Insights on a Low-Cost Recursive Least-Squares Algorithm for Adaptive Noise Cancellation

Roxana Mihăescu, Cristian Stanciu, Cristian Anghel, Lucian Stanciu
Politehnica University of Bucharest, Romania
Email: roxana.2010.mihaescu@gmail.com
{cristian, canghel, lucians}@comm.pub.ro

***Abstract*—Adaptive Noise Cancellation (ANC) belongs to the interference cancellation class. It employs an adaptive filter to estimate a perturbation signal, which corrupts a primary acoustic source. In most of the corresponding applications, the goal is to imitate an original speech signal. This paper proposes the use of a low-complexity recursive least-squares (RLS) adaptive algorithm for the ANC procedure. The combination between the RLS method and the dichotomous coordinate descent (DCD) iterations offers good performance with acceptable arithmetic costs. Simulation results are provided in order to demonstrate the validity of the ANC system based on the RLS-DCD adaptive algorithm.**

Keywords: adaptive noise cancellation; recursive least-squares; dichotomous coordinate descent.

I. INTRODUCTION

Modern technology allows the deployment of telecommunication networks in challenging environments, which frequently introduce strong acoustic interference. The high-quality communication performed in extremely noisy surroundings, such as airplane cockpits or social gatherings, requires the real-time estimation of corrupted acoustic signals (usually speech sequences).

With the development of adaptive algorithms, the field of Adaptive Noise Cancellation (ANC) has also been the subject of intensive study [1]-[3]. The workhorse of signal processing systems employing adaptive methods is the Least Mean Squares (LMS) family [2]-[6]. Although the classical LMS adaptive algorithms were improved to a certain degree, their performances are limited when working with highly correlated signals. A new generation of efficiently implementable adaptive systems is required in order to increase the noise cancellation capabilities.

The standard recursive least-squares (RLS) adaptive methods have attractive convergence properties [2]-[6]. However, the classical solutions for directly solving the corresponding matrix inversion problem have high arithmetic complexities and require large amounts of computational resources. Moreover, the implementations employing the traditional RLS algorithms suffer from occasional numeric instability caused by higher order arithmetical operations, such as divisions. Although the Fast RLS (FRLS) [5] considerably reduces the arithmetic effort, it is not stable when working with nonstationary signals, such as speech.

In [7]-[9], the prohibitive nature of the RLS methods was approached using the combination with the dichotomous coordinate descent (DCD) iterations. The DCD part of the algorithm replaces the classical matrix inversion problem with an auxiliary system of equations, which is solved using only additions and bit-shifts. The solution is based on the statistical properties of the input signals and reduces the overall arithmetic complexity to a value proportional to L, which is used to denote the adaptive filter's length. The resulting RLS-DCD algorithm is a numerically stable alternative, offering comparable results in terms of adaptation speed and precision, with a considerably reduced computational effort [7]-[11]. By comparison, the classical RLS method has a complexity of $O(L^3)$, which can be reduced using Woodburry's identity to $O(L^2)$ – both methods are considered unaffordable for practical applications [2][5].

The original RLS-DCD solution was rarely tested with colored signals, such as speech sequences [8], [9]. It was later effectively applied for stereophonic acoustic echo cancelation (SAEC) setups requiring the estimation of multiple unknown systems [10]. This paper proposes the use of the RLS-DCD method for ANC systems employed in real-time recovery of speech signals. A theoretical model is presented and tested using different types of acoustic interference, with low Signal-to-Noise Ratio (SNR). Although the number of adaptive filter coefficients associated with ANC applications is lower than the case of acoustic echo cancellation (AEC) scenarios, the reduction in terms of arithmetic workload (in comparison to the classical RLS) is valuable for mobile devices (i.e., headphones, mobile phones, etc.). As a consequence, the compromise between arithmetic complexity and performance is analyzed, and a comparison is performed with the standard RLS.

The paper is organized as follows. In Section II, the theoretical model of the ANC setup is defined. Section III describes a new approach on the theory associated with RLS adaptive algorithms and Section IV introduces a low-complexity RLS-type method, which is suitable for acoustic applications, such as the ANC. The performances of the proposed adaptive method are demonstrated using simulations in Section V. The standard RLS adaptive algorithm is employed as a reference. Finally, in Section VI, a few conclusions are stated regarding the compromise between arithmetic complexity and the performance of the ANC system using a low-complexity RLS method.

II. System Model

Figure 1 illustrates the ANC scheme. By using the notation n for the discrete time index, we denote the *desired* signal $d(n)$ as the accumulation between the relevant signal $s(n)$ and the corrupting sequence $q(n)$ (also called the *interference* signal). The *input* of the adaptive algorithm $x(n)$ is a *reference* signal, which is linearly correlated with the interference $q(n)$. In literature, the relation between $x(n)$ and $q(n)$ is usually modelled through a finite impulse response (FIR) filter, which generates $q(n)$ using $x(n)$ as the input. In practical ANC applications, the samples corresponding to $x(n)$ and $d(n)$ are available through microphones [3]. The influence of the physical distance between the two acoustic sensors is represented in Figure 1 through the delay factor D, which is associated with the length of the mentioned FIR filter.

The purpose of the ANC system is to generate an estimate $y(n)$ of $q(n)$ (using the adaptive filter) and subtract it from the desired signal. Consequently, the error signal $e(n)$ is an estimate of $s(n)$, i.e., $e(n) \rightarrow s(n)$. The *error* of the adaptive algorithm is used to adjust the coefficients of the adaptive filter in order to minimize the noise interference. In an optimal situation, $e(n)$ is composed of the signal $s(n)$, free of the noise interference $q(n)$.

For the theoretical model of the adaptive algorithm we denote by $\hat{\mathbf{h}}(n)$ the L x 1 vector comprising the adaptive filter's variable coefficients at time index n, i.e.,

$$\hat{\mathbf{h}}(n) = \left[h_0(n),\ h_1(n),\ \ldots, h_{L-1}(n)\right]^T, \tag{1}$$

where ^{T} is the transpose of a matrix/vector. The output of the filter $y(n)$ is generated by performing the convolution between $\hat{\mathbf{h}}(n-1)$ and the L dimensional vector $\mathbf{x}(n)$ formed with the most recent input samples:

$$\mathbf{x}(n) = \left[x(n),\ x(n-1),\ \ldots, x(n-L+1)\right]^T. \tag{2}$$

Consequently, the error signal can be expressed as:

$$e(n) = d(n) - y(n) = d(n) - \hat{\mathbf{h}}^T(n-1)\mathbf{x}(n). \tag{3}$$

The core of the ANC system presented in Figure 1 is the adaptive algorithm. The usual methods employed for the update of $\hat{\mathbf{h}}(n)$ are the LMS-type adaptive algorithms, which have reduced performance when working with highly correlated input signals. In the ANC case, the samples of signal $x(n)$ can be associated with speech, music, engine noise or other (highly correlated) acoustic signals. In such circumstances, the RLS-based systems can generate superior performance (in comparison to the LMS class) through their de-correlation properties. Despite the attractive features of the RLS algorithms, the classical versions employ arithmetically costly methods for computing the corresponding matrix inverse and solving the associated system of equations. Consequently, excessive workloads are imposed on signal processing chips, which usually handle multiple tasks.

III. A Different Approach on the RLS Algorithm

The RLS-DCD adaptive algorithm was proposed as a stable alternative for other low-complexity RLS versions (such as the FRLS). Initially, the method was mostly employed for processing weakly correlated signals and later for the identification of long unknown acoustic systems (e.g., the AEC/SAEC scenarios) [7]-[11]. The corresponding least-squares cost function is defined as:

$$J(n) = \sum_{i=0}^{n} \lambda^{n-i}\left[d(i) - \hat{\mathbf{h}}^T(n)\mathbf{x}(i)\right]^2, \tag{4}$$

where we denote by λ $(0 << \lambda < 1)$ the forgetting factor associated with the memory of the algorithm [1]. The minimization of $J(n)$ requires the solution to the normal equations [2], [3]:

$$\mathbf{R}_x(n)\hat{\mathbf{h}}(n) = \mathbf{p}_{xd}(n), \tag{5}$$

where $\mathbf{R}_x(n)$ is the L x L correlation matrix of the input signal and $\mathbf{p}_{xd}(n)$ is an L x 1 vector. The matrix $\mathbf{R}_x(n)$ and vector $\mathbf{p}_{xd}(n)$ are known for every iteration of the adaptive filter and can be expressed recursively using the forgetting factor:

$$\mathbf{R}_x(n) = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^T(i) = \lambda\mathbf{R}_x(n-1) + \mathbf{x}(n)\mathbf{x}^T(n), \tag{6}$$

$$\mathbf{p}_{xd}(n) = \sum_{i=0}^{n} \lambda^{n-i}\mathbf{x}(i)d(i) = \lambda\mathbf{p}_{xd}(n-1) + d(n)\mathbf{x}(n). \tag{7}$$

The direct computation of the solution associated with (5) has an arithmetic complexity of $O(L^3)$ and is considered an impossible task even for the most advanced signal processing chips. In [7], [8] a new approach was proposed by transforming the normal equations (5) into an auxiliary system, which is solved using iterative methods. The goal is to express (5) as:

$$\mathbf{R}_x(n)\left[\hat{\mathbf{h}}(n-1) + \Delta\hat{\mathbf{h}}(n)\right] = \mathbf{p}_{xd}(n), \tag{8}$$

where $\Delta\hat{\mathbf{h}}(n) = \hat{\mathbf{h}}(n) - \hat{\mathbf{h}}(n-1)$ is regarded as the new

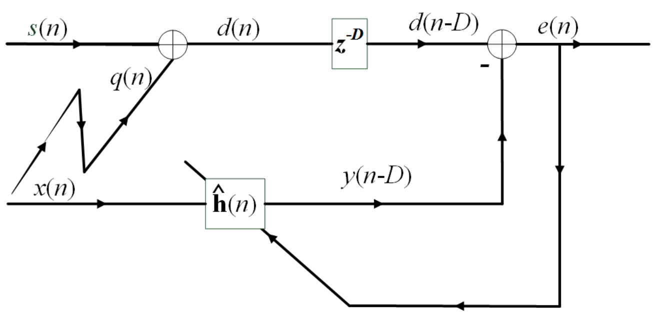

Figure 1. The ANC scheme

unknown (or *solution*) vector, which is used to update the adaptive filter through accumulation.

Considering that the solution for (5) is approximately known at time index n-1, a residual vector can be defined as [7], [8]:

$$\mathbf{r}(n-1) = \mathbf{p}_{xd}(n-1) - \mathbf{R}_x(n-1)\hat{\mathbf{h}}(n-1). \tag{9}$$

Additionally, the changes between consecutive iterations, corresponding to the elements in (5), can be denoted as:

$$\Delta\mathbf{R}_x(n) = \mathbf{R}_x(n) - \mathbf{R}_x(n-1), \tag{10}$$

$$\Delta\mathbf{p}_{xd}(n) = \mathbf{p}_{xd}(n) - \mathbf{p}_{xd}(n-1). \tag{11}$$

The auxiliary system of linear equations can be obtained by using (9), (10) and (11) to extract $\mathbf{R}_x(n)\Delta\hat{\mathbf{h}}(n)$ from (8) and to setup $\Delta\hat{\mathbf{h}}(n)$ as the new unknown vector:

$$\begin{aligned}\mathbf{R}_x(n)\Delta\hat{\mathbf{h}}(n) &= \mathbf{p}_{xd}(n) - [\mathbf{R}_x(n-1) + \Delta\mathbf{R}_x(n)]\hat{\mathbf{h}}(n-1)\\ &= \mathbf{p}_{xd}(n-1) + \Delta\mathbf{p}_{xd}(n) - \\ &\quad -\mathbf{R}_x(n-1)\hat{\mathbf{h}}(n-1) - \Delta\mathbf{R}_x(n)\hat{\mathbf{h}}(n-1)\\ &= \mathbf{r}(n-1) + \Delta\mathbf{p}_{xd}(n) - \Delta\mathbf{R}_x(n)\hat{\mathbf{h}}(n-1)\\ &= \mathbf{p}_0(n).\end{aligned} \tag{12}$$

Although the solution of (12) would require the inverse of the same matrix $\mathbf{R}_x(n)$ as in the case of (5), the reduction in arithmetic complexity is determined by the few values of $\Delta\hat{\mathbf{h}}(n)$, which can be computed for any time index n in order to achieve good convergence properties. It can also be noticed that (12) requires the residual vector corresponding to the n-1 time index. After several computations are performed, the values comprising $\mathbf{r}(n)$ can be expressed using the elements of the auxiliary system of equations [7], [8]:

$$\begin{aligned}\mathbf{r}(n) &= \mathbf{p}_{xd}(n) - \mathbf{R}_x(n)\hat{\mathbf{h}}(n)\\ &= \mathbf{p}_{xd}(n-1) + \Delta\mathbf{p}_{xd}(n)\\ &\quad -\mathbf{R}_x(n-1)\left[\hat{\mathbf{h}}(n-1) + \Delta\hat{\mathbf{h}}(n)\right] - \Delta\mathbf{R}_x(n)\hat{\mathbf{h}}(n)\\ &= \mathbf{r}(n-1) + \Delta\mathbf{p}_{xd}(n) - \mathbf{R}_x(n-1)\Delta\hat{\mathbf{h}}(n) - \Delta\mathbf{R}_x(n)\hat{\mathbf{h}}(n)\\ &= \mathbf{p}_0(n) - \mathbf{R}_x(n-1)\Delta\hat{\mathbf{h}}(n) - \Delta\mathbf{R}_x(n)\Delta\hat{\mathbf{h}}(n)\\ &= \mathbf{p}_0(n) - \mathbf{R}_x(n)\Delta\hat{\mathbf{h}}(n).\end{aligned} \tag{13}$$

In accordance with (6) and (7), after some algebra the values of $\mathbf{r}(n)$ can also be determined in a recursive manner, i.e.,

$$\mathbf{r}(n) = \lambda\mathbf{r}(n-1) + e(n)\mathbf{x}(n). \tag{14}$$

The approach described in the current section re-states the least-squares problem by targeting the computation of the variation associated with the adaptive filter's coefficients between two consecutive time indexes. Therefore, the number of significant values in $\Delta\hat{\mathbf{h}}(n)$ is considerably smaller than the entire set of coefficients corresponding to $\hat{\mathbf{h}}(n)$, which is directly computed in the classical RLS versions. A major reduction in arithmetic complexity can be achieved when using the DCD iterations.

IV. THE RLS-DCD ADAPTIVE ALGORITHM

The symmetric positive-definite property of the matrix $\mathbf{R}_x(n)$ makes possible the combination between the RLS algorithm and the DCD method [7]-[10]. We propose to use the resulting algorithm (i.e., the RLS-DCD - presented in Table I) for real time retrieval of speech signals in ANC scenarios.

In step 1 of the adaptive method the correlation matrix is updated by exploiting its transpose property, i.e., $\mathbf{R}_x(n) = \mathbf{R}_x^T(n)$. The modification is performed by copying the upper-left L-1 x L-1 block of $\mathbf{R}_x(n-1)$ to the lower-right L-1 x L-1 submatrix of $\mathbf{R}_x(n)$, and by computing only the first column [8]-[10]. Consequently, the complexity associated with step 1 is reduced to a value proportional to the length of the adaptive filter [8], [10]. The main diagonal of $\mathbf{R}_x(n)$ is initialized using the identity matrix $\mathbf{I}_L$ and the constant value δ, in order to avoid processing a singular matrix in the initial stages of the adaption course.

TABLE I. THE RLS-DCD ALGORITHM

Initialization	$\hat{\mathbf{h}}(0)=\mathbf{0},\ \mathbf{r}(0)=0,\ \mathbf{R}_{\mathbf{x}}(0)=\delta\mathbf{I}_L$	x	+
for n = 1, 2, …			
Step 1	$\mathbf{R}_{\mathbf{x}}^{(1)}(n)=\lambda\mathbf{R}^{(1)}(n-1)+x(n)\mathbf{x}(n)$	L	$2L$
Step 2	$e(n)=d(n)-\hat{\mathbf{h}}^T(n-1)\mathbf{x}(n)$	L	L
Step 3	$\mathbf{r}(n)=\lambda\mathbf{r}(n-1)+e(n)\mathbf{x}(n)$	L	$2L$
Step 4 (DCD)	$\mathbf{R}_x(n)\Delta\hat{\mathbf{h}}(n)=\mathbf{r}(n)\Rightarrow\Delta\hat{\mathbf{h}}(n),\mathbf{r}(n)$	0	$2N_uL+$ $+L+M_b$
Step 5	$\hat{\mathbf{h}}(n)=\hat{\mathbf{h}}(n-1)+\Delta\hat{\mathbf{h}}(n)$	0	L

After determining the error of the filter in step 2, the DCD portion of the algorithm is processed in stages 3 and 4. The residual vector $\mathbf{r}(n)$ is updated using the forgetting factor in the sense of correlating $e(n)$ with the input vector $\mathbf{x}(n)$. The classical system of equations is replaced by an auxiliary problem. Although, the same matrix requires an inversion, the statistical properties of the new model allow for much simpler operations. For each of the *maximum number of allowed* (or *successful*) *updates* (denoted by N_u), the DCD performs a search and an update for a value of the solution vector, which is initialized with zero. Considering that only the main diagonal of $\mathbf{R}_x(n)$ comprises contributions of squared numbers (positive numbers), then it is safe to assume that (statistically) the rest of the matrix has negligible quantities regarding the *decision process*. Accordingly, the choice of performing an update to $\Delta\hat{\mathbf{h}}(n)$ is based on finding the maximum absolute value in the residual vector and comparing it to the corresponding number (situated on the same position) on the main diagonal of $\mathbf{R}_x(n)$, which is scaled using half of the *step size*, denoted by α. If the first term of the comparison is larger, then an update is performed on the corresponding position of $\Delta\hat{\mathbf{h}}(n)$ and the residual vector is also modified to reflect the newest change in the solution vector. The values comprising $\Delta\hat{\mathbf{h}}(n)$ and $\hat{\mathbf{h}}(n)$ are considered to be represented using a fixed-point format with M_b bits.

Table II describes the behavior of the DCD iterations with a *leading element* [7]-[11]. The name of the method is associated to its *greedy* manner of searching the most probable locations where updates could be performed. The DCD is employed at step 4 in Table I, i.e., for each time index n. The key factor associated with the reduction in arithmetic workload is the choice of the step size α. The selection starts with the parameter H, which is the maximum expected amplitude of the values comprising the solution vector, i.e., the numbers in $\Delta\hat{\mathbf{h}}(n)$ are expected to be in the interval $(-H, H)$. Correspondingly, the value of α can be initialized with $H/2$ and is halved for each time a comparison fails to lead to a successful update. By choosing H as a power of 2, any multiplication with α can be performed on signal processing chips as a bit-shift. Moreover, the smaller the step size becomes, the closer to the Least Significant Bit (LSB) is the contribution added in $\Delta\hat{\mathbf{h}}(n)$ (the update operation can have positive or negative contributions). The complete update procedure is presented in Table II (the end of each *for iteration*). It comprises an adjustment performed for one the solution vector values and an entire residual vector modification. The values of $\mathbf{r}(n)$ tend to become smaller as the DCD produces updates to $\Delta\hat{\mathbf{h}}(n)$.

The DCD portion of the algorithms ends when one of two conditions is met [7]. Firstly, if N_u updates are performed, then the planned arithmetic effort is finished and the current values of $\mathbf{r}(n)$ and $\Delta\hat{\mathbf{h}}(n)$ are considered to be the produced results. Otherwise, if enough comparisons are unsuccessful, then the value of the step size becomes too small (i.e., the value of m corresponds to a value greater than the position of the LSB) and the DCD is stopped. Although in the last mentioned case the algorithm stops when $m>M_b$ and the number of performed updates is smaller than the planned value N_u, $\mathbf{r}(n)$ and $\Delta\hat{\mathbf{h}}(n)$ are still considered valid results and used by the RLS-DCD.

In any DCD ending situation, step 5 in Table I uses the determined solution vector to modify the adaptive filter coefficients $\hat{\mathbf{h}}(n-1)$ and generate the new vector corresponding to the adaptive algorithm, i.e., $\hat{\mathbf{h}}(n)$. It is important to mention that because $\Delta\hat{\mathbf{h}}(n)$ is altered for a maximum number of times equal to N_u, only several coefficients are really modified (for any given time index n), as the other remain with their initial values. However, the invested arithmetic effort is enough to generate sufficient performance, as already shown in [7]-[10] for AEC and SAEC scenarios.

Relevant information about arithmetic complexity is also displayed in Table I. It can be noticed that the DCD method uses no multiplications or divisions. The corresponding computational effort is influenced by the length of the adaptive filter L, by the number of *successful iterations* N_u (which is usually smaller than 10) and by the number of bits M_b used to represent the values comprising $\Delta\hat{\mathbf{h}}(n)$. The leading DCD employs only bit-shifts of the operands and no more than $(2N_u+1)L+M_b$ additions [7]-[9]. Considering that we propose to use the RLS-DCD algorithm for ANC scenarios, the value of L is expected to be comparable to N_u and M_b.

The overall complexity of the RLS-DCD can be further reduced by choosing the forgetting factor as $\lambda=1-1/(KL)$, where K and the filter length L are positive integers, powers of 2. Therefore, any multiplication with λ can be replaced by a bit-shift and one subtraction. The total amount of arithmetic operations corresponding to the algorithm described in Table I is represented by $3L$ multiplications and less than $6L+2N_uL+M_b$ additions for every time index n [8].

TABLE II. THE DCD WITH A LEADING ELEMENT

Initialization	$\Delta\hat{\mathbf{h}} = \mathbf{0},\ \alpha = H/2,\ m = 2$
for k = 1, 2, …N$_u$	
$[val,\ poz] = \max\{\lvert r_0(n)\rvert, \lvert r_1(n)\rvert \ldots \lvert r_{L-1}(n)\rvert\}$ $v = val;\ p = poz;$	
while $\left(\left(v \le (\alpha/2)R_{x;\,p,p}(n)\right)$ ***and*** $\left(m \le M_{\mathrm{b}}\right)\right)$ $m = m+1;\ \ \alpha = \alpha/2;$ ***end***	
if $(m>M_b)$ ***exit DCD***	
$\hat{h}_p(n) = \hat{h}_p(n) + sign\{r_p(n)\}\alpha$ $\mathbf{r}(n) = \mathbf{r}(n) - sign\{r_p(n)\}\alpha\mathbf{R}_x^{(p)}(n)$ ***end for iteration***	

We notice that the value of M_b has a limited influence on the number of arithmetic operations. However, the parameter is relevant for their complexity.

V. SIMULATIONS

Simulations results are presented for the context illustrated in Figure 1, using the RLS-DCD and RLS adaptive algorithms. The performance of the ANC system is analyzed using time domain plots and spectrograms with 256 points Fourier Transforms for the generated error signals. The reference RLS method employs Woodburry's identity to estimate the inverse of the correlation matrix and to solve the classical least-squares system of equations.

The acoustic test signals are sampled with a frequency of 8 kHz, using 16 bits/sample. The goal is to recover interference-free speech sequences available in the $s(n)$ waveforms [12]. The desired signal is generated by filtering the interference $x(n)$ with a Matlab generated low-pass filter and adding the output $q(n)$ to $s(n)$. The Matlab filter is a *fir1* impulse response with 13 coefficients and a cut-off frequency of 0.475 of the sampling frequency.

The length of the adaptive filter is L=25 and the corresponding forgetting factor is set to $\lambda = 1 - 1/(16L)$. Correspondingly, the L values comprising the RLS-DCD solution vector are represented in the numerical interval (-H, H)=(-1,1) using M_b bits. The parameter M_b directly influences the precision of the ANC system and is varied in order to study the compromise between the performance and complexity. Furthermore, $\Delta\hat{\mathbf{h}}(n)$ is updated for a maximum number of N_u=4 times per every time index n.

The first simulation compares the performance of the RLS-DCD and RLS algorithms using Gaussian noise as acoustic interference. The $s(n)$ and $q(n)$ signals have the same power (i.e., the corresponding SNR has the value 0 dB). It can be noticed in Figure 2 that increasing the number of bits used for the representation of the adaptive filter coefficients leads to better estimates of interference samples and a better reduction in noise level. Additionally, the comparison performed with the RLS spectrogram indicates

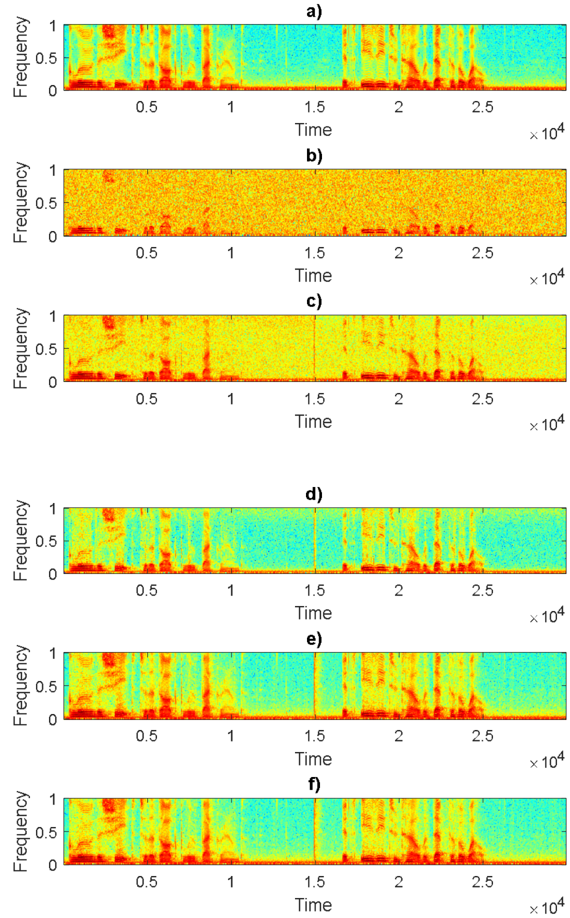

Figure 2. Spectrograms with 256 Fourier Transforms – the interference is Gaussian noise (SNR=0 dB): a) The speech sequence to be recovered; RLS-DCD error signal with b) M_b=3, c) M_b=6, d) M_b=8, e) M_b=16; f) RLS error signal

that higher values of the parameter M_b provide similar performance from the RLS-DCD method, with lower arithmetic effort. Figures 3 and 4 provide results in the time domain for the scenario. The original speech is compared to the corrupted signal (Figure 3) and the recovered sequences (i.e., the error signals) are displayed for the RLS method, respectively the RLS-DCD algorithm with M_b=16 (Figure 4). Both plots in Figure 4 indicate an obvious reduction in noise level.

For the second simulation (Figure 5), the interference signal $x(n)$ is acoustic engine noise. The same value is used for the SNR (0 dB). In comparison to the previous scenario, it can be noticed that the settings M_b=8 and M_b=16 do not provide the same performance rating anymore. The properties of the second interference type require more precision in order to generate similar results between the RLS-DCD and the RLS methods. The time domain signals illustrated in Figures 6 and 7 also demonstrate that the RLS-DCD algorithm can provide similar performance with the RLS using considerably less arithmetic resources.

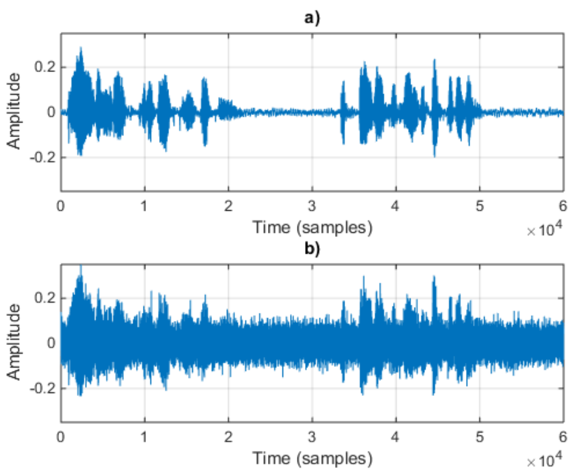

Figure 3. Acoustic signals in the time domain: a) The speech sequence to be recovered; b) Corrupted speech signal: Gaussian noise with SNR=0 dB

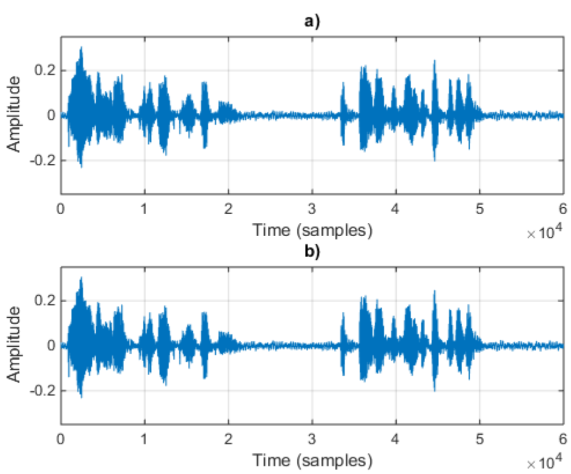

Figure 4. Acoustic signals in the time domain; Gaussian noise interference (SNR=0 dB) a) The recovered speech signal using RLS-DCD with M_b=16; b) The recovered signal using RLS

The spectrograms corresponding to a third experiment are illustrated in Figure 8. The speech $s(n)$ is corrupted for the first half of the simulation by engine sound, which is afterwards replaced by music. The SNR is set to -10 dB for the entire scenario. The change in interference produces a spike in each error spectrogram and the adaptive algorithms require an adaptation period. It can also be noticed that the music is harder to eliminate from the desired signal (the corresponding interference leaves easier noticeable traces in the error signal). As a consequence, the correlation properties of the interference signals have an important influence on the performance of the adaptive algorithms.

The performance of the adaptive filter is also illustrated in the time domain (Figures 9 and 10), for scenario 3. The change in interference type [see Figure 9.b)] generates short spikes in the error signals plotted in Figure 10. It can be noticed that the reaction time of the RLS-DCD adaptive method is similar to the RLS.

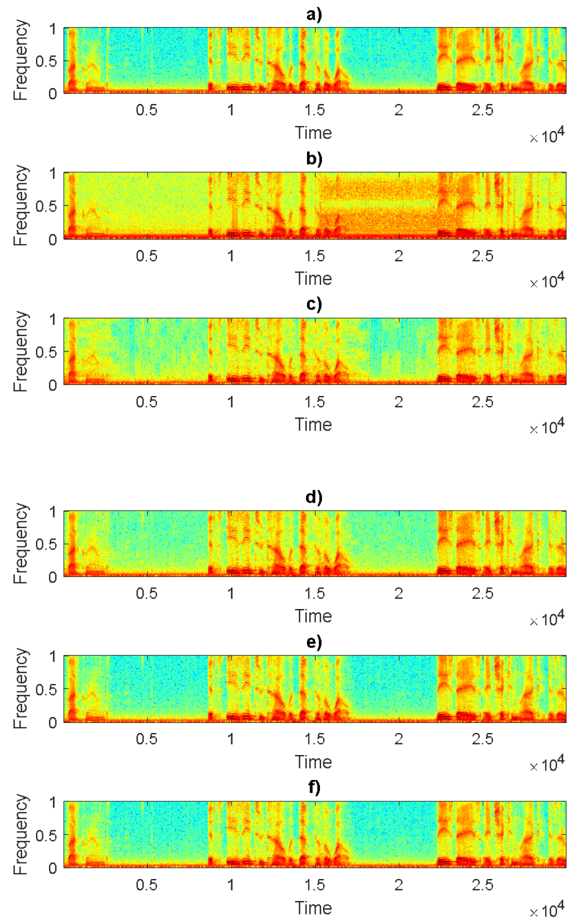

Figure 5. Spectrograms with 256 Fourier Transforms – the interference is engine noise (SNR=0 dB): a) The speech sequence to be recovered; RLS-DCD error signal with b) M_b=3, c) M_b=6, d) M_b=8, e) M_b=16; f) RLS error signal

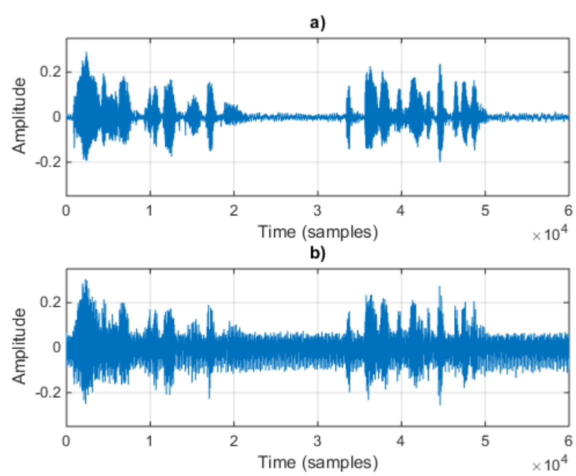

Figure 6. Acoustic signals in the time domain: a) The speech sequence to be recovered; b) Corrupted speech signal: engine noise with SNR=0 dB

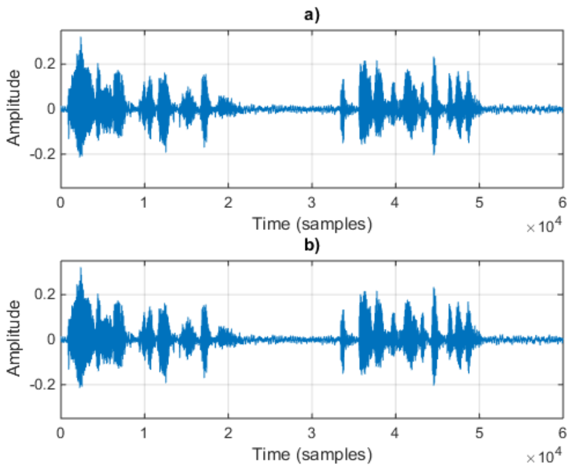

Figure 7. Acoustic signals in the time domain; engine noise interference (SNR=0 dB) a) The recovered speech signal using RLS-DCD with M_b=16; b) The recovered signal using RLS

VI. Conclusions

This paper has presented a low-complexity RLS algorithm for ANC scenarios. The RLS-DCD adaptive method is based on a reinterpretation of the classical least-squares problem and replaces the standard system of linear equations with an auxiliary system. The new unknown vector (i.e., the solution vector) represents the changes of the adaptive filter between consecutive iterations. The model favors low-complexity solutions, such as the leading DCD iterations, which computes a low number of values for the update stage of the filter's coefficients.

The DCD method exploits the statistical properties of the correlation matrix associated with the input signal and solves the proposed auxiliary system using only bit-shifts and additions. Moreover, the overall combination between the RLS and the DCD requires no divisions and the multiplication volume can be drastically reduced by choosing an appropriate value for the forgetting factor. The RLS-DCD is also a stable alternative, which has comparable performance with the RLS method implemented using Woodburry's identity.

Simulations have been performed in order to analyze the behavior of the proposed system. The results indicate that the RLS-DCD has attractive performance when working with correlated signals, such as speech. The computational efficiency of the proposed adaptive algorithm recommends it as a suitable candidate for ANC implementations on signal processing chips for mobile devices.

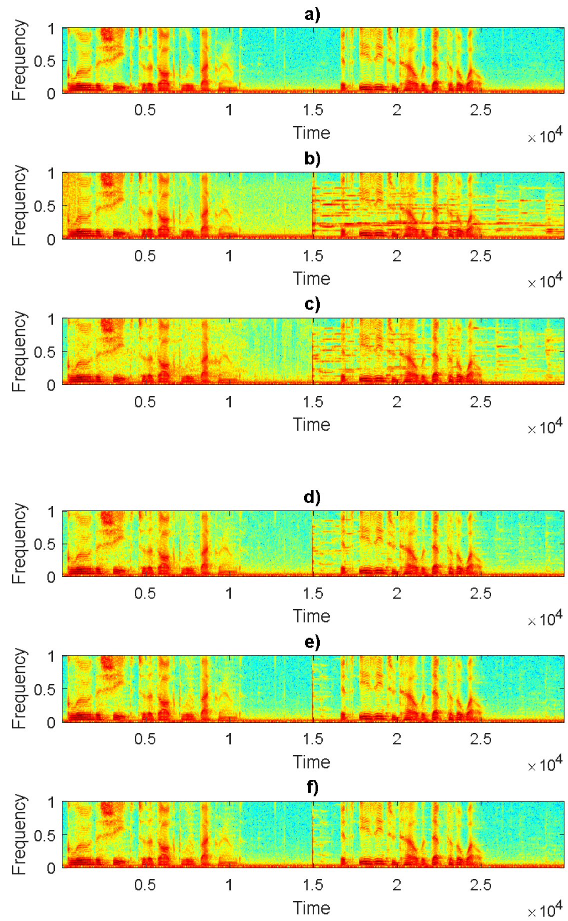

Figure 8. Spectrograms with 256 Fourier Transforms - the interference is engine noise, which changes to music at time index 15000 (SNR=-10 dB): a) The speech sequence to be recovered; RLS-DCD error signal with b) M_b=3, c) M_b=6, d) M_b=8, e) M_b=16; f) RLS error signal

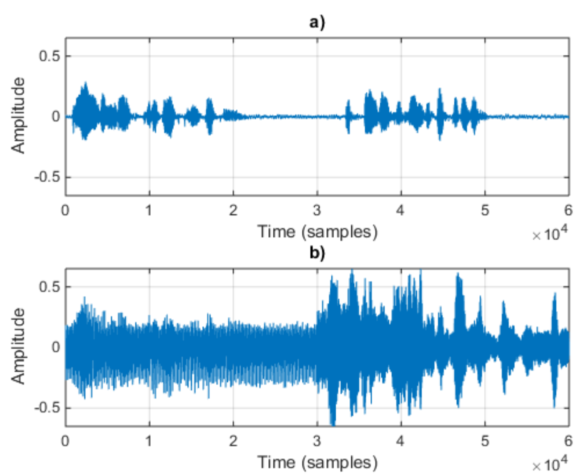

Figure 9. Acoustic signals in the time domain: a) The speech sequence to be recovered; b) Corrupted speech signal: engine noise (first half) and music (second half); SNR=0 dB

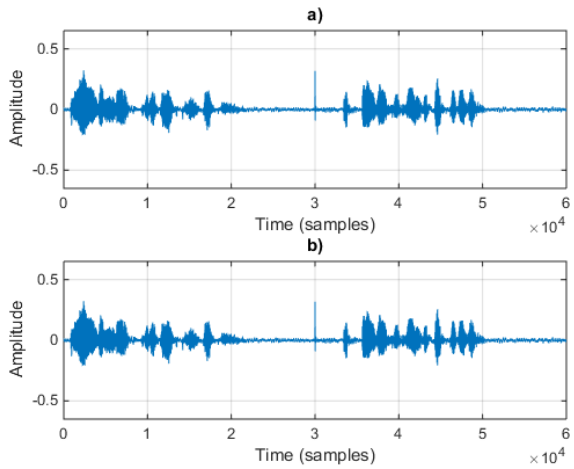

Figure 10. Acoustic signals in the time domain; interference: engine noise (first half) and music (second half); SNR=0 dB a) The recovered speech signal using RLS-DCD with M_b=16; b) The recovered signal using RLS

ACKNOWLEDGMENT

This work was supported by European Space Agency under Grant no. 4000121222/17/NL/CBi and University Politehnica of Bucharest under Grant no. 18/05.10.2017.

REFERENCES

[1] C. Stanciu, L. Stanciu, and R. Mihăescu, "Low Complexity Recursive Least-Squares Algorithm for Adaptive Noise Cancellation," in *Proc. ICN*, 2017, pp. 102–106.

[2] S. Haykin, *Adaptive Filter Theory*. Fourth Edition, Upper Saddle River, NJ: Prentice-Hall, 2002.

[3] E. Hänsler and G. Schmidt, *Acoustic Echo and Noise Control – A Practical Approach*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[4] A. H. Sayed, *Adaptive Filters*. New York, NY: Wiley, 2008.

[5] J. Benesty, C. Paleologu, T. Gänsler, and S. Ciochină, *A Perspective on Stereophonic Acoustic Echo Cancellation*, Springer-Verlag, Berlin, Germany, 2011.

[6] B. Farhang-Boroujeny, *Adaptive Filters - Theory and Applications*. Second Edition, John Wiley & Sons, Inc., Hoboken, New Jersey, 2013.

[7] Y. V. Zakharov and T. C. Tozer, "Multiplication-free iterative algorithm for LS problem," *IEE Electronics Lett.*, vol. 40, pp. 567–569, Apr. 2004.

[8] Y. V. Zakharov, G. P. White, and J. Liu, "Low-complexity RLS algorithms using dichotomous coordinate descent iterations," *IEEE Trans. Signal Processing*, vol. 56, pp. 3150–3161, July 2008.

[9] J. Liu, Y. V. Zakharov, and B. Weaver, "Architecture and FPGA design of dichotomous coordinate descent algorithms," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 56, pp. 2425–2438, Nov. 2009.

[10] C. Stanciu, J. Benesty, C. Paleologu, T. Gänsler, and S. Ciochină, "A widely linear model for stereophonic acoustic echo cancellation," *Signal Processing*, vol. 93, pp. 511–516, Feb. 2013.

[11] C. Stanciu and C. Anghel, "Numerical of the DCD-RLS Algorithm for Stereo Acoustic Echo Cancellation," in *Proc. COMM*, 2014, pp.65–68.

[12] The Open Speech Repository, http://www.voiptroubleshooter.com/open_speech/ [last accessed: 02.09.2017].