Low-complexity widely linear RLS filter using DCD iterations

Fernando G. Almeida Neto, Vítor H. Nascimento and Yuriy V. Zakharov

***Resumo*— Recentemente, filtros adaptativos amplamente lineares estão sendo usados para acessar completamente estatísticas de segunda ordem de sinais impróprios, com o objetivo de melhorar a estimação. Essa característica torna esses filtros vantajosos em relação aos seus equivalentes estritamente lineares, apesar de apresentarem maior complexidade computacional. Nesse sentido, com o intuito de reduzir o custo computacional do RLS amplamente linear (WL-RLS), em um artigo anterior foi proposta uma versão de complexidade reduzida – que foi chamada de RC-WL-RLS – e que apresentou um quarto da complexidade computacional do WL-RLS. Contudo, o algoritmo obtido ainda manteve complexidade $\mathcal{O}(N^2)$ (em que N representa o comprimento do vetor regressor). No presente trabalho, o RC-WL-RLS é modificado e é proposta uma modificação do algoritmo dichotomous coordinate descent (DCD) para iterativamente resolver as equações normais. Com essa abordagem, o número de multiplicações por iteração é reduzido e um algoritmo numericamente estável e de complexidade linear com N é obtido. Simulações são realizadas para comprovar o funcionamento do algoritmo proposto.**

***Palavras-Chave*— Estimação amplamente linear, Estimação com complexidade reduzida, Algoritmo RLS amplamente linear de complexidade reduzida, Algoritmo dichotomous coordinate descent (DCD)**

***Abstract*— Widely linear (WL) adaptive filters have been receiving much attention recently, since they take advantage of the full second-order statistics of improper signals. While this approach provides estimation gains if compared to their strictly linear (SL) counterparts, WL algorithms generally present higher computational complexity. In order to reduce the computational cost, we presented in a previous paper a reduced-complexity (RC) version of the WL-RLS algorithm – the RC-WL-RLS algorithm – which was shown to be 4 times less expensive than the WL-RLS, but still keeps the $\mathcal{O}(N^2)$ complexity (where N is the length of the regressor vector). In this paper, we modify the RC-WL-RLS, applying the dichotomous coordinate descent (DCD) algorithm to iteratively solve the normal equations. With this approach, we reduce the number of multiplications per iteration and obtain a numerically stable widely-linear RLS algorithm with computational complexity linear on N. Simulations are presented to support our approach.**

***Keywords*— Widely linear estimation, Reduced-complexity estimation, Reduced-complexity widely linear RLS, Dichotomous coordinate descent (DCD) algorithm**

I. INTRODUCTION

It is well-known that the Recursive Least-Squares (RLS) algorithm [1] converges faster than the Least Mean-Squares (LMS) algorithm [1], and that the RLS is also less sensitive to the spread of the eigenvalues in the covariance matrix. However, while the LMS presents linear computational cost, the RLS complexity is quadratic with the regressor vector length ($\mathcal{O}(N^2)$). Moreover, the RLS algorithm also suffers with numerical instability, requiring more care during the implementation in order to avoid divergence. The computational cost is also aggravated if we use a widely linear (WL) RLS algorithm [2]. In this case, the length of the regressor vector is duplicated to use both the complex data and their conjugate, which provides smaller mean-square error (MSE) for improper (or non-circular) input [3]. This approach is used to account for full second-order statistics of improper signals and achieve better estimation.

In order to reduce the complexity of the WL-RLS, a reduced-complexity (RC) version of the WL-RLS was proposed in [4]. In that paper, a linear transformation was used to obtain an algorithm with exactly the same performance, but with a complexity of one-fourth the complexity of the WL-RLS. Although it was shown that the RC-WL-RLS has complexity similar to the strictly linear (SL) RLS algorithm, the cost was still $\mathcal{O}(N^2)$ and the numerical stability still remained unsolved. (See Table III to compare the number of real operations per iteration of the algorithms, where we call strictly linear (SL) RLS the traditional complex RLS algorithm.)

Note that the quadratic cost of the algorithms comes from the update of the inverse autocorrelation matrix. In [5] and [6], the dichotomous coordinate descent (DCD) was proposed as an alternative to iteratively solve the normal equations. It was shown in [7] that the DCD is an efficient and stable alternative to implement the RLS algorithm in FPGAs, since this algorithm uses less multiplications to solve the normal equations, and less multipliers lead to the reduction of the consumed area. The DCD is particularly attractive for the reason that one can obtain an RLS algorithm with complexity proportional to N. Moreover, since the DCD-RLS does not propagate the inverse autocorrelation matrix, it does not suffer from numerical instability.

In this paper we modify the DCD algorithm to use with the RC-WL-RLS. As a result, we obtain an algorithm that presents low-complexity – that is, linear on N– and which is numerically stable. We compare the obtained algorithm (the DCD-WL-RLS) with the RC-WL-RLS and show that the new approach provides a low-complexity algorithm with almost the same performance of the RC-WL-RLS, which is supported by our simulations.

This paper is organized as follows: Section II presents the RC-WL-RLS algorithm. In Section III we modify the RC-

Fernando G. Almeida Neto and Vítor H. Nascimento are with the Dept. of Eletronic Systems, Escola Politécnica of the Univeristy of São Paulo, São Paulo, SP, Brasil. e-mail: {fganeto, vitor}@lps.usp.br.

Yuriy V. Zakharov is with the Dept. of Electronics of the University of York, York, U.K.. e-mail: yury.zakharov@york.ac.uk

WL-RLS to obtain the DCD-WL-RLS. In Section IV, we present the version of DCD algorithm used here. Simulations comparing the algorithms are performed in Section V, and we conclude the paper in Section VI.

II. The RC-WL-RLS algorithm

The RC-WL-RLS was first presented in [4] as a reduced-complexity version of the WL-RLS algorithm. The main difference between those algorithms is that using the linear transformation

$$\mathbf{U} = \begin{bmatrix} 1 & j \\ 1 & -j \end{bmatrix} \otimes \mathbf{I}_N,$$

where $j = \sqrt{-1}$, $\mathbf{I}_N$ is the N-order identity matrix and $\otimes$ represents the Kronecker product [8], we can modify the complex WL-RLS regressor vector

$$\mathbf{s}_{WL}(k) = \begin{bmatrix} \mathbf{s}_R(k) + j\mathbf{s}_I(k) \\ \mathbf{s}_R(k) - j\mathbf{s}_I(k) \end{bmatrix}$$

to the real RC-WL-RLS regressor

$$\mathbf{s}_{RC}(k) = \begin{bmatrix} \mathbf{s}_R^T(k) & \mathbf{s}_I^T(k) \end{bmatrix}^T,$$

where $\mathbf{s}_R(n)$ and $\mathbf{s}_I(n)$ are $N \times 1$ vectors containing the real and the imaginary parts of the data, respectively. In this case, using a real regressor vector, we replace many complex-complex multiplications (which cost 4 real multiplications and 2 real sums each) by real-complex multiplications (which cost only 2 real multiplications each), reducing the complexity. In [4], it is shown that both the algorithms achieve exactly the same performance, since $\mathbf{U}\mathbf{U}^H/4 = \mathbf{I}_{2N}$. Thus, the RC-WL-RLS is able to use full second-order statistics, but with reduced complexity. The algorithm is summarized in Table I.

TABLE I
RC-WL-RLS ALGORITHM

Initialization
$\mathbf{w}_{RC}(0) = \mathbf{0}_{2N\times 1}$
$\mathbf{P}_{RC}(0) = \delta\mathbf{I}_{2N}$

for $k = 0, 1, \ldots$
$\gamma_{RC}(k) = (\lambda_{RC} + \mathbf{s}_{RC}^H(k)\mathbf{P}_{RC}(k)\mathbf{s}_{RC}(k))^{-1}$
$\mathbf{k}_{RC}(k) = \mathbf{P}_{RC}(k)\mathbf{s}_{RC}(k)\gamma_{RC}(k)$
$e_{RC}(k) = d(k) - \mathbf{w}_{RC}^H(k)\mathbf{s}_{RC}(k)$
$\mathbf{w}_{RC}(k+1) = \mathbf{w}_{RC}(k) + \mathbf{k}_{RC}(k)e_{RC}^*(k)$
$\mathbf{P}_{RC}(k+1) = \frac{1}{\lambda_{RC}}\left[\mathbf{P}_{RC}(k) + \mathbf{k}_{RC}(k)\mathbf{k}_{RC}^H(n)\gamma_{RC}(k)\right]$

Note that $d(k)$ is the desired signal and $e_{RC}(k)$ is the estimation error. $\mathbf{w}_{RC}(k)$ is a vector containing the coefficient estimates, while $\mathbf{P}_{RC}(k)$ is the estimate of the inverse covariance matrix. We call $\mathbf{k}_{RC}(k)$ the Kalman gain and λ_{RC} and $\gamma_{RC}(k)$ are the forgetting factor and the conversion factor, respectively. We use an identity matrix multiplied by a small constant δ to initialize $\mathbf{P}_{RC}(0)$, and $\mathbf{w}_{RC}(0)$ is initialized with a zero-vector.

From Table III, where we present the computational complexity of this algorithm, one could think that, besides the complexity reduction of the RC-WL-RLS, one could do better, since the algorithm is still quadratic in the complexity. In Section III we present the DCD-WL-RLS algorithm, which presents still lower computational complexity.

III. The DCD-WL-RLS algorithm

In this section, we modify the transversal DCD-RLS proposed in [7] to develop our DCD-WL-RLS with linear complexity cost. We assume in this paper that both $\mathbf{s}_R(k)$ and $\mathbf{s}_I(k)$ are tap-delay lines, i.e.,

$$\mathbf{s}_R(k) = \begin{bmatrix} s_R(k) & s_R(k-1) & \cdots & s_R(k-N+1) \end{bmatrix}^T$$

and

$$\mathbf{s}_I(k) = \begin{bmatrix} s_I(k) & s_I(k-1) & \cdots & s_I(k-N+1) \end{bmatrix}^T,$$

respectively. Since $\mathbf{s}_{WL}(k)$ is composed of two tap-delay lines, the fast algorithm in [7] cannot be directly used. We show below how the DCD-RLS can be modified to work with WL data in the present situation.

When we deal with the RC-WL-RLS algorithm, for each k iteration we want to find the solution for the normal equations

$$\mathbf{R}(k) = \mathbf{w}(k)\boldsymbol{\beta}(k), \tag{1}$$

where

$$\mathbf{R}(k) = \sum_{i=0}^{k} \lambda^{k-i}\mathbf{s}_{RC}(i)\mathbf{s}_{RC}^T(i) = \lambda\mathbf{R}(k-1) + \mathbf{s}_{RC}(k)\mathbf{s}_{RC}^T(k) \tag{2}$$

is the $2N \times 2N$ correlation matrix and λ is the forgetting factor. $\mathbf{R}(k)$ is real and symmetric. $\boldsymbol{\beta}(n)$ is a vector given by

$$\boldsymbol{\beta}(k) = \sum_{i=0}^{k} \lambda^{k-i}d(i)\mathbf{s}_{RC}(i) = \lambda\boldsymbol{\beta}(k-1) + d(k)\mathbf{s}_{RC}(k). \tag{3}$$

We can write eq. (2) as

$$\mathbf{R}(k) = \begin{bmatrix} \mathbf{R}_R(k) & \mathbf{R}_{RI}(k) \\ \mathbf{R}_{RI}^T(k) & \mathbf{R}_I(k) \end{bmatrix},$$

where

$$\mathbf{R}_R(k) = \sum_{i=0}^{k} \lambda^{k-i}\mathbf{s}_R(i)\mathbf{s}_R^T(i) = \lambda\mathbf{R}_R(k-1) + \mathbf{s}_R(k)\mathbf{s}_R^T(k), \tag{4}$$

$$\mathbf{R}_I(k) = \sum_{i=0}^{k} \lambda^{k-i}\mathbf{s}_I(i)\mathbf{s}_I^T(i) = \lambda\mathbf{R}_I(k-1) + \mathbf{s}_I(k)\mathbf{s}_I^T(k), \tag{5}$$

and

$$\mathbf{R}_{RI}(k) = \sum_{i=0}^{k} \lambda^{k-i}\mathbf{s}_R(i)\mathbf{s}_I^T(i) = \lambda\mathbf{R}_{RI}(k-1) + \mathbf{s}_R(k)\mathbf{s}_I^T(k) \tag{6}$$

are $N \times N$ matrices. Note that $\mathbf{R}_R(k)$ and $\mathbf{R}_I(k)$ are symmetric matrices, but $\mathbf{R}_{RI}(k)$ is not symmetric in general. $\mathbf{R}_R(k)$ and $\mathbf{R}_I(k)$ can be evaluated using the low-cost update proposed in [7]:

$$\mathbf{R}_R(k) = \begin{bmatrix} r_R(k) & \boldsymbol{\rho}_R^T(k) \\ \boldsymbol{\rho}_R(k) & \mathbf{R}_R^{(1:N-1,1:N-1)}(k-1) \end{bmatrix}, \tag{7}$$

where the notation $\mathbf{R}_R^{(1:N-1,1:N-1)}(k-1)$ stands for a matrix with the elements of $\mathbf{R}_R(k-1)$ from row 1 to $N-1$ and from column 1 to $N-1$. $r_R(k)$ is a real number and $\boldsymbol{\rho}_R(k)$ is an $(N-1)\times 1$ vector. From eq. (7), we note that we only need to update the first column of $\mathbf{R}_R(k)$, since we already have $\mathbf{R}_R^{(1:N-1,1:N-1)}(k-1)$ from the last iteration and $\mathbf{R}_R(k)$ is symmetric. In order to obtain the first row of $\mathbf{R}_R(k)$ we

only need to copy the values of the first column. Thus, we calculate only the first column of $\mathbf{R}_R(k)$ – that is $\mathbf{R}_R^{(1:N,1)}(k)$ – to update eq. (7), i.e.,

$$\mathbf{R}_R^{(1:N,1)}(k) = \lambda \mathbf{R}_R^{(1:N,1)}(k-1) + s_R(k)\mathbf{s}_R(k).$$

Similarly, we update the first column of eq. (5) using

$$\mathbf{R}_I^{(1:N,1)}(k) = \lambda \mathbf{R}_I^{(1:N,1)}(k-1) + s_I(k)\mathbf{s}_I(k).$$

In order to update $\mathbf{R}_{RI}(k)$, we write (6) as the matrix

$$\mathbf{R}_{RI}(k) = \begin{bmatrix} r_{RI}(k) & \boldsymbol{\rho}_{RI}^T(k) \\ \boldsymbol{\rho}_{IR}(k) & \mathbf{R}_{RI}^{(1:N-1,1:N-1)}(k-1) \end{bmatrix}, \quad (8)$$

where

$$r_{RI}(k) = \sum_{i=0}^{k} \lambda^{k-i} s_R(i) s_I(i),$$

$$\boldsymbol{\rho}_{RI}(k) = \sum_{i=0}^{k} \lambda^{k-i} s_R(i) \mathbf{s}_I(i-1), \quad (9)$$

$$\boldsymbol{\rho}_{IR}(k) = \sum_{i=0}^{k} \lambda^{k-i} s_I(i) \mathbf{s}_R(i-1) \quad (10)$$

and

$$\begin{aligned} &\mathbf{R}_{RI}^{(1:N-1,1:N-1)}(k-1) = \\ &\quad = \sum_{i=0}^{k} \lambda^{k-i} \mathbf{s}_R^{(1:N-1)}(i-1) \mathbf{s}_I^{(1:N-1)}(i-i) \\ &\quad = \sum_{i=0}^{k-1} \lambda^{k-i} \mathbf{s}_R^{(2:N)}(i) \mathbf{s}_I^{(2:N)}(i) \\ &\quad = \mathbf{R}_{RI}^{(2:N,2:N)}(k). \end{aligned} \quad (11)$$

Note from eq. (11) that $\mathbf{R}_{RI}^{(1:N-1,1:N-1)}(k-1)$ is a block matrix obtained from iteration $k-1$, which does not need to be updated. Equations (9) and (10) show that, in general $\boldsymbol{\rho}_{RI} \neq \boldsymbol{\rho}_{IR}$, which means that they need to be calculated separately. To obtain the first column of $\mathbf{R}_{RI}(k)$, we define the column vector

$$\begin{aligned} \mathbf{R}_{RI}^{(1:N,1)}(k) &= \begin{bmatrix} r_{RI}(k) & \boldsymbol{\rho}_{IR}^T(k) \end{bmatrix}^T \\ &= \textstyle\sum_{i=0}^{k} \lambda^{k-i} s_I(i) \mathbf{s}_R(i) \\ &= \textstyle\sum_{i=0}^{k-1} \lambda^{k-i} s_I(i) \mathbf{s}_R(i) + s_I(k)\mathbf{s}_R(k) \\ &= \lambda \mathbf{R}_{RI}^{(1:N,1)}(k-1) + s_I(k)\mathbf{s}_R(k), \end{aligned} \quad (12)$$

which is recursively updated. The first row of $\mathbf{R}_{RI}(k)$ is updated in a similar manner, i.e.,

$$\mathbf{R}_{RI}^{(1,2:N)}(k) = \lambda \mathbf{R}_{RI}^{(1,2:N)}(k-1) + s_R(k)\mathbf{s}_I^{(2:N)^T}(k). \quad (13)$$

Note that in eq. (13) we do not calculate the first element of the row, since it is already computed in eq. (12). Using this approach, we can calculate matrix $\mathbf{R}(k)$ with the update of three $N \times N$ block-matrices. Recall that for each block-matrix, we only need to update the first row – in the case of $\mathbf{R}_R(k)$ and $\mathbf{R}_I(k)$– or the first row and the first column (in the case of $\mathbf{R}_{RI}(k)$). We use this low-cost form to update matrix $\mathbf{R}(k)$ in the DCD-WL-RLS algorithm (see Table III).

Assume that we update $\mathbf{R}(k)$ with the procedure shown before. In order to obtain the DCD-WL-RLS, recall eq. (1), but in the instant $k-1$, i.e.,

$$\mathbf{R}(k-1) = \mathbf{w}(k-1)\boldsymbol{\beta}(k-1). \quad (14)$$

We define the residue of the normal equations, after obtaining an approximate solution $\hat{\mathbf{w}}(k-1)$, as the vector

$$\mathbf{r}(k-1) = \boldsymbol{\beta}(k-1) - \mathbf{R}(k-1)\hat{\mathbf{w}}(k-1). \quad (15)$$

Moreover, define

$$\Delta\mathbf{R}(k) = \mathbf{R}(k) - \mathbf{R}(k-1), \quad (16)$$

$$\Delta\boldsymbol{\beta}(k) = \boldsymbol{\beta}(k) - \boldsymbol{\beta}(k-1), \quad (17)$$

and

$$\Delta\mathbf{w}(k) = \mathbf{w}(k) - \hat{\mathbf{w}}(k-1). \quad (18)$$

Substituting eqs. (16), (17) and (18) in (14), after some algebra manipulation, we obtain

$$\mathbf{R}(k)\Delta\mathbf{w}(k) = \boldsymbol{\beta}_0(k), \quad (19)$$

where

$$\boldsymbol{\beta}_0(k) = \mathbf{r}(k-1) + \Delta\boldsymbol{\beta}(k) - \Delta\mathbf{R}(k)\hat{\mathbf{w}}(k-1). \quad (20)$$

Solving equation (19) instead of (1) is a way of taking advantage of the previous solution $\hat{\mathbf{w}}(k-1)$ to reduce the complexity of the algorithm. This is due to the iterative structure of DCD – solving (19) is equivalent to start DCD with $\hat{\mathbf{w}}(k-1)$ as initial condition, reducing the number of DCD iterations (and the complexity) necessary at each time instant. We apply the DCD algorithm of Table IV to iteratively solve this system of equations in the DCD-WL-RLS algorithm and obtain the approximate solution

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \Delta\hat{\mathbf{w}}(k). \quad (21)$$

We can also write eq. (15) in terms of $\boldsymbol{\beta}_0(k)$ and $\Delta\hat{\mathbf{w}}(k)$, using eqs. (20) and (21), i.e.,

$$\mathbf{r}(k) = \boldsymbol{\beta}_0(k) - \mathbf{R}(k)\Delta\hat{\mathbf{w}}(k).$$

Recall (16) and (17). Using the second identity of (2) and (3) in eqs. (16) and (17), respectively, we can express

$$\Delta\mathbf{R}(k) = (1-\lambda)\mathbf{R}(k-1) + \mathbf{s}_{RC}(k)\mathbf{s}_{RC}^T(k) \quad (22)$$

and

$$\Delta\boldsymbol{\beta}(k) = (1-\lambda)\boldsymbol{\beta}_0(k) + d(k)\mathbf{s}_{RC}(k). \quad (23)$$

Define the filter estimative

$$y(k) = \mathbf{s}_{RC}^T(k)\hat{\mathbf{w}}(k-1).$$

Using (22) and (23) in eq. (15),

$$\begin{aligned} &\Delta\mathbf{R}(k)\hat{\mathbf{w}}(k-1) = \\ &\quad = (\lambda-1)\left[\boldsymbol{\beta}(k-1) - \mathbf{r}(k-1)\right] + \mathbf{s}_{RC}(k)y(k), \end{aligned} \quad (24)$$

which can be used in (20) to obtain

$$\boldsymbol{\beta}_0(k) = \lambda\mathbf{r}(k-1) + e(k)\mathbf{s}_{RC}(k),$$

where $e(k) = d(k) - y(k)$ corresponds to the *a priori* error. The DCD-WL-RLS algorithm is summarized in Table II,

TABLE II

DCD-WL-RLS ALGORITHM

Step	Equation
	Initialization: $\Delta\hat{\mathbf{w}}(0) = \mathbf{0}_{2N\times 1}$, $\mathbf{r}_{2N\times 1}(0) = \mathbf{0}$, $\mathbf{R}_R = \delta\mathbf{I}_N$, $\mathbf{R}_I = \delta\mathbf{I}_N$, $\mathbf{R}_{RI} = \mathbf{0}_N$
	for $k = 1, 2, \ldots$
1	$\mathbf{R}_R^{(1,1:N)}(k) = \lambda\mathbf{R}_R^{(1,1:N)}(k-1) + s_R(k)\mathbf{s}_R(k)$
2	$\mathbf{R}_I^{(1,1:N)}(k) = \lambda\mathbf{R}_I^{(1,1:N)}(k-1) + s_I(k)\mathbf{s}_I(k)$
3	$\mathbf{R}_{RI}^{(1,1:N)}(k) = \lambda\mathbf{R}_{RI}^{(1,1:N)}(k-1) + s_I(k)\mathbf{s}_R(k)$
4	$\mathbf{R}_{RI}^{(2:N,1)}(k) = \lambda\mathbf{R}_{RI}^{(2:N,1)}(k-1) + s_R(k)\mathbf{s}_I^{(2:N)^T}(k)$
5	$\mathbf{R}_R(k)$, $\mathbf{R}_I(k)$, $\mathbf{R}_{RI}(k) \Rightarrow \mathbf{R}(k)$
6	$y(k) = \mathbf{s}_{RC}^T(k)\hat{\mathbf{w}}(k)$
7	$e(k) = d(k) - y(k)$
8	$\boldsymbol{\beta}_0(k) = \lambda\mathbf{r}(k-1) + e(k)\mathbf{s}_{RC}(k)$
9	$\mathbf{R}(k)\Delta\mathbf{w}(k) = \mathbf{p}_0(k) \Rightarrow \Delta\hat{\mathbf{w}}(k)$, $\mathbf{r}(k)$
10	$\hat{\mathbf{w}}(k) = \hat{\mathbf{w}}(k-1) + \Delta\hat{\mathbf{w}}(k)$

where we also provide the initialization of the algorithm. Note that the DCD algorithm, which is presented in Section IV, is used to solve the 9^{th} row of Table II. Table III compares the complexity of the developed algorithm with the SL, WL and RC-WL-RLS. Note that there are two lines referring to the DCD-WL-RLS. The first presents the complexity considering that λ can be any real number such that $0 < \lambda < 1$. The second assumes that $\lambda = 1 - 2^{-m}$, $m \in \mathbb{N}$, which allows the substitution of multiplications by bit-shifts and sums in steps 1, 2, 3, 4 and 8. N_u is the maximum number of successful iterations in DCD (see Section IV).

TABLE III

COMPUTATIONAL COMPLEXITY OF THE SL-RLS, WL-RLS, RC-WL-RLS AND DCD-WL-RLS IN TERMS OF REAL OPERATIONS PER ITERATION

Algorithm	+	×	÷
SL-RLS	$6N^2 + 14N - 1$	$7N^2 + 21N + 1$	1
WL-RLS	$24N^2 + 28N - 1$	$28N^2 + 42N + 1$	1
RC-WL-RLS	$6N^2 + 11N$	$8N^2 + 14N + 1$	1
DCD-WL-RLS	$N(8 + 4N_u + 2M_b) + N_u - 1$	$14N - 2$	-
DCD-WL-RLS ($\lambda = 1 - 2^{-m}$)	$N(14 + 4N_u + 2M_b) + N_u - 1$	$8N - 2$	-

IV. THE DCD ALGORITHM

The dichotomous coordinate descent (DCD) algorithm [5], [6] was proposed as a multiplication-free alternative to solve the least-squares (LS) problem equations [1]. The algorithm is implemented using sums and bit-shift operations to avoid multiplications and divisions – see Table IV, where we present the DCD algorithm version considered in this paper.

The cyclic DCD algorithm updates the solution vector $\Delta\hat{\mathbf{w}}$ in the cyclic order $n = 1, \ldots, N$, which is used by the DCD-WL-RLS algorithm to solve the normal equations. We assume that the elements of $\Delta\hat{\mathbf{w}}$ have amplitude such that $|\Delta\hat{\mathbf{w}}_p| \leq H$ (where H is a positive number) and that we use M_b bits to represent them. The constant α is the step-size of the DCD. N_u is the maximum number of successful iterations and is usually chosen as $N_u << N$. With this approach, the algorithm updates the most significant bits first and the less significant bits are updated later.

Note that during the DCD operation, we have two kinds of iterations: the successful and the unsuccessful iterations. The first kind occurs when the condition in step 3 is true, and the updates in steps $4-6$ are performed. Otherwise, the update is called "unsuccessful". The computational complexity of the DCD algorithm, presented in Table IV, is a random variable, which motivates the calculation of the worst case complexity. For the worst case, the complexity corresponds to $P_m = 0$ multiplications and $P_a \leq N(2N_u + M_b - 1) + N_u$ sums. Recall that this complexity is a pessimistic bound, since in general this upper bound is not reached.

TABLE IV

DCD ALGORITHM

Step	Equation
	Initialization: $\Delta\hat{\mathbf{w}} = \mathbf{0}_{2N\times 1}$, $\mathbf{r} = \boldsymbol{\beta}_0$ $\alpha = H$, $q = 0$
	for $m = 1, \ldots, M_b$
1	$\alpha = \alpha/2$
2	flag $= 0$
	for $n = 1, \ldots, 2N$
3	if $\lvert\mathbf{r}_p\rvert > (\alpha)/2)\mathbf{R}_{p,p}$ then
4	$\Delta\hat{\mathbf{w}}_p = \Delta\hat{\mathbf{w}}_p + \text{sign}(\mathbf{r}_p)\alpha$
5	$\mathbf{r} = \mathbf{r} - \text{sign}(\mathbf{r}_p)\alpha\mathbf{R}^{(1:2N,1)}$
6	$q = q + 1$
7	if $q > N_u$, the algorithm stops
8	if flag $= 1$, then repeat step 2

V. SIMULATIONS

In this section, we compare our new DCD-WL-RLS with the RC-WL-RLS algorithm. For this purpose, we consider the same identification system model used in [4], where a non-circular signal is applied to justify the WL approach. We assume that the input signal is given by

$$\mathbf{s}(k) = \sqrt{1-\epsilon^2}\mathbf{s}_R(k) + j\epsilon\mathbf{s}_I(k),$$

where $\mathbf{s}_R(k)$ and $\mathbf{s}_I(k)$ are two real-valued uncorrelated complex-Gaussian processes, with zero-mean and variance $\sigma_R^2 = \sigma_I^2 = 1$. We can choose ϵ between 0 and 1, and the condition for circularity occurs when $\epsilon = 1/\sqrt{2}$. However, we use $\epsilon = 0.1$ – which corresponds to a non-circular process – to perform our simulations. In figures 1 and 2 we assume that the optimum coefficients of the system to be identified are given by

$$w_{opt,i}^1 = \beta_1[(1 + \cos(2\pi(i-3)/5) - j(1 + \cos(2\pi(i-3)/10))], \quad (25)$$

for $i = 1, 2, \ldots, 5$ and $\beta_1 = 0.432$. In figure 3, in order to present the tracking performance of the DCD-WL-RLS, we start the simulation with the optimum coefficients of eq. 25 and after 200 iterations replace them by

$$w_{opt,i}^2 = \beta_2[(1 + \cos(\pi(i-3)/5) - j(1 + \cos(5\pi(i-3)/10))], \quad (26)$$

where $\beta_2 = \beta_1/4$. We assume that there is Gaussian noise $\eta(k)$ added to the desired signal and that the signal-to-noise ratio (SNR) is 40dB. We obtain the desired signal with

$$d(k) = \mathcal{R}e\{\mathbf{w}_{opt}^H\mathbf{s}(k)\} + \eta(k).$$

We define the RC-WL-RLS forgetting factor $\lambda_{RC} = 1 - 2^{-6}$ and we initialize $\mathbf{P}_{RC}(0) = 10^{-4} \cdot \mathbf{I}_{10}$, for $N = 5$.

For the DCD-WL-RLS algorithm, we choose λ_{DCD} equal to λ_{RC}, and we define $H = 1$. In our simulations, we compare the mean-square error (MSE) of the RC-WL-RLS and the DCD-WL-RLS algorithms in three situations: 1) varying the number of bits M_b as 2, 4, 8 or 16 bits and setting $N_u = 4$; 2) setting $M_b = 16$ bits and choosing N_u to be 1, 2, 4 or 8; and 3) changing the system coefficients to verify the tracking performance of the DCD-WL-RLS algorithm. We run 100 simulations to obtain the MSE as MSE $= E\{e^2(k)\}$. Figures 1, 2 and 3 show the results. Note from figure 1 that for a fixed

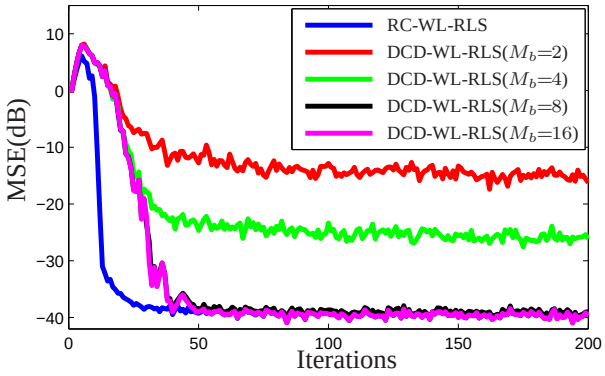

Fig. 1. MSE comparison between the RC-WL-RLS and the DCD-WL-RLS, using $N_u = 4$ and $M_b = 2, 4, 8, 16$, for non-circular input. 100 simulations were performed.

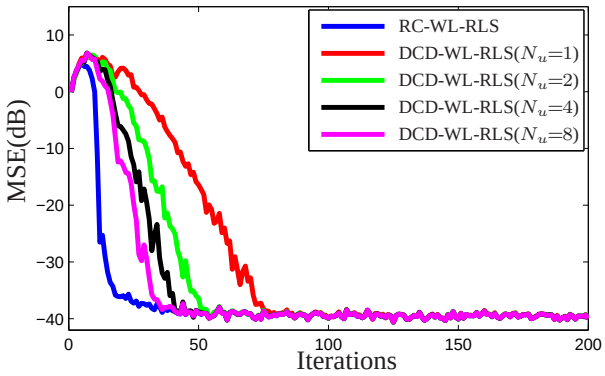

Fig. 2. MSE comparison between the RC-WL-RLS and the DCD-WL-RLS, using $M_b = 16$ and $N_u = 1, 2, 4, 8$, for non-circular input.100 simulations were performed.

N_u, the precision of the DCD-WL-RLS algorithm increases as the number of bits M_b increases, when compared to the RC-WL-RLS implemented with Matlab precision. With this information, we can choose M_b to guarantee an specific MSE – of course bounded by the SNR – and use the algorithm with the smaller number of bits necessary for a given implementation. In figure 2, we can see that for a fixed number of bits, when we increase N_u, we accelerate the convergence. However, even for $N_u = 1$, we note that the algorithm achieves a steady-state MSE similar to the RC-WL-RLS MSE, though 4 times slower than the RC-WL-RLS algorithm. In fact, when we apply the DCD to solve the normal equations, we reduce the complexity to $\mathcal{O}(N)$, but we pay for this reduction with the need of some more iterations to achieve the steady-state MSE. Since we can control the additional number of iterations by choosing a convenient N_u, this approach can be very attractive for low-cost implementations.

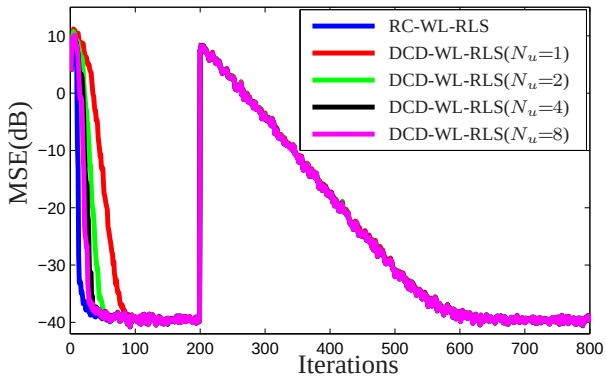

Fig. 3. Tracking performance of the RC-WL-RLS and the DCD-WL-RLS algorithms, when the system coefficients change. Non-circular input is used for a fixed $M_b = 16$ and $N_u = 1, 2, 4, 8$. 100 simulations were performed.

In figure 3, we show the DCD-WL-RLS algorithm subject to a modification of the system coefficients, which is used to verify the tracking performance. Note that for the different values of N_u proposed, the algorithm presents approximately the same performance when the system changes. As shown in [7], the tracking performance of the DCD-WL-RLS does not depend as much on N_u.

VI. Conclusions

In this paper, we propose a widely-linear RLS algorithm that is numerically stable (since it avoids the propagation of the inverse of the autocovariance matrix) and has a computational complexity that is linear on the filter length. Our simulations showed that the modification proposed slows down the initial convergence of the algorithm, but with a smart choice of the DCD parameters, the algorithm quickly recovers the performance and achieves the same steady-state MSE achieved by the RC-WL-RLS. The tracking performance of the DCD-WL-RLS does not depend as much on N_u. An analytical study of the DCD-WL-RLS is still under research, since traditional tools of the RLS analysis can not be applied here.

References

[1] A.H. Sayed, *Adaptive filters*, Wiley-IEEE Press, 2008.

[2] Jae-Jin Jeon, "RLS adaptation of widely linear minimum output energy algorithm for DS-CDMA systems," in *Proc. of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop*, July 2005, pp. 98 – 102.

[3] B. Picinbono and P. Chevalier, "Widely linear estimation with complex data," *IEEE Transaction on Signal Processing*, vol. 43, no. 8, pp. 2030 –2033, Aug. 1995.

[4] F.G.A. Neto, V.H. Nascimento, and M.T.M. Silva, "Reduced-complexity widely linear adaptive estimation," in *7th International Symposium on Wireless Communication Systems*, Sep. 2010, pp. 399–403.

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

[6] Y. Zakharov, G. White, and Jie Liu, "Fast RLS algorithm using dichotomous coordinate descent iterations," in *Signals, Systems and Computers. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, Nov. 2007, pp. 431 –435.

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

[8] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, MA, 1991.