger and R. Rabenstein, "Adaptive Volterra filters for nonlinear acoustic echo cancellation." in *NSIP*, 1999, pp. 679–683.

[3] A. Fermo, A. Carini, and G. L. Sicuranza, "Low-complexity nonlinear adaptive filters for acoustic echo cancellation in GSM handset receivers," *European transactions on telecommunications*, vol. 14, no. 2, pp. 161–169, 2003.

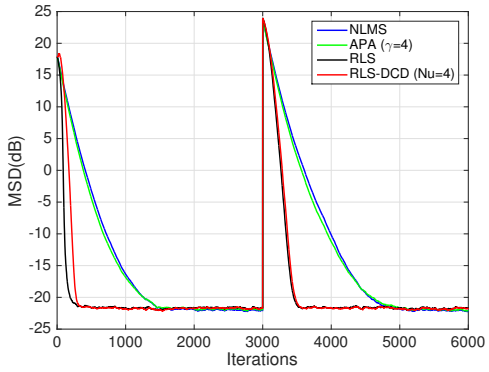


(a)

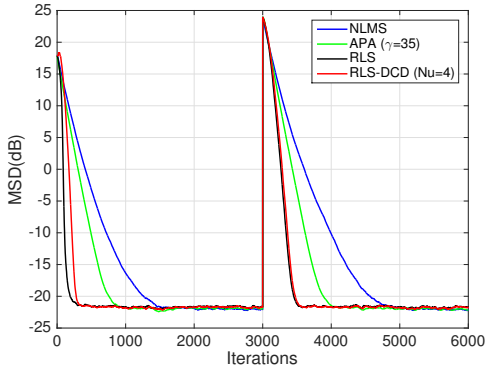


(b)

Fig. 3. Comparison of the number of additions between NLMS, RLS, APA ($\gamma = 4$), APA ($\gamma = 35$) and RLS-DCD for $N_u = 4$. (a) $K = 4$; (b) $N = 20$.



(a)



(b)

Fig. 4. MSD and tracking performance comparison between RLS, NLMS, RLS-DCD and APA using (a) $\gamma = 4$, (b) $\gamma = 35$, $M_b = 16$, $N_u = 4$, $K = 4$, $N = 20$ and $M = 110$.

[4] G. L. Sicuranza and A. Carini, "A multichannel hierarchical approach to adaptive Volterra filters employing filtered-X affine projection algorithms," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1463–1473, 2005.

[5] L. Yao and C.-C. Lin, "Identification of nonlinear systems by the genetic programming-based Volterra filter," *Signal Processing, IET*, vol. 3, no. 2, pp. 93–105, 2009.

[6] A. H. Sayed, *Adaptive filters*. John Wiley & Sons, 2011.

[7] A. Ramos, J. Apolinário, and S. Werner, "Multichannel fast qrd-rls adaptive filtering: Block-channel and sequential-channel algorithms based on updating backward prediction errors," *Signal Processing*, vol. 87, no. 7, pp. 1781–1798, Jan. 2007.

[8] M. Shoaib, S. Werner, and J. A. Apolinário, "Multichannel fast QR-decomposition algorithms: Weight extraction method and its applications," *IEEE Trans. Signal Process.*, vol. 58, no. 1, pp. 175–188, 2010.

[9] Y. Zakharov, G. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3150–3161, 2008.

[10] F. G. Almeida Neto, V. H. Nascimento, and Y. V. Zakharov, "Low-complexity widely linear RLS filter using DCD iterations," in *XXX SBrT (Brazilian Symposium on Telecommunications)*, 2012.

[11] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley-Interscience, 2000.

[12] V. H. Nascimento and M. T. M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing:*, R. Chellappa and S. Theodoridis, Eds. Chennai: Academic Press, 2014, vol. 1, Signal Processing Theory and Machine Learning, pp. 619—761.

[13] M. D. Miranda, M. Gerken, and M. T. M. Silva, "Efficient implementation of error-feedback LSL algorithm," *Electronics Letters*, vol. 35, no. 16, pp. 1308–1309, 1999.

[14] J. Liu, Y. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 11, pp. 2425–2438, 2009.

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