A FAMILY OF RECURSIVE LEAST-SQUARES ADAPTIVE ALGORITHMS SUITABLE FOR FIXED-POINT IMPLEMENTATION

Constantin Paleologu, Silviu Ciochină, and Andrei Alexandru Enescu

Telecommunications Department, University Politehnica of Bucharest, Romania
e-mail: {pale, silviu, aenescu}@comm.pub.ro

ABSTRACT

The main feature of the least-squares adaptive algorithms is their high convergence rate. Unfortunately, they encounter numerical problems in finite precision implementation and especially in fixed-point arithmetic. The objective of this paper is twofold. First, an analysis of the finite precision effects of the recursive least-squares (RLS) algorithm is performed, outlining some specific problems that could appear in fixed-point implementation; consequently, we present a modified version of the RLS algorithm suitable for fixed-point implementation, using an asymptotically unbiased estimator for the algorithm's cost. Second, we extend the procedure for the case of QR-decomposition-based least-squares lattice (QRD-LSL) adaptive algorithm, a "fast" member of RLS family, with good numerical properties. The reduced dynamics of the algorithm's parameters leads to facility for fixed-point implementation. The simulations performed on a fixed-point digital signal processor (DSP) sustain the theoretical findings. Also, as a practical aspect of this work, we illustrate the performance of the proposed QRD-LSL algorithm for noise reduction.

Index Terms— Adaptive filters, fixed-point implementation, noise reduction, QR-decomposition-based least-squares lattice (QRD-LSL) algorithm, recursive least-square (RLS) algorithm.

1. INTRODUCTION

The Recursive Least Squares (RLS) algorithm is one of the most popular adaptive algorithms, mainly due to its fast convergence rate [1]. Nevertheless, there are some major drawbacks related to the high computational complexity and the large dynamic range of the algorithm's variables. The first issue could be overcome by using a fast RLS algorithm, in the meaning that the computational cost increases linearly with the number of adjustable parameters. The last drawback is more severe and could cause unwanted effects in a fixed-point arithmetic context, such as overflow or stalling phenomena [2]. In this paper we focus on some numerical problems of the RLS algorithm and present a modified version of this algorithm, which is more suitable for fixed-point implementation. For practical reasons, the proposed procedure is applied to the QR-decomposition-based least-squares lattice (QRD-LSL) algorithm, which is a fast member of the RLS family with robust numerical behavior.

The QRD-LSL algorithm [3] combines the good numerical properties of QR-decomposition and the desirable features of a recursive least-squares lattice. Whereas the recursive QR-decomposition-based recursive least-squares (QRD-RLS) algorithm [1] requires a high computational load on the order of L^2 (where L is the filter order), in terms of both the number of processing cells and the computation per iteration, the QRD-LSL implementation is fast in the sense that these numbers are reduced to a linear dependence on L. This algorithm exploits the shifting property of serialized input data (the Toeplitz structure of the data matrix) to perform joint-process estimation in a fast manner. By virtue of this facts, the QRD-LSL algorithm is endowed with a highly desirable set of operational and implementation characteristics such as good numerical properties (inherited from QR-decomposition), good convergence properties (due to the RLS nature), and a high level of computational efficiency (resulted from the modular, lattice-like structure). The combination of these characteristics makes the QRD-LSL a powerful adaptive algorithm, suitable for a wide range of applications, e.g., echo cancellation, interference rejection, or noise reduction [4]–[10].

Another implication of the modular structure of the QRD-LSL algorithm is that it lends itself to the use of very large-scale integration (VLSI) technology for its hardware implementation. Of course, the use of this sophisticated technology can be justified only if the application of interest calls for the use of VLSI chips in large number. Otherwise, a digital signal processor (DSP) implementation represents a proper solution. In this case, an important practical aspect is related to the dynamic range of the algorithm's parameters. It is known that in two-complement fixed-point implementation context the

This work was supported by the UEFISCSU Romania under Grants PN-II-"Idei" no. 65/01.10.2007 and no. 331/01.10.2007.

absolute values of all involved parameters have to be smaller than 1. In the case of the classical QRD-LSL algorithm the cost functions asymptotically increase; theoretically, they are upper bounded by $1/(1-\lambda)$, where λ is the exponential weighting factor ($0 < \lambda \leq 1$) [1]. When dealing with a value of λ very close to 1 (which is the case in most of the applications due to stability reasons [11]), very large values of the cost functions are expected. In order to prevent any unwanted overflow phenomenon it is necessary to scale the cost function. As a consequence, the major drawback is the precision loss because of these factors.

In the first part of this work we analyze the behavior of the RLS algorithm in fixed-point arithmetic, revealing some specific problems that could appear in this context. In order to overcome these potential issues we present a modified version of the RLS algorithm that is suitable for fixed-point implementation. The main idea is to use an asymptotically unbiased estimator for the algorithm's cost function, in order to reduce the dynamic range of this parameter. The idea can be applied to other RLS-based algorithms. In this paper we extend the procedure for the case of QRD-LSL algorithm. Consequently, a modified version of this algorithm is obtained. The reduced dynamic of the parameters leads to facilities for fixed-point implementation.

The paper is organized as follows. Section 2 contains certain backgrounds of the classical RLS algorithm, outlining several specific problems that could appear in fixed-point implementation. In Section 3 we establish a connection between the dynamic range of variables and the initial convergence rate, and we present a modified version of the RLS algorithm suitable for fixed-point implementation. The modified version of the QRD-LSL algorithm is developed in Section 4. The simulation results are presented in Section 5. A summarized discussion of the main results is given in Section 6. Finally, Section 7 briefly concludes this work.

2. RLS ALGORITHM BACKGROUND AND ANALYSIS

The well-known RLS adaptive algorithm [1] uses a cost function defined as an estimate of the mean square-error, i.e.,

$$J(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| e(i) \right|^2 = \lambda J(n-1) + \left| e(n) \right|^2, \qquad (1)$$

where $0 < \lambda \leq 1$ is the exponential weighting factor and $e(i)$ is the difference between the desired response $d(i)$ and the output $y(i)$ produced by an adaptive transversal filter. That is,

$$e(i) = d(i) - y(i) = d(i) - \mathbf{w}^H(n)\mathbf{x}(i), \qquad (2)$$

where $\mathbf{x}(i)$ is the tap-input vector at time i and $\mathbf{w}(n)$ is the tap-weight vector at time n. The superscript H denotes Hermitian transposition (transposition and complex conjugation).

This estimate of the cost function induces similar estimates for the correlation matrix $\mathbf{\Phi}(n)$ and the cross-correlation vector $\boldsymbol{\theta}(n)$, i.e.,

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i)\mathbf{x}^H(i) = \lambda \mathbf{\Phi}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n), \quad (3)$$

$$\boldsymbol{\theta}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i) d^*(i) = \lambda \boldsymbol{\theta}(n-1) + \mathbf{x}(n) d^*(n), \quad (4)$$

where superscript * denotes complex conjugation. The optimum value of the tap-weight vector $\mathbf{w}(n)$, for which the cost function $J(n)$ from (1) attains its minimum value is defined by the normal equation written in matrix form:

$$\mathbf{\Phi}(n)\mathbf{w}(n) = \boldsymbol{\theta}(n). \qquad (5)$$

The regular procedure is to apply the matrix inversion lemma in (3) in order to solve (5). Denoting

$$\mathbf{P}(n) = \mathbf{\Phi}^{-1}(n) \qquad (6)$$

and defining the Kalman vector

$$\mathbf{k}(n) = \frac{1}{\lambda} \cdot \frac{\mathbf{P}(n-1)\mathbf{x}(n)}{1 + \frac{1}{\lambda}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)} \qquad (7)$$

the inverse of the estimate of the correlation matrix is computed in a recursive manner as [1]

$$\mathbf{P}(n) = \frac{1}{\lambda}\mathbf{P}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1). \qquad (8)$$

Finally, the recursive equation for updating the tap-weight vector is

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n), \qquad (9)$$

where $\alpha(n)$ is the a priori estimation error defined by

$$\alpha(n) = d(n) - \mathbf{w}^H(n-1)\mathbf{x}(n) \qquad (10)$$

The initial value of $\mathbf{P}(n)$ is chosen

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I}, \tag{11}$$

where δ is the regularization parameter (a positive constant) and $\mathbf{I}$ is the identity matrix. This initial value assures the non-singularity of the correlation matrix $\mathbf{\Phi}(n)$. In the case of a stationary environment or a slowly time-varying one, the parameter δ should be assigned a small value for high signal-to-noise ratio (SNR) and a large value for low SNR [12].

Next, in order to analyze the behavior of the RLS algorithm in finite precision implementation, let us examine some of its main parameters. Following (1) and (3), the expectations of the cost function $J(n)$ and of the matrix $\mathbf{\Phi}(n)$ are

$$E\{J(n)\} \cong \frac{1-\lambda^n}{1-\lambda} E\left\{\left|e(n)\right|^2\right\}, \tag{12}$$

$$E\{\mathbf{\Phi}(n)\} \cong \frac{1-\lambda^n}{1-\lambda}\mathbf{R}, \tag{13}$$

where $\mathbf{R}$ is the correlation matrix of input data. It can be noticed that $J(n)$ is a biased estimate of $E\{|e(n)|^2\}$ and similarly $\mathbf{\Phi}(n)$ is a biased estimate of $\mathbf{R}$, i.e.,

$$E\{J(n)\}\Big|_{n\to\infty} \cong \frac{1}{1-\lambda} E\left\{\left|e(n)\right|^2\right\}, \tag{14}$$

$$E\{\mathbf{\Phi}(n)\}\Big|_{n\to\infty} \cong \frac{1}{1-\lambda}\mathbf{R}. \tag{15}$$

Some classes of applications, e.g., [6], [7] require a high memory algorithm, which means that the value of the exponential weighting factor λ is very close to 1. In this case very large values for the parameters from (14) and (15) could result, causing unwanted finite precision effects in a practical implementation. Apparently, the RLS algorithm avoids this problem by using the inverse of the matrix $\mathbf{\Phi}(n)$. Therefore, the maximum values of the elements of the matrix $\mathbf{P}(n)$ result in the initialization phase of the algorithm, according to (11). Nevertheless, the "reverse" problem persists because the values of the elements of the matrix $\mathbf{P}(n)$ decrease towards very small values close to zero, when λ is close to 1.

For example let us consider the following scenario. The fixed point two's complement arithmetic with a word length of $B + 1$ bits is used and the input signal is a white Gaussian noise, so that $\mathbf{R} = \sigma_x^2\mathbf{I}$, where σ_x^2 is the input signal variance. We assume the input signal power upper bounded, so that

$$\sigma_x^2 \le a, \tag{16}$$

where a is a positive constant. In addition, for the RLS algorithm to work, a persistent excitation condition [13] must be imposed, i.e.,

$$\sigma_x^2 \ge b, \tag{17}$$

where b is a positive constant. Following (13) results

$$\mathbf{\Phi}(n)\Big|_{n\to\infty} = \frac{\sigma_x^2}{1-\lambda}\mathbf{I}, \tag{18}$$

$$\mathbf{P}(n)\Big|_{n\to\infty} = \frac{1-\lambda}{\sigma_x^2}\mathbf{I}. \tag{19}$$

Consequently, the elements of the main diagonal of $\mathbf{P}(n)$, denoted here by $\mathbf{P}_{(i,i)}(n)$, are asymptotically bounded by

$$\frac{1-\lambda}{a} \le \mathbf{P}_{(i,i)}(n) \le \frac{1-\lambda}{b}. \tag{20}$$

On the other hand, according to (11), it results that

$$\mathbf{P}_{(i,i)}(0) = \frac{1}{\delta}. \tag{21}$$

Therefore, a scaling procedure is required to avoid overflow phenomenon. The scaling factor $0 < s < 1$ has to be chosen such that

$$sM < 1, \tag{22}$$

where

$$M = \max\left\{\frac{1}{\delta}, \frac{1-\lambda}{b}\right\}. \tag{23}$$

Nevertheless, reducing the values of the elements of $\mathbf{P}(n)$ by scaling may lead to a stalling phenomenon. This phenomenon appears when $\mathbf{P}(n)$ becomes a zeros matrix, so that, according to (8), the RLS algorithm is "frozen". To avoid this situation it is necessary that

$$s\frac{1-\lambda}{a} > 2^{-B}. \tag{24}$$

Consequently,

$$\frac{a\cdot 2^{-B}}{1-\lambda} < s < \frac{1}{M} \tag{25}$$

and that implies

$$2^B > \frac{aM}{1-\lambda}. \tag{26}$$

In (23) the case $(1 - \lambda)/b > 1/\delta$ is improbable for a value of λ very close to 1, so that usually $M = 1/\delta$ and the algorithm is not sensitive to decrease of excitation.

3. MODIFIED RLS COST FUNCTION

Taking into account the previous discussion it would be very helpful to use an unbiased estimator of the matrix $\mathbf{\Phi}(n)$. For this reason, the cost function from (1) can be modified as follows [14]:

$$\bar{J}(n) = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\left|e(i)\right|^2 = \lambda\bar{J}(n-1) + (1-\lambda)\left|e(n)\right|^2. \tag{27}$$

In this case

$$E\left\{\bar{J}(n)\right\} \cong (1-\lambda^n)E\left\{\left|e(n)\right|^2\right\} \tag{28}$$

is an asymptotically unbiased estimator of the mean square-error.

Following this idea we have to perform the same modification in (3) and (4) obtaining

$$\begin{aligned}\bar{\mathbf{\Phi}}(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i) = \\ &= \lambda\bar{\mathbf{\Phi}}(n-1) + (1-\lambda)\mathbf{x}(n)\mathbf{x}^H(n)\end{aligned} \tag{29}$$

$$\begin{aligned}\bar{\boldsymbol{\theta}}(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)d^*(i) = \\ &= \lambda\bar{\boldsymbol{\theta}}(n-1) + (1-\lambda)\mathbf{x}(n)d^*(n)\end{aligned} \tag{30}$$

According,

$$E\left\{\bar{\mathbf{\Phi}}(n)\right\} \cong (1-\lambda^n)\mathbf{R} \tag{31}$$

is an asymptotically unbiased estimator of the correlation matrix.

As a consequence of these modifications, the Kalman vector from (7) has to be evaluated as [14]

$$\bar{\mathbf{k}}(n) = \frac{\bar{\mathbf{P}}(n-1)\mathbf{x}(n)}{\frac{\lambda}{1-\lambda} + \mathbf{x}^H(n)\bar{\mathbf{P}}(n-1)\mathbf{x}(n)}, \tag{32}$$

where $\bar{\mathbf{P}}(n)$ is the inverse of the matrix $\bar{\mathbf{\Phi}}(n)$. The others algorithm's relations remain the same [i.e., equations (8), (9), and (10)].

Next, let us perform a brief convergence analysis of this modified RLS algorithm. First we assume that the desired response $d(i)$ and the tap-input vector $\mathbf{x}(i)$ [see (2)] are related by the linear regression model

$$d(i) = \mathbf{w}_0^H\mathbf{x}(i) + e_0(i), \tag{33}$$

where $\mathbf{w}_0$ is the regression parameter vector of the model and $e_0(i)$ is the measurement noise, assumed to be white with zero mean and variance σ_0^2, and independent of $\mathbf{x}(i)$. As a result of the initialization procedure, (29) becomes

$$\bar{\mathbf{\Phi}}(n) = \lambda^n\delta\mathbf{I} + \bar{\mathbf{\Phi}}_0(n), \tag{34}$$

where $\bar{\mathbf{\Phi}}_0(n)$ is a particular solution, i.e.,

$$\bar{\mathbf{\Phi}}_0(n) = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i). \tag{35}$$

Using (34), (35), (33), and (30) in (5) we get

$$\begin{aligned}&(1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)\mathbf{w}(n) + \lambda^n\delta\mathbf{w}(n) = \\ &\quad = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)\mathbf{w}_0 + (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)e_0^*(i)\end{aligned} \tag{36}$$

Taking the expectation of both sides of (36) and taking into account that

$$E\left\{\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)e_0^*(i)\right\} = \mathbf{0}, \tag{37}$$

$$E\left\{\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)\right\} = \frac{\mathbf{R}}{1-\lambda}, \tag{38}$$

we obtain

$$(\mathbf{R} + \lambda^n\delta\mathbf{I})E\left\{\mathbf{w}(n)\right\} = \mathbf{R}\mathbf{w}_0. \tag{39}$$

Therefore,

$$E\left\{\mathbf{w}(n)\right\}\Big|_{n\to\infty} = \mathbf{w}_0 \tag{40}$$

so that $\mathbf{w}(n)$ is an asymptotically unbiased estimator of $\mathbf{w}_0$. The initial convergence rate of the modified algorithm depends on how "fast" the product $\lambda^n\delta$ decreases to zero.

Performing the same analysis for the classical RLS algorithm we get

$$\left(\mathbf{R}+\left(1-\lambda\right)\lambda^{n}\delta\mathbf{I}\right)E\left\{\mathbf{w}\left(n\right)\right\}=\mathbf{R}\mathbf{w}_{0}. \quad (41)$$

Comparing (39) to (41) it is obvious that the classical RLS algorithm has a faster initial convergence rate than the modified algorithm for the same λ and δ. Anyway, if we use for the classical RLS algorithm a value of the regularization parameter equal to δ and for the modified algorithm the value

$$\delta'=\delta\left(1-\lambda\right) \quad (42)$$

both RLS algorithms achieve the same initial convergence rate.

Finally, let us analyze the dynamic range of the elements of the matrix $\bar{\mathbf{P}}(n)$. In a similar manner as in the case of the classical RLS algorithm we find

$$M'=\max\left\{\frac{1}{\delta'},\frac{1}{b}\right\}, \quad (43)$$

$$2^{-B}a<s<\frac{1}{M'}, \quad (44)$$

so that

$$2^{B}>aM'. \quad (45)$$

For $\delta'=\delta$, the probability of stalling phenomenon becomes significantly lower. Nevertheless, if M' is chosen as $M'=1/\delta'$, the probability of overflow due to small signal becomes higher.

It can be concluded that the fast initial convergence rate, being dependent on δ, is conditioned by the numerical resolution. Even in the modified version, the estimate of the mean square error is biased at the beginning of the process and similar property results for the matrix $\bar{\mathbf{\Phi}}(n)$. In addition, the initial value $\mathbf{P}(0)$ from (11) is an arbitrary starting point, without any relation with the input signal, but important for the initial convergence rate. We propose a more natural way for initialization of the cost function, suggested by the well-known averaging algorithm

$$\tilde{J}\left(n\right)=\begin{cases}\dfrac{n-1}{n}\tilde{J}\left(n-1\right)+\dfrac{1}{n}\left|e\left(n\right)\right|^{2}, & 1\le n\le N\\ \dfrac{N-1}{N}\tilde{J}\left(n-1\right)+\dfrac{1}{N}\left|e\left(n\right)\right|^{2}, & n>N\end{cases} \quad (46)$$

In the first N steps, $\tilde{J}(n)$ is the sample mean of $|e(n)|^2$, i.e.,

$$\tilde{J}\left(n\right)=\frac{1}{n}\sum_{i=1}^{n}\left|e\left(i\right)\right|^{2} \quad (47)$$

Alternatively, the above cost function can be written as

$$\tilde{J}\left(n\right)=\lambda\left(n\right)\tilde{J}\left(n-1\right)+\left(1-\lambda\left(n\right)\right)\left|e\left(n\right)\right|^{2} \quad (48)$$

where

$$\lambda\left(n\right)=\begin{cases}\dfrac{n-1}{n} & \text{for} \quad 1<n\le\dfrac{1}{1-\lambda}\\ \lambda & \text{for} \quad n>\dfrac{1}{1-\lambda}\end{cases} \quad (49)$$

For $n = 1$, $\lambda(1) = 0$ is not acceptable so that the iterations start from $n = 2$, considering the initial value

$$\bar{\mathbf{P}}\left(1\right)=x\left(0\right)^{-2}\mathbf{I} \quad (50)$$

In order to agree assumption (17) the algorithm starts only when $x(0) > b$. The initial convergence rate depends on the starting value $x^2(0)$ but this dependence is considerably reduced because of the low memory behavior (i.e., small λ) in the initial part of the process. The main advantage of the algorithm consists of the fact that $\bar{\mathbf{\Phi}}(n)$ is an unbiased estimator of $\mathbf{R}$, almost every time.

4. QRD-LSL ALGORITHMS WITH REDUCED DYNAMICS OF PARAMETERS

The classical RLS is not very frequently used in practical application mainly due to its high computational complexity (on the order of L^2). For this reason, the fast RLS algorithms are preferred in practice (e.g., [15]). Among these, the QRD-LSL algorithm represents one of the most attractive choices, mainly due to its robust numerical features [9].

In order to extend the idea of the modified cost function from Section 3 to the case of the QRD-LSL algorithm, let us consider the time series $x(1),x(2),\ldots,x(n)$ (i.e., the input signal) that occupies the time interval $1\le i\le n$, assuming that $x(i)=0$ for $i\le 0$. Most of the notations from [1] will be involved in the following development. The data matrix used in a least-squares estimation problem can be expressed as

$$\mathbf{A}_{m+1}(n)=\begin{bmatrix}x^{*}(1) & 0 & 0\\ \mathbf{d}_{f,m-1}(n-1) & \mathbf{A}_{m-1}(n-2) & \mathbf{d}_{b,m-1}(n-2)\\ x^{*}(n) & \mathbf{x}_{m-1}^{H}(n-1) & x^{*}(n-m)\end{bmatrix}, \quad (51)$$

where subscript $m = 1, 2, \ldots, L$ is the prediction order and

$$\mathbf{A}_{m-1}(n-2) = \begin{bmatrix} x^*(1) & 0 & \cdots & 0 \\ x^*(2) & x^*(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x^*(n-2) & x^*(n-1) & \cdots & x^*(n-m) \end{bmatrix}, \quad (52)$$

$$\mathbf{x}_{m-1}(n-1) = \left[x(n-1), \ldots, x(n-m+1)\right]^T, \quad (53)$$

$$\mathbf{d}_{f,m-1}(n-1) = \left[x^*(2), \ldots, x^*(n-1)\right]^T, \quad (54)$$

$$\mathbf{d}_{b,m-1}(n-2) = \left[0, \ldots, x^*(n-m-1)\right]^T. \quad (55)$$

where superscript T denotes transposition. Let us denote by $\mathbf{Q}_m(n)$ an n-by-n unitary matrix and by $\mathbf{R}_m(n)$ an $(m+1)$-by-$(m+1)$ upper triangular matrix. The exponential weighting matrix from the classical QR-decomposition is

$$\mathbf{\Lambda}_c(n) = \text{diag}\left\{\lambda^{n-1}, \lambda^{n-2}, \ldots, 1\right\} = \begin{bmatrix} \lambda\mathbf{\Lambda}_c(n-1) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (56)$$

In order to approach the cost function from (27), a modified form of the previous matrix will be used in our development, i.e.,

$$\mathbf{\Lambda}(n) = (1-\lambda)\text{diag}\left\{\lambda^{n-1}, \lambda^{n-2}, \ldots, 1\right\} = \begin{bmatrix} \lambda\mathbf{\Lambda}(n-1) & \mathbf{0} \\ \mathbf{0}^T & 1-\lambda \end{bmatrix}. \quad (57)$$

Using (57) and following the QR-decomposition we have:

$$\begin{bmatrix} 1 & \mathbf{0}^T & 0 \\ \mathbf{0} & \mathbf{Q}_{m-2}(n-2) & \mathbf{0} \\ 0 & \mathbf{0}^T & 1 \end{bmatrix} \mathbf{\Lambda}^{1/2}(n)\mathbf{A}_{m+1}(n)$$

$$= \begin{bmatrix} (1-\lambda)^{1/2}\lambda^{(n-1)/2}x^*(1) & \mathbf{0}^T & 0 \\ \lambda^{1/2}\mathbf{p}_{f,m-2}(n-1) & \lambda^{1/2}\mathbf{R}_{m-2}(n-2) & \lambda^{1/2}\mathbf{p}_{b,m-2}(n-2) \\ \lambda^{1/2}\mathbf{v}_{f,m-2}(n-1) & \mathbf{0} & \lambda^{1/2}\mathbf{v}_{b,m-2}(n-2) \\ (1-\lambda)^{1/2}x^*(n) & (1-\lambda)^{1/2}\mathbf{x}_{m-1}^H(n-1) & (1-\lambda)^{1/2}x^*(n-m) \end{bmatrix} \quad (58)$$

Let $\mathbf{B}(n)$ denote the matrix on the right-hand term of (58). We use a unitary matrix $\mathbf{P}(n-2)$ to annihilate the vector $\mathbf{v}_{b,m-2}(n-2)$, except for its first element, denoted by $\sqrt{J_{m-1}^b(n-2)}$. A new element is generated, namely, $\pi_{f,m-1}(n-1)$, in the first column. Next, we use the unitary matrix $\mathbf{T}_{m-2}(n-1)$ to update the vectors $\mathbf{p}_{f,m-2}(n-1)$, $\mathbf{p}_{b,m-2}(n-2)$, and the matrix $\mathbf{R}_{m-2}(n-2)$. Angle-normalized forward and delayed backward prediction errors, $\varepsilon_{f,m-1}(n)$ and $\varepsilon_{b,m-1}(n-1)$, are generated in complex conjugate forms and all the elements of the vector $\mathbf{x}_{m-1}^H(n-1)$ are annihilated, i.e.,

$$\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{T}_{m-2}(n-1) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{P}(n-2) & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \mathbf{B}(n)$$

$$= \begin{bmatrix} (1-\lambda)^{1/2}\lambda^{(n-1)/2}x^*(1) & \mathbf{0}^T & 0 \\ \mathbf{p}_{f,m-2}(n) & \mathbf{R}_{m-2}(n-1) & \mathbf{p}_{b,m-2}(n-1) \\ \lambda^{1/2}\pi_{f,m-1}(n-1) & \mathbf{0}^T & \lambda^{1/2}\sqrt{J_{m-1}^b(n-2)} \\ \lambda^{1/2}\mathbf{v}_{f,m-1}(n-1) & \mathbf{0} & \mathbf{0} \\ (1-\lambda)^{1/2}\varepsilon_{f,m-1}^*(n) & \mathbf{0}^T & (1-\lambda)^{1/2}\varepsilon_{b,m-1}^*(n-1) \end{bmatrix} \quad (59)$$

Let $\mathbf{C}(n)$ denote the matrix on the right-hand term of (59). We can write:

$$\begin{bmatrix} 1 & \mathbf{0}^T & 0 & \mathbf{0}^T & 0 \\ \mathbf{0} & \mathbf{I}_{m-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & \mathbf{0}^T & c_{b,m-1}(n-1) & \mathbf{0}^T & s_{b,m-1}^*(n-1) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{n-m-1} & \mathbf{0} \\ 0 & \mathbf{0}^T & -s_{b,m-1}(n-1) & \mathbf{0}^T & c_{b,m-1}(n-1) \end{bmatrix} \mathbf{C}(n)$$

$$= \begin{bmatrix} (1-\lambda)^{1/2}\lambda^{(n-1)/2}x^*(1) & \mathbf{0}^T & 0 \\ \mathbf{p}_{f,m-2}(n) & \mathbf{R}_{m-2}(n-1) & \mathbf{p}_{b,m-2}(n-1) \\ \pi_{f,m-1}(n) & \mathbf{0}^T & \sqrt{J_{m-1}^b(n-1)} \\ \lambda^{1/2}\mathbf{v}_{f,m-1}(n-1) & \mathbf{0} & \mathbf{0} \\ (1-\lambda)^{1/2}\varepsilon_{f,m}^*(n) & \mathbf{0}^T & 0 \end{bmatrix} \quad (60)$$

where $\mathbf{I}$ denotes identity matrices. Furthermore, the following updates result:

$$c_{b,m-1}(n-1) = \frac{\lambda^{1/2}\sqrt{J_{m-1}^b(n-2)}}{\sqrt{J_{m-1}^b(n-1)}}, \quad (61)$$

$$s_{b,m-1}(n-1)=\frac{(1-\lambda)^{1/2}\varepsilon^{*}_{b,m-1}(n-1)}{\sqrt{J^{b}_{m-1}(n-1)}}, \quad (62)$$

$$J^{b}_{m-1}(n-1)=\lambda J^{b}_{m-1}(n-2)+(1-\lambda)\left|\varepsilon_{b,m-1}(n-1)\right|^{2}, \quad (63)$$

$$\varepsilon_{f,m}(n)=c_{b,m-1}(n-1)\varepsilon_{f,m-1}(n) - s^{*}_{b,m-1}(n-1)\pi^{*}_{f,m-1}(n-1)\left(\frac{\lambda}{1-\lambda}\right)^{1/2}, \quad (64)$$

$$\pi^{*}_{f,m-1}(n)=c_{b,m-1}(n-1)\lambda^{1/2}\pi^{*}_{f,m-1}(n-1) + s_{b,m-1}(n-1)\varepsilon_{f,m-1}(n)(1-\lambda)^{1/2} \quad (65)$$

In the same manner, another two transformations over the matrix $\mathbf{B}(n)$ are performed, i.e.,

$$\begin{bmatrix}1 & \mathbf{0}^{T}\\ \mathbf{0} & \mathbf{T}_{m-2}(n-1)\end{bmatrix}\cdot\begin{bmatrix}\mathbf{K}(n-1) & \mathbf{0}\\ \mathbf{0}^{T} & 1\end{bmatrix}\cdot\mathbf{B}(n)$$

$$=\begin{bmatrix}\lambda^{1/2}\sqrt{J^{f}_{m-1}(n-1)} & \mathbf{0}^{T} & \lambda^{1/2}\pi_{b,m-1}(n-1)\\ \mathbf{p}_{f,m-2}(n) & \mathbf{R}_{m-2}(n-1) & \mathbf{p}_{b,m-2}(n-1)\\ \mathbf{0} & \mathbf{O} & \lambda^{1/2}\mathbf{v}_{b,m-1}(n-1)\\ (1-\lambda)^{1/2}\varepsilon^{*}_{f,m-1}(n) & \mathbf{0}^{T} & (1-\lambda)^{1/2}\varepsilon^{*}_{b,m-1}(n-1)\end{bmatrix} \quad (66)$$

Let $\mathbf{D}(n)$ denote the matrix on the right-hand term of (66). We can write:

$$\begin{bmatrix}c_{f,m-1}(n) & \mathbf{0}^{T} & s^{*}_{f,m-1}(n)\\ \mathbf{0} & \mathbf{I}_{n-2} & \mathbf{0}\\ s_{f,m-1}(n) & \mathbf{0}^{T} & c_{f,m-1}(n)\end{bmatrix}\mathbf{D}(n)$$

$$=\begin{bmatrix}\sqrt{J^{f}_{m-1}(n)} & \mathbf{0}^{T} & \pi_{b,m-1}(n)\\ \mathbf{p}_{f,m-2}(n) & \mathbf{R}_{m-2}(n-1) & \mathbf{p}_{b,m-2}(n-1)\\ \mathbf{0} & \mathbf{O} & \lambda^{1/2}\mathbf{v}_{b,m-1}(n-1)\\ 0 & \mathbf{0}^{T} & (1-\lambda)^{1/2}\varepsilon^{*}_{b,m}(n)\end{bmatrix} \quad (67)$$

Similarly, a set of recursive relations for the forward prediction part of the algorithm are obtained, i.e.,

$$c_{f,m-1}(n)=\frac{\lambda^{1/2}\sqrt{J^{f}_{m-1}(n-1)}}{\sqrt{J^{f}_{m-1}(n)}}, \quad (68)$$

$$s_{f,m-1}(n)=\frac{(1-\lambda)^{1/2}\varepsilon^{*}_{f,m-1}(n)}{\sqrt{J^{f}_{m-1}(n)}}, \quad (69)$$

$$J^{f}_{m-1}(n)=\lambda J^{f}_{m-1}(n-1)+(1-\lambda)\left|\varepsilon_{f,m-1}(n)\right|^{2}, \quad (70)$$

$$\varepsilon_{b,m}(n)=c_{f,m-1}(n)\varepsilon_{b,m-1}(n-1) - s^{*}_{f,m-1}(n)\pi^{*}_{b,m-1}(n-1)\left(\frac{\lambda}{1-\lambda}\right)^{1/2}, \quad (71)$$

$$\pi^{*}_{b,m-1}(n)=c_{f,m-1}(n)\lambda^{1/2}\pi^{*}_{b,m-1}(n-1) + s_{f,m-1}(n)\varepsilon_{b,m-1}(n-1)(1-\lambda)^{1/2}. \quad (72)$$

Finally, for the joint-process estimation part of the algorithm we have

$$\varepsilon_{m+1}(n)=c_{b,m}(n)\varepsilon_{m}(n)-s^{*}_{b,m}(n)p^{*}_{m}(n-1)\left(\frac{\lambda}{1-\lambda}\right)^{1/2}, \quad (73)$$

$$p^{*}_{m}(n)=c_{b,m}(n)\lambda^{1/2}p^{*}_{m}(n-1)+s_{b,m}(n)\varepsilon_{m}(n)(1-\lambda)^{1/2}. \quad (74)$$

Summarizing, the proposed algorithm uses (61)–(65) for the backward prediction part, together with (68)–(72) for the forward prediction part, and (73), (74) for the joint-process estimation. Let us call this modified algorithm by QRD-LSL-m1. According to the discussion from the end of Section 3, this type of algorithm achieve a slower initial convergence rate than the classical one (for the same initialization parameters) but we may use a variable exponential weighting factor according to (49) in order to speed up the initial convergence of the algorithm. Therefore, it results a second modified algorithm, which we called QRD-LSL-m2.

The computational complexity of the proposed algorithms is similar with the complexity of the classical QRD-LSL (i.e., around $20L$), while the computational amount of the RLS algorithm is around $3L^{2}$.

5. SIMULATION RESULTS

For the first set of experiments we consider an adaptive "system identification" configuration [1]. In this class of applications an adaptive filter is used to provide a linear model that represents the best fit (in some sense) to an unknown system. The adaptive filter and the unknown system are driven by the same input; the unknown system output supplies the desired response for the adaptive filter. These two signals are used to compute the estimation error, in order to adjust the filter coefficients. Our input signal is

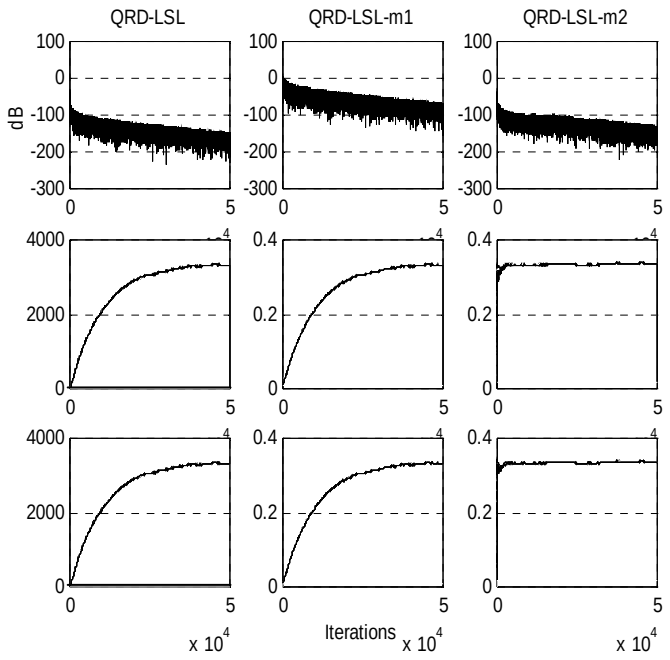

Fig. 1. Square errors [dB] and the cost functions of the classical QRD-LSL algorithm (*column 1*) and the modified versions QRD-LSL-m1 (*column 2*) and QRD-LSL-m2 (*column 2*), in a system identification setup. *Row 1* – Square errors [dB]; *Row 2* – J^b cost functions; *Row 3* – J^f cost functions.

a random sequence with an uniform distribution in the interval (–1;1). The order of the adaptive filter is $M = 64$. In Fig. 1 are presented the convergence curves and the evolution of the cost functions for the classical QRD-LSL algorithm and its modified versions, QRD-LSL-m1 and QRD-LSL-m2, using $\lambda = 0.9999$. In the case of both modified algorithms the values of the cost functions can not exceed 1 [according to (63) and (70)]. Hence, due to the reduced dynamic range of these parameters, the "effort" for scaling procedures is significantly reduced, On the other hand, the cost functions of the classical algorithm will be upper bounded (theoretical) by $1/(1 - \lambda)$, which leads to large values when λ is close to 1. Also, it can be noticed that the QRD-LSL-m2 achieve the same initial convergence rate as the classical QRD-LSL algorithm.

The previous simulation was performed using the full precision of Matlab programming environment. Next, the algorithms are implemented in fixed-point precision, using a fixed-point DSP with a word length of 16 bits (15 bits for the magnitude and one sign bit). The usage of a higher precision (e.g., 24 or 32 bits) could lead to better performances but also increases the implementation costs. As a practical aspect of this work we choose to illustrate the algorithms performance in a noise reduction scenario (Fig. 2) [1]. In this type of application, the adaptive filter is use to synthesize at its output a replica of the perturbation that corrupts the voice signal.

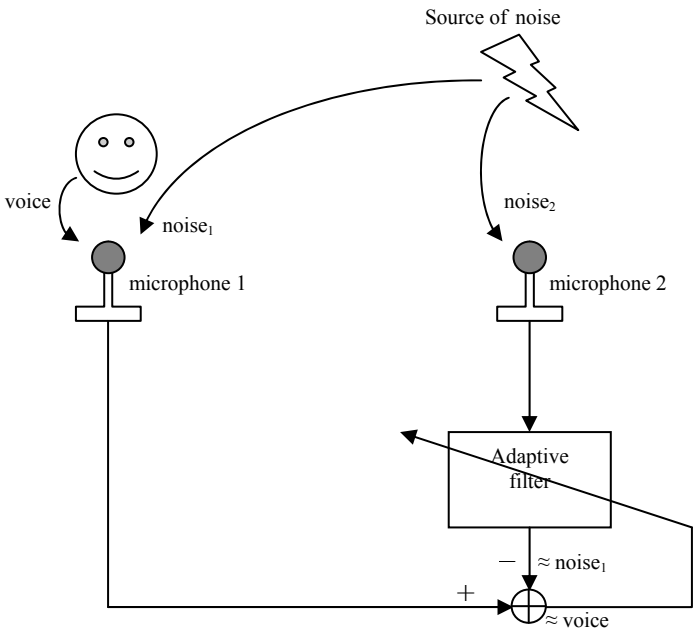

Fig. 2. Adaptive noise reduction scheme.

In the original QRD-LSL algorithm the asymptotic value for the cost functions are $1/(1 - \lambda)$. Since λ is very close to 1, the original algorithm will certainly produce overflow and thus needs to be scaled, i.e., the cost functions must be right-shifted by a number of bits such chosen as to avoid the overflow in the convergence state. A simple calculus shows that the optimum number of bits to shift-right the cost functions is $B_s = \lceil -\log_2(1 - \lambda) \rceil$, where $\lceil \bullet \rceil$ denotes superior integer round. Nevertheless, this further leads to the reduction of the effective number of bits, especially when λ is very close to 1 and eventually to a low signal to quantization noise ratio, altering the algorithm performances. For this reasons we choose to compare the QRD-LSL-m2 algorithm (since it has a faster initial converge rate as compared to QRD-LSL-m1) with the normalized least-mean-square (NLMS) algorithm [1], which is one of the most common solution for noise reduction [16], [17]. Since the computational amount of the NLMS is around $3L$, this algorithm is "cheaper" (in terms of complexity) as compared with the proposed QRD-LSL algorithm. Nevertheless, the performances of the NLMS algorithm are strongly reduced when high order adaptive filters and non-stationary inputs (e.g., speech) are used. In these cases, the RLS-based algorithms rule.

The results of the noise reduction experiment are presented in Figs. 3 and 4, using two type of noise, i.e., a white Gaussian noise with SNR = 10dB (Fig. 3) and a highway noise (Fig. 4). The last one is more severe because it is a non-stationary signal. In both cases the QRD-LSL-m2 algorithm outperforms the NLMS algorithm.

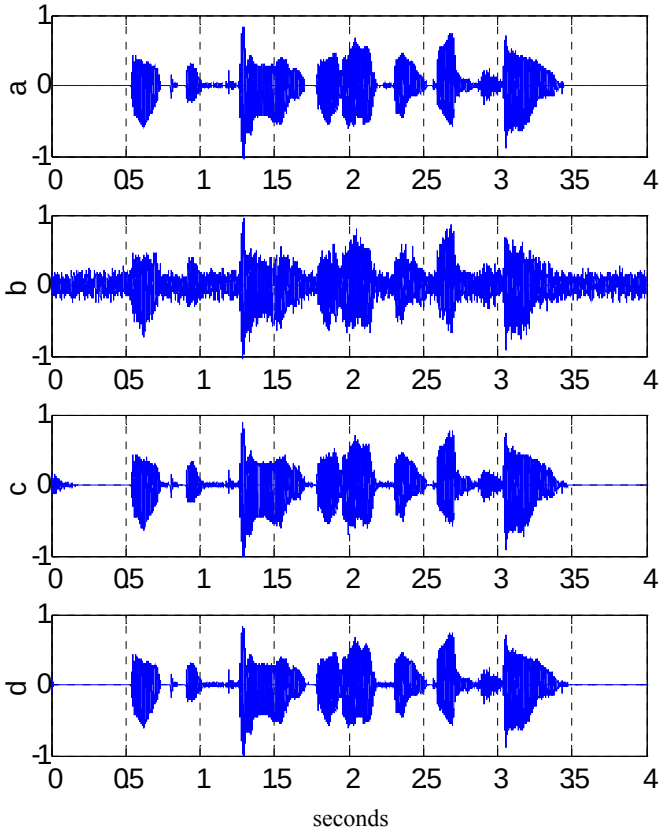

Fig. 3. (a) original signal; (b) corrupted signal (with white Gaussian noise); (c) recovered signal using the NLMS algorithm; (d) recovered signal using the QRD-LSL-m2 algorithm.

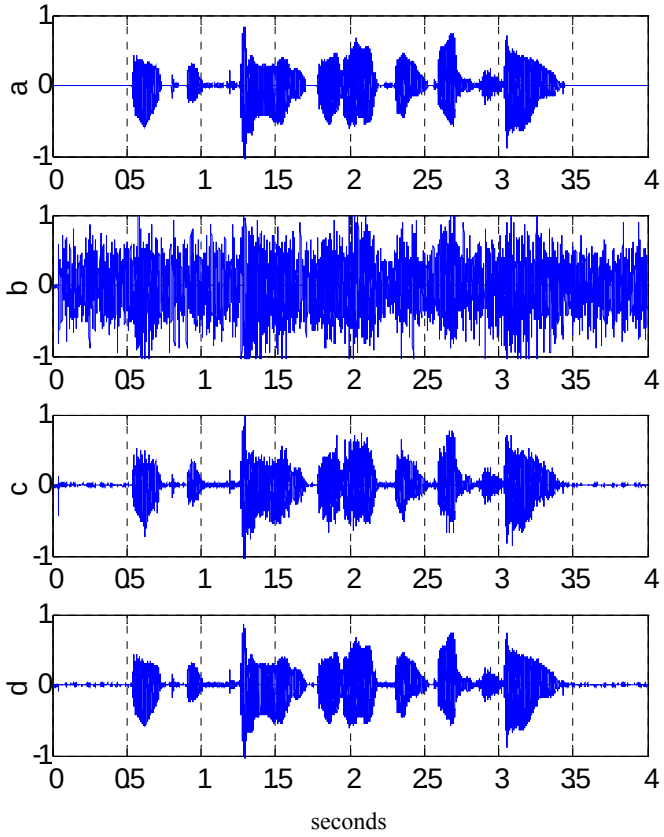

Fig. 4. (a) original signal; (b) corrupted signal (with highway noise); (c) recovered signal using the NLMS algorithm; (d) recovered signal using the QRD-LSL-m2 algorithm.

In the first case (Fig. 3) the subjective tests indicate a mean opinion score (MOS) of 3.9 for the NLMS algorithm and 4.5 for the QRD-LSL-m2 algorithm. In the second case the difference becomes more apparent, i.e., 3.2 for the NLMS algorithm and 4.1 for the QRD-LSL-m2. Note that the MOS scale is from 1 to 5, where 1 means very poor and 5 means excellent quality. This was evaluated in a subjective manner, as the average of the scores given by 20 listeners.

6. DISCUSSION

A first goal of this paper was to present and analyze a modified version of the RLS adaptive algorithm with improved features for fixed-point implementation. The basic idea was to use an asymptotically unbiased estimator for the cost function. In this manner we try to prevent the stalling phenomenon which may appear when a high memory RLS algorithm is implemented using fixed-point arithmetic. A brief convergence analysis of the RLS algorithms was performed, together with a discussion concerning the proper scale factor, which has to be chosen in order to avoid the overflow and stalling effects.

In the case of the modified RLS algorithm only the initial convergence rate is affected when it operates with the same value of the regularization parameter as the classical RLS algorithm. Choosing the value of this parameter according to (42), the modified algorithm achieves the same initial convergence rate as the classical one. Moreover, the variable exponential weighting factor from (49) speeds up the initial convergence rate of this algorithm, leading to a reasonable compromise between the convergence rate and dynamic range of the algorithm's parameters.

The procedure presented in the case of the RLS algorithm was developed and applied in the case of the QRD-LSL algorithm, which is a fast member of the RLS family. Two modified versions of the QRD-LSL algorithm were proposed. Based on the asymptotically unbiased estimator for the cost functions, we improve the behavior of these algorithms when dealing with fixed-point arithmetic. As expected, only the initial convergence rate of the QRD-LSL-m1 algorithm is affected when it operates with the same parameters as the classical QRD-LSL algorithm. Also, the variable exponential weighting factor used for QRD-LSL-m2 algorithm speeds up its initial

convergence rate. The simulations performed in both Matlab and fixed-point DSP support the theoretical findings.

7. CONCLUSIONS

A class of RLS algorithms suitable for fixed-point implementation was presented in this paper. The proposed approach was applied in the case of the QRD-LSL algorithm. The performance of the resulted algorithm was evaluated in a noise reduction scenario, obtaining promising results.

8. REFERENCES

[1] S. Haykin, *Adaptive Filter Theory – Fourth Edition*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2002.

[2] T. Adali and S. H. Ardalan, "Convergence and error analysis of the fixed point RLS algorithm with correlated inputs," *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 1990*, vol. 3, pp. 1479-1482.

[3] I.K. Proudler, J.G. McWhirter, and T.J. Shepherd, "QRD-based lattice filter algorithms," *Proc. SPIE*, vol. 1152, pp. 56-67, 1989.

[4] J.-T. Yuan, **"**A modified QRD for smoothing and a QRD-LSL smoothing algorithm," *IEEE Trans. Signal Processing*, vol. 47, no. 5, pp. 1414-1420, May 1999.

[5] J.-T. Yuan and J.-N. Lee, "Narrow-band interference rejection in DS/CDMA systems using adaptive (QRD-LSL)-based nonlinear ACM interpolators," *IEEE Trans. Vehicular Technology*, vol. 52, no. 2, pp. 374-379, Mar. 2003.

[6] C. Paleologu, S. Ciochină, and A.A. Enescu, "A network echo canceler based on a SRF QRD-LSL adaptive algorithm implemented on Motorola StarCore SC140 DSP," *Lecture Notes in Computer Science, Springer-Verlag*, vol. 3124, pp. 560-567, June 2004.

[7] G. Rombouts and M. Moonen, "Fast QRD-lattice-based unconstrained optimal filtering for acoustic noise reduction," *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 6, pp. 1130-1143, Nov. 2005.

[8] J.-T. Yuan, C.-A. Chiang, and C.-H. Wu, "A square-root-free QRD-LSL interpolation algorithm," *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, ICASSP 2008*, pp. 3813-3816, Apr. 2008.

[9] C. Paleologu, A. A. Enescu, S. Ciochină, and F. Albu, "QRD-LSL Adaptive Algorithms Suitable for Fixed-Point Implementation," *Proc. IEEE Advanced International Conference on Telecommunications (AICT)*, pp. 163-167, Venice, Italy, May 2009.

[10] C. Paleologu, F. Albu, A. A. Enescu, and S. Ciochină, "Modified SRF-QRD-LSL Adaptive Algorithm with Improved Numerical Robustness," *IARIA International Journal on Advances in Systems and Measurements*, vol. 2, no. 1, pp. 56-65, 2009.

[11] J. Benesty and Y. Huang, Eds, *Adaptive Signal Processing--Applications to Real-World Problems*. Springer-Verlag, Berlin, Germany, 2003.

[12] G. V. Moustakides, "Study of the Transient Phase of the Forgetting Factor RLS," *IEEE Transactions on Signal Processing*, vol. 45, no. 10, pp. 2468-2476, Oct. 1997.

[13] M. H. Verhaegen, "Round-off Error Propagation in Four Generally-Applicable, Recursive Least-Squares Estimation Schemes," *Automatica*, vol. 25, no. 3, pp. 437-444, 1989.

[14] S. Ciochină, C. Paleologu, and A.A. Enescu, "On the Behaviour of RLS Adaptive Algorithm in Fixed-Point Implementation," *Proc. of IEEE Int. Symp. on Signals, Circuits and Systems*, *SCS 2003*, vol. 1, pp. 57-60, July 2003.

[15] C. Paleologu, A. A. Enescu, and S. Ciochină, "Recursive Least-Squares Lattice Adaptive Algorithm Suitable for Fixed-Point Implementation," *Proc of IEEE International Conference on Electronics, Circuits and Systems*, *ICECS 2006*, pp.1105-1108, Dec. 2006

[16] E. Haensler and G. Schmidt, Eds., *Topics in Acoustic Echo and Noise Control.* Springer-Verlag, Berlin, Germany, 2006.

[17] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Noise Reduction in Speech Processing.* Springer-Verlag, Berlin, Germany, 2009.