Low-complexity widely linear RLS filter using DCD iterations

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

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 – and that the RLS complexity is quadratic with the regressor vector length ($O(N^2)$). Moreover, the RLS algorithm also suffers from 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 $O(N^2)$ 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-WL-RLS algorithm to support our approach.

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 ($O(N^2)$). 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.
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

\[ U = \begin{bmatrix} 1 & j \\ 1 & -j \end{bmatrix} \otimes I_N, \]

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

\[ s_{WL}(k) = \begin{bmatrix} s_R(k) + js_I(k) \\ s_R(k) - js_I(k) \end{bmatrix} \]

to the real RC-WL-RLS regressor

\[ s_{RC}(k) = \begin{bmatrix} s^R_R(k) & s^T_I(k) \end{bmatrix}^T, \]

where \( s_R(n) \) and \( s_I(n) \) 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 all the algorithms achieve exactly the same performance, since \( UU^H/4 = I_{2N} \). 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**

<table>
<thead>
<tr>
<th>Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_{RC}(0) = 0 )</td>
</tr>
<tr>
<td>( P_{RC}(0) = \delta I_{2N} )</td>
</tr>
<tr>
<td>for ( k = 0, 1, \ldots )</td>
</tr>
<tr>
<td>( \gamma_{RC}(k) = (K_{RC} + \gamma_{RC}(k)P_{RC}(k)s_{RC}(k))^{-1} )</td>
</tr>
<tr>
<td>( k_{RC}(k) = P_{RC}(k)k_{RC}(k)\gamma_{RC}(k) )</td>
</tr>
<tr>
<td>( e_{RC}(k) = d(k) - w_{RC}(k)s_{RC}(k) )</td>
</tr>
<tr>
<td>( w_{RC}(k + 1) = w_{RC}(k) + k_{RC}(k)e_{RC}(k) )</td>
</tr>
<tr>
<td>( P_{RC}(k + 1) = \frac{1}{\lambda_{RC}} [P_{RC}(k) + k_{RC}(k)k_{RC}(k)\gamma_{RC}(k)] )</td>
</tr>
</tbody>
</table>

Note that \( d(k) \) is the desired signal and \( e_{RC}(k) \) is the estimation error. \( w_{RC}(k) \) is a vector containing the coefficient estimates, while \( P_{RC}(k) \) is the estimate of the inverse covariance matrix. We call \( k_{RC}(k) \) the Kalman gain and \( \lambda_{RC} \) and \( \gamma_{RC}(k) \) are the forgetting factor and the conversion factor, respectively. We use an identity matrix multiplied by a small constant \( \delta \) to initialize \( P_{RC}(0) \), and \( w_{RC}(0) \) 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 \( s_R(k) \) and \( s_I(k) \) are tap-delay lines, i.e.,

\[ s_R(k) = \begin{bmatrix} s_R(k) & s_R(k-1) & \cdots & s_R(k-N+1) \end{bmatrix}^T \]

and

\[ s_I(k) = \begin{bmatrix} s_I(k) & s_I(k-1) & \cdots & s_I(k-N+1) \end{bmatrix}^T, \]

respectively. Since \( s_{WL}(k) \) 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

\[ R(k) = w(k)\beta(k), \tag{1} \]

where

\[ R(k) = \sum_{i=0}^{k} \lambda^{k-i}s_{RC}(i)s_{RC}^T(i) = \lambda R(k-1) + s_{RC}(k)s_{RC}^T(k) \tag{2} \]

is the \( 2N \times 2N \) correlation matrix and \( \lambda \) is the forgetting factor. \( R(k) \) is real and symmetric. \( \beta(n) \) is a vector given by

\[ \beta(k) = \sum_{i=0}^{k} \lambda^{k-i}d(i)s_{RC}(i) = \lambda \beta(k-1) + d(k)s_{RC}(k). \tag{3} \]

We can write eq. (2) as

\[ R(k) = \begin{bmatrix} R_R(k) & R_{RI}(k) \\ R_{RI}^T(k) & R_I(k) \end{bmatrix}, \]

where

\[ R_R(k) = \sum_{i=0}^{k} \lambda^{k-i}s_R(i)s^T_R(i) = \lambda R_R(k-1) + s_R(k)s^T_R(k), \tag{4} \]

\[ R_I(k) = \sum_{i=0}^{k} \lambda^{k-i}s_I(i)s^T_I(i) = \lambda R_I(k-1) + s_I(k)s^T_I(k), \tag{5} \]

and

\[ R_{RI}(k) = \sum_{i=0}^{k} \lambda^{k-i}s_R(i)s^T_I(i) = \lambda R_{RI}(k-1) + s_R(k)s^T_I(k), \tag{6} \]

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

\[ R_R(k) = \begin{bmatrix} r_R(k) & \rho_{R}^{T}(k) \\ \rho_R(k) & R_R^{(1:N-1,1:N-1)}(k-1) \end{bmatrix}, \tag{7} \]

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

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

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

$$R^{(1,N,1)}_R(k) = \lambda R^{(1,N,1)}_R(k-1) + s_f(k)s_f(k).$$

In order to update $R_{RI}(k)$, we write (6) as the matrix

$$R_{RI}(k) = \begin{bmatrix} r_{RI}(k) \\ \rho_{RI}(k) \end{bmatrix} = \begin{bmatrix} r_{RI}(k) \\ \rho_{RI}(k) \end{bmatrix} R^{(1,N-1,1,N-1)}_{RI}(k-1)$$

where

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

$$\rho_{RI}(k) = \sum_{i=0}^{k} \lambda^{k-i} s_{RI}(i) s_I(i-1),$$

and

$$R^{(1,N-1,1,N-1)}_{RI}(k-1) = \sum_{i=0}^{k-1} \lambda^{k-i} s_{RI}(i) s_{RI}(i-1)$$

$$= \sum_{i=0}^{k} \lambda^{k-i} s_{RI}(i) s_{RI}(i-1)$$

$$= \begin{bmatrix} r_{RI}(k) \\ \rho_{RI}(k) \end{bmatrix}.$$  

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

$$R^{(1,N,1)}_{RI}(k) = \begin{bmatrix} r_{RI}(k) \\ \rho_{RI}(k) \end{bmatrix} = \sum_{i=0}^{k} \lambda^{k-i} s_{RI}(i) s_{RI}(i)$$

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

$$= \lambda R^{(1,N,1)}_{RI}(k-1) + s_I(k)s_R(k),$$

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

$$R^{(1,2,N)}_{RI}(k) = \lambda R^{(1,2,N)}_{RI}(k-1) + s_R(k)s_T(k).$$  

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 $R(k)$ 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 $R_R(k)$ and $R_{RI}(k)$– or the first row and the first column (in the case of $R_{RI}(k)$). We use this low-cost form to update matrix $R(k)$ in the DCD-WL-RLS algorithm (see Table III).

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

$$R(k - 1) = w(k - 1)R(k - 1)b(k - 1).$$  

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

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

Moreover, define

$$\Delta R(k) = R(k) - R(k - 1),$$

$$\Delta \beta(k) = \beta(k) - \beta(k - 1),$$

and

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

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

$$R(k)\Delta w(k) = \beta_0(k),$$

where

$$\beta_0(k) = r(k - 1) + \Delta \beta(k) - \Delta R(k)\hat{w}(k - 1).$$

Solving equation (19) instead of (1) is a way of taking advantage of the previous solution $\hat{w}(k - 1)$ to reduce the complexity of the algorithm. This is due to the iterative structure of DCD – solving (19) is equivalent to start DCD with $\hat{w}(k - 1)$ 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

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

We can also write eq. (15) in terms of $\beta_0(k)$ and $\Delta \hat{w}(k)$, using eqs. (20) and (21), i.e.,

$$r(k) = \beta_0(k) - R(k)\Delta \hat{w}(k).$$

Recall $r(k) = \beta_0(k) - R(k)\Delta \hat{w}(k).$

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

$$\Delta R(k) = (1 - \lambda)R(k - 1) + s_{RC}(k)s_{RC}^T(k)$$

and

$$\Delta \beta(k) = (1 - \lambda)\beta_0(k) + d(k)s_{RC}(k).$$

Define the filter estimate

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

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

$$\Delta R(k)\hat{w}(k - 1) =$$

$$= (\lambda - 1)[\beta(k - 1) - r(k - 1)] + s_{RC}(k)y(k),$$

which can be used in (20) to obtain

$$\beta_0(k) = \lambda r(k - 1) + c(k)s_{RC}(k),$$

where $c(k) = d(k) - y(k)$ corresponds to the a priori error. The DCD-WL-RLS algorithm is summarized in Table II,
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 9th 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 \( \lambda \) 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).

### 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

\[
s(k) = \sqrt{1 - \epsilon^2} s_R(k) + j \epsilon s_I(k),
\]

where \( s_R(k) \) and \( s_I(k) \) are two real-valued uncorrelated complex-Gaussian processes, with zero-mean and variance \( \sigma_R^2 = \sigma_I^2 = 1 \). We can choose \( \epsilon \) 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

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

for \( i = 1, 2, \ldots, 5 \) and \( \beta_i = 0.432 \). 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

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

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

\[
d(k) = \mathcal{R}e\{w^R_{\text{opt}}(k)\} + \eta(k).
\]

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 \( \lambda_{DCD} \) equal to \( \lambda_{RC} \), 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

\[ MSE(dB) = \begin{array}{c}
-40 \\
-30 \\
-20 \\
-10 \\
0 \\
10 \\
\end{array} 
\]

\[ \text{Iterations} \]

![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.](image1)

\[ MSE(dB) = \begin{array}{c}
-40 \\
-30 \\
-20 \\
-10 \\
0 \\
10 \\
\end{array} 
\]

\[ \text{Iterations} \]

![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.](image2)

\[ MSE(dB) = \begin{array}{c}
-40 \\
-30 \\
-20 \\
-10 \\
0 \\
10 \\
\end{array} 
\]

\[ \text{Iterations} \]

![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.](image3)

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


