Lattice RLS for Nonstationary Signal Processing

Victor I. DJIGAN

VLSI Development Laboratory, ELVEES R&D Center of Microelectronics,
POB 19, Centralny Prospect, Zelenograd, Moscow, Russia 124460

djigan@elvees.ru

Abstract. *The paper presents the modification of lattice RLS adaptive filtering algorithms for the case of nonstationary signal processing. The modification includes the using of sliding window and dynamic regularization in the adaptive filter correlation matrix estimation. The algorithms can be implemented by means of sequential or parallel computations. Based on sequential computations, 30 regularized prewindowed, sliding widow and regularized sliding window lattice RLS algorithms were developed. A family of the same algorithms with parallel computations includes 21 units. Lattice algorithms are developed for the application in single-channel adaptive filters with complex-valued weights. Some computation procedures of the algorithms are listed. The results of the algorithm simulation are also presented.*

Keywords

Adaptive filtering, RLS, fast RLS, lattice RLS.

1. Introduction

The Recursive Least Squares (RLS) algorithms [1] are among the most efficient algorithms in adaptive signal processing. They provide the smallest transient response and steady-state residual errors in comparing with the simple gradient algorithms [2]. Despite the large arithmetic complexity (number of the required arithmetic operations per sample) the RLS algorithms are also used in applications as modern Digital Signal Processors (DSP) support the efficient implementations of the computationally complex algorithms [3].

In practice, the complexity of RLS algorithms has to be further increased if nonstationary signals are processed. It takes place due to the need to use the regularization of the adaptive filter correlation matrix [4] or to estimate the matrix over a sliding window (SW) with a limited number of samples [5]. The modifications are the reasons of two or four time increase of the complexity in comparing with that of the infinitively growing window (prewindowed, PW) RLS algorithms. It is the price that has to be paid for the ability of nonstationary signal processing. If there are a few DSPs for the implementation of an adaptive filter, the computational load per processor can be decreased two or four times, if RLS algorithms are developed in the form, fitted to parallel computations [6].

There is another class of adaptive filtering RLS algorithms called lattice [7-11]. These order-recursive algorithms do not compute weights explicitly. The algorithms are used in applications where the required output signal of an adaptive filter is an error signal only. With the proper initialization the lattice RLS algorithms are mathematically identical to the time-recursive least squares algorithms [1], [6].

The lattice algorithms are so-called fast, i.e. $O(N)$ complexity, RLS adaptive filtering algorithms. Here, N is the number of adaptive filter weights. The average algorithms complexity exceeds that of the fast RLS algorithms [1]. However lattice RLS adaptive filtering algorithms are known as stable ones that is a reason why they are also used in applications.

Unfortunately, there are no papers which describe the application of the regularization in lattice algorithms, SW lattice algorithms are presented by the papers [12, 13] only, and there are no papers on the join using of regularization and SW in the algorithms.

The purpose of the given paper is to present the basic ideas of the using of the above mentioned modifications in lattice RLS algorithms and to present the implementation of some of the algorithms by means of sequential and parallel computations.

2. General Suggestions

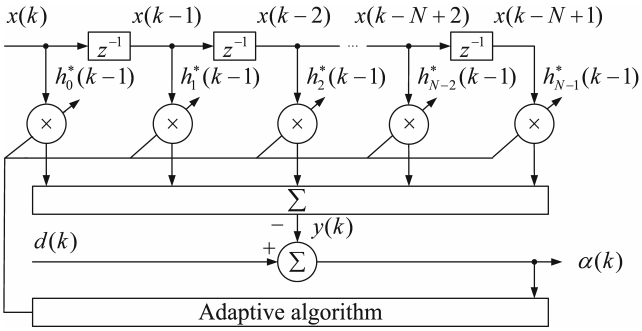

Fig. 1. Single-channel adaptive filter.

Fig. 1 shows the structure of a single-channel adaptive filter with complex-valued weights $h_n(k)$, where k is a current sample number, and $x(k)$, $d(k)$, $y(k)$, $\alpha(k)$ are input, reference, output and a priori error signals of the filter. The output signal is produced as $y(k)=\mathbf{h}^H_N(k\text{-}1)\mathbf{x}(k)$, where $\mathbf{h}_N(k)=[h(k), h(k\text{-}1),\ldots, h(k\text{-}N\text{+}2), h(k\text{-}N\text{+}1)]^T$ is a vector of adaptive filter weighs and $\mathbf{x}_N(k)=[x(k), x(k\text{-}1),\ldots, x(k\text{-}N\text{+}2), x(k\text{-}N\text{+}1)]^T$ is a vector of adaptive filter input signals. A priori error is determined as $\alpha(k)=d(k)\text{-}y(k)$. The following suggestions are used over the paper. Matrices and vectors are denoted by capital and regular characters, superscripts H and T denote Hermitian transpose and transposition of a vector or a matrix, subscripts N and F denote the dimension of vectors and square matrices, superscript $()^*$ denotes the conjugating of a variable.

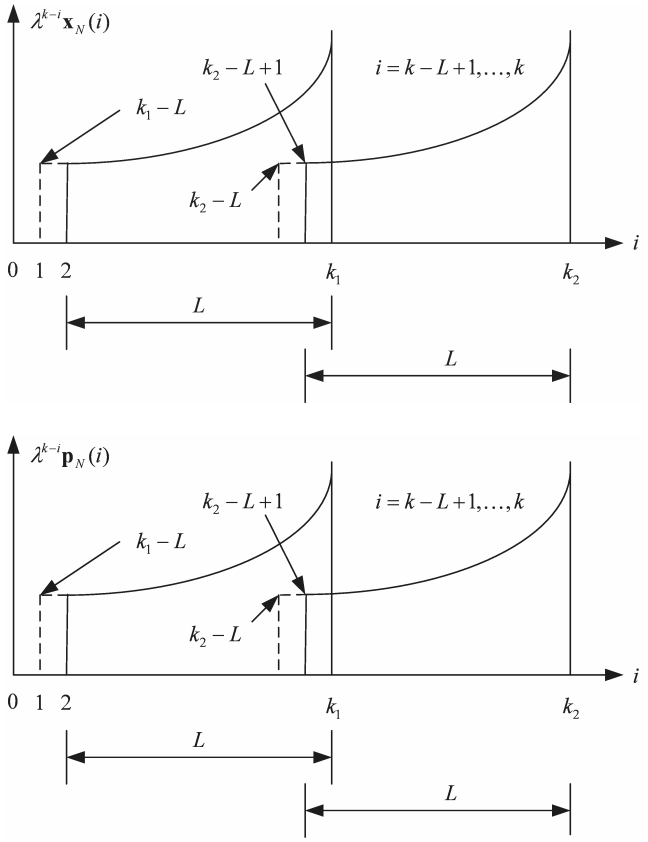

Fig. 2. Sliding windows.

Papers [4-6] describe basic rules of RLS algorithms modification in case of the application of dynamic regularization and SW. Fig. 2 demonstrates the application of exponentially weighted SW to the vectors of input signal $\mathbf{x}_N(k)$ and to the vectors of artificial regularization signal $\mathbf{p}_N(k)$. Here λ is a forgetting factor, L is the sliding window length, defined by the number of samples. Due to downdating in estimation of adaptive filter correlation matrix (shown by dashed lines in Fig. 2), the SW RLS algorithms complexity is increased two times in comparing with PW RLS algorithms. If both SW and regularizations are used, the complexity is increased four times. These modifications produce the sets of similar computations that depend on one of the processed signals. The modifications are also applied to lattice algorithms. The computations can be executed sequentially or in parallel, see Fig. 3.

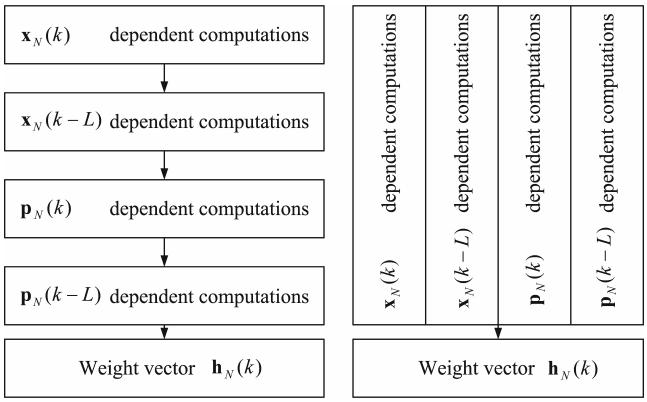

Fig. 3. Sequential and parallel computations in RLS algorithms.

3. Sequential and Parallel Lattice RLS Algorithms

It is known ten basic forms of lattice adaptive filtering algorithms [11]. They are a priori and a posteriori errors algorithm, the mentioned algorithm with feedbacks, only a priori or only posteriori error algorithms, the mentioned algorithms with feedbacks, a posteriori error normalized algorithm, a posteriori error QR Decomposition (QRD) algorithm, QRD algorithm in state-space, and a priori error square root free QRD algorithm.

Lattice RLS algorithms development is based on the using of the formulas for the inversion of (N+1)-th order correlation matrices [7]. These formulas are valid for any n=1,…,N order. In PW case, they allow to calculate the vectors of linear prediction filter weights as

$$\mathbf{a}_{n+1}(k)=\begin{bmatrix}\mathbf{a}_n(k)\\0\end{bmatrix}+\begin{bmatrix}0\\\mathbf{b}_n(k-1)\end{bmatrix}\Gamma_{x_U}^{f(n-1)}(k),$$

$$\mathbf{b}_{n+1}(k)=\begin{bmatrix}0\\\mathbf{b}_n(k-1)\end{bmatrix}+\begin{bmatrix}\mathbf{a}_n(k)\\0\end{bmatrix}\Gamma_{x_U}^{b(n-1)}(k).$$

A priori prediction errors of the filters are calculated as

$$\alpha_{x_U}^{f(n+1)}(k)=\alpha_{x_U}^{f(n)}(k)+\Gamma_{x_U}^{f(n)*}(k-1)\alpha_{x_U}^{b(n)}(k-1),$$

$$\alpha_{x_U}^{b(n+1)}(k)=\alpha_{x_U}^{b(n)}(k-1)+\Gamma_{x_U}^{b(n)*}(k-1)\alpha_{x_U}^{f(n)}(k),$$

and a posteriori prediction errors are calculated as

$$e_{x_U}^{f(n+1)}(k)=e_{x_U}^{f(n)}(k)+\Gamma_{x_U}^{f(n)*}(k)e_{x_U}^{b(n)}(k-1),$$

$$e_{x_U}^{b(n+1)}(k)=e_{x_U}^{b(n)}(k-1)+\Gamma_{x_U}^{b(n)*}(k)e_{x_U}^{f(n)}(k).$$

Similarly, a priori and a posteriori modeling errors are calculated as

$$\alpha_{x_U}^{(n+1)}(k)=\alpha_{x_U}^{(n)}(k)+\gamma_{x_U}^{(n)*}(k-1)\alpha_{x_U}^{b(n)}(k),$$

$$e_{x_U}^{(n+1)}(k) = e_{x_U}^{(n)}(k) + \gamma_{x_U}^{(n)*}(k) e_{x_U}^{b(n)}(k).$$

Here,
$\Gamma_{x_U}^{f(n)}(k) = -\Delta_{x_U}^{(n)}(k) \big/ E_{x_U}^{b(n)}(k-1)$,
$\Gamma_{x_U}^{b(n)}(k) = = -\Delta_{x_U}^{(n)*}(k) \big/ E_{x_U}^{f(n)}(k)$,
$\gamma_{x_U}^{(n)}(k) = -D_{x_U}^{(n)}(k) \big/ E_{x_U}^{b(n)}(k)$ are reflection coefficients, $E_{x_U}^{f(n)}(k)$, $E_{x_U}^{b(n)}(k)$ are the energies of the linear prediction errors, and $\Delta_{x_U}^{(n)}(k) = \lambda \Delta_{x_U}^{(n)}(k-1) + e_{x_U}^{b(n)}(k-1)\alpha_{x_U}^{f(n)*}(k)$,
$\kappa_{x_U}^{(n)}(k) = \lambda \kappa_{x_U}^{(n)}(k-1) + \alpha_{x_U}^{b(n)}(k) e_{x_U}^{(n)*}(k)$.

Because of the limited size of the paper, details of the modified lattice algorithms development are omitted. One can refer to [6-11] for basic steps of the PW RLS lattice algorithms development, which have to be used sequentially or in parallel for the modification of adaptive filter correlation matrix as [5] or [6] in SW and regularization cases. If a forgetting factor λ is used, downdated signals are weighed by a $\mu=\lambda^L$ scale. In the paper, subscripts U and D are used to denoted updating and downdating operations, i.e. operations caused by the signals $x_U(k)=x(k)$, $p_U(k)=p(k)$, $d_U(k)=d(k)$, $x_D(k)=x(k\text{-}L)$, $p_D(k)=p(k\text{-}L)$, $d_D(k)=d(k\text{-}L)$. An example of a sequential SW lattice RLS algorithm with a priori and a posteriori errors is shown below. Here, steps 3) to 15) are updating and the steps 16) to 28) are downdating operations, δ is the parameter initial regularization.

Initialization : $E_{x_D}^{f(n)}(0) = \delta^2, E_{x_D}^{b(n)}(0) = \delta^2 \lambda^{-(n-1)}$,
$E_{x_U}^{b(n)}(0) = \delta^2 \lambda^{-(n-1)}, \Delta_{x_D}^{(n)}(0) = 0, \kappa_{x_D}^{(n)}(0) = 0,$
0) $\alpha_{x_U}^{b(n)}(0) = 0, \alpha_{x_D}^{b(n)}(0) = 0, e_{x_U}^{b(n)}(0) = 0,$
$e_{x_D}^{b(n)}(0) = 0, \Gamma_{x_D}^{f(n)}(0) = 0, \Gamma_{x_D}^{b(n)}(0) = 0, \gamma_{x_D}^{(n)}(0) = 0,$
$n = 1,2,\ldots,N$

For $k = 1,2,\ldots,K$

1) $\alpha_{x_U}^{f(1)}(k) = x(k), \alpha_{x_U}^{b(1)}(k) = x(k), e_{x_U}^{f(1)}(k) = x(k),$
$e_{x_U}^{b(1)}(k) = x(k), \alpha_{x_U}^{(1)}(k) = d(k), e_{x_U}^{(1)}(k) = d(k)$

2) $\alpha_{x_D}^{f(1)}(k) = x(k-L), \alpha_{x_D}^{b(1)}(k) = x(k-L),$
$e_{x_D}^{f(1)}(k) = x(k-L), e_{x_D}^{b(1)}(k) = x(k-L),$
$\alpha_{x_D}^{(1)}(k) = d(k-L), e_{x_D}^{(1)}(k) = d(k-L)$

For $n = 1,2,\ldots,N$

3) $E_{x_U}^{f(n)}(k) = \lambda E_{x_D}^{f(n)}(k-1) + e_{x_U}^{f(n)}(k)\alpha_{x_U}^{f(n)*}(k)$
4) $E_{x_U}^{b(n)}(k) = \lambda E_{x_D}^{b(n)}(k-1) + e_{x_U}^{b(n)}(k)\alpha_{x_U}^{b(n)*}(k)$
5) $\Delta_{x_U}^{(n)}(k) = \lambda \Delta_{x_D}^{(n)}(k-1) + e_{x_U}^{b(n)}(k-1)\alpha_{x_U}^{f(n)*}(k)$
6) $\kappa_{x_U}^{(n)}(k) = \lambda \kappa_{x_D}^{(n)}(k-1) + \alpha_{x_U}^{b(n)}(k) e_{x_U}^{(n)*}(k)$
7) $\alpha_{x_U}^{f(n+1)}(k) = \alpha_{x_U}^{f(n)}(k) + \Gamma_{x_D}^{f(n)*}(k-1)\alpha_{x_U}^{b(n)}(k-1)$
8) $\alpha_{x_U}^{b(n+1)}(k) = \alpha_{x_U}^{b(n)}(k-1) + \Gamma_{x_D}^{b(n)*}(k-1)\alpha_{x_U}^{f(n)}(k)$
9) $\alpha_{x_U}^{(n+1)}(k) = \alpha_{x_U}^{(n)}(k) + \gamma_{x_D}^{(n)*}(k-1)\alpha_{x_U}^{b(n)}(k)$
10) $\Gamma_{x_U}^{f(n)}(k) = -\Delta_{x_U}^{(n)}(k) \big/ E_{x_U}^{b(n)}(k-1)$
11) $\Gamma_{x_U}^{b(n)}(k) = -\Delta_{x_U}^{(n)*}(k) \big/ E_{x_U}^{f(n)}(k)$
12) $\gamma_{x_U}^{(n)}(k) = -\kappa_{x_U}^{(n)}(k) \big/ E_{x_U}^{b(n)}(k)$
13) $e_{x_U}^{f(n+1)}(k) = e_{x_U}^{f(n)}(k) + \Gamma_{x_U}^{f(n)*}(k) e_{x_U}^{b(n)}(k-1)$
14) $e_{x_U}^{b(n+1)}(k) = e_{x_U}^{b(n)}(k-1) + \Gamma_{x_U}^{b(n)*}(k) e_{x_U}^{f(n)}(k)$
15) $e_{x_U}^{(n+1)}(k) = e_{x_U}^{(n)}(k) + \gamma_{x_U}^{(n)*}(k) e_{x_U}^{b(n)}(k)$
16) $E_{x_D}^{f(n)}(k) = E_{x_U}^{f(n)}(k) - \mu e_{x_D}^{f(n)}(k)\alpha_{x_D}^{f(n)*}(k)$
17) $E_{x_D}^{b(n)}(k) = E_{x_U}^{b(n)}(k) - \mu e_{x_D}^{b(n)}(k)\alpha_{x_D}^{b(n)*}(k)$
18) $\Delta_{x_D}^{(n)}(k) = \Delta_{x_U}^{(n)}(k) - \mu e_{x_D}^{b(n)}(k-1)\alpha_{x_D}^{f(n)*}(k)$
19) $\kappa_{x_D}^{(n)}(k) = \kappa_{x_U}^{(n)}(k) - \mu \alpha_{x_D}^{b(n)}(k) e_{x_D}^{(n)*}(k)$
20) $\alpha_{x_D}^{f(n+1)}(k) = \alpha_{x_D}^{f(n)}(k) + \Gamma_{x_U}^{f(n)*}(k)\alpha_{x_D}^{b(n)}(k-1)$
21) $\alpha_{x_D}^{b(n+1)}(k) = \alpha_{x_D}^{b(n)}(k-1) + \Gamma_{x_U}^{b(n)*}(k)\alpha_{x_D}^{f(n)}(k)$
22) $\alpha_{x_D}^{(n+1)}(k) = \alpha_{x_D}^{(n)}(k) + \gamma_{x_U}^{(n)*}(k)\alpha_{x_D}^{b(n)}(k)$
23) $\Gamma_{x_D}^{f(n)}(k) = -\Delta_{x_D}^{(n)}(k) \big/ E_{x_D}^{b(n)}(k-1)$
24) $\Gamma_{x_D}^{b(n)}(k) = -\Delta_{x_D}^{(n)*}(k) \big/ E_{x_D}^{f(n)}(k)$
25) $\gamma_{x_D}^{(n)}(k) = -\kappa_{x_D}^{(n)}(k) \big/ E_{x_D}^{b(n)}(k)$
26) $e_{x_D}^{f(n+1)}(k) = e_{x_D}^{f(n)}(k) + \Gamma_{x_D}^{f(n)*}(k) e_{x_D}^{b(n)}(k-1)$
27) $e_{x_D}^{b(n+1)}(k) = e_{x_D}^{b(n)}(k-1) + \Gamma_{x_D}^{b(n)*}(k) e_{x_D}^{f(n)}(k)$
28) $e_{x_D}^{(n+1)}(k) = e_{x_D}^{(n)}(k) + \gamma_{x_D}^{(n)*}(k) e_{x_D}^{b(n)}(k)$

End for n
End for k

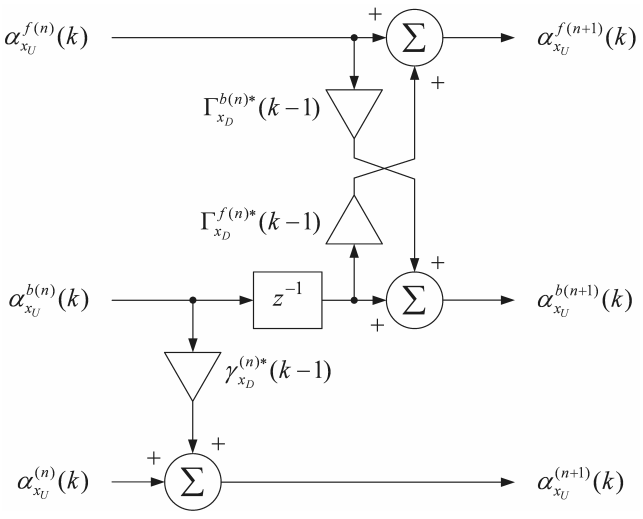

Fig. 4. A stage of a priori errors lattice filter.

A structure of an adaptive filter stage for a priori errors calculation (above equations 7-9) is shown in Fig. 4. A posteriori errors (above equations 13-15) are calculated by means of the same stage with the using of a posteriori refection coefficients $\Gamma^{f(n)}(k)$, $\Gamma^{b(n)}(k)$, $\gamma^{(n)}(k)$, instead of a priori ones. The definitions a priori and a posteriori are caused by the indexes of discrete time (k-1) and (k) in the reflection coefficients that are used in the formulas for errors calculation.

The structure of a priori error calculation in an adaptive filter (above equations 7-9 and 20-22) is shown in Fig. 5. A posteriori errors (above equations 13-15 and 26-28) are calculated, using the same structure but using a posteriori reflection coefficients instead of a priori ones.

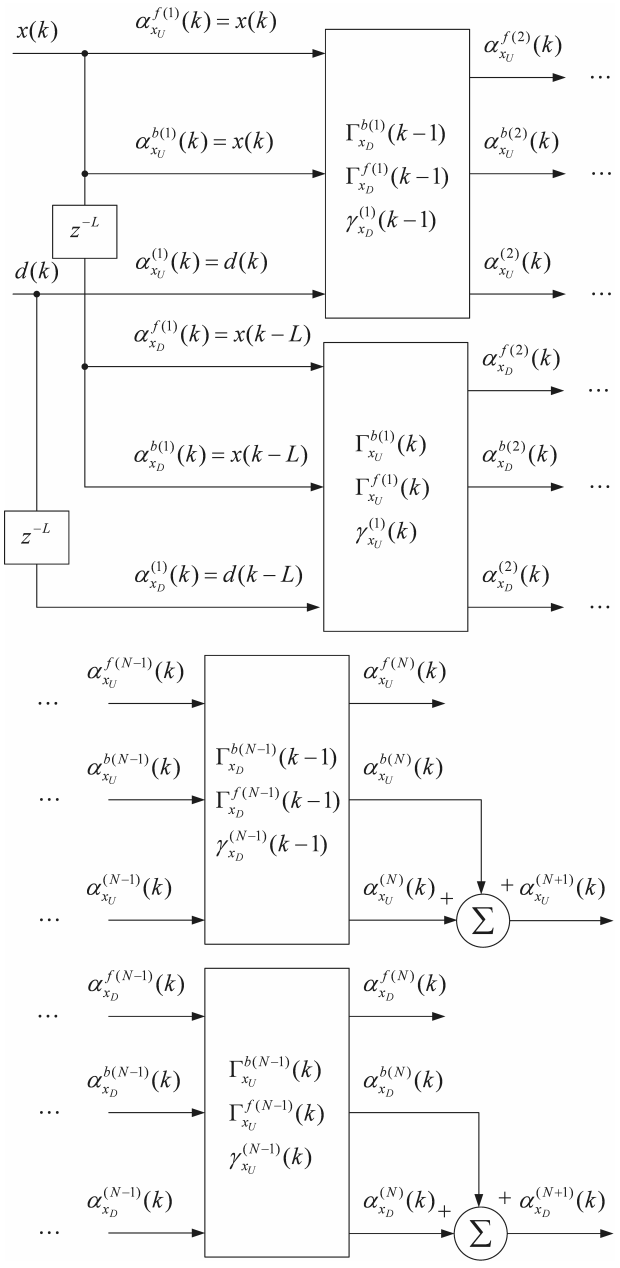

Fig. 5. A priori errors calculation in lattice filter.

An equivalent parallel form of the above algorithm is presented below. The structure for the error calculations is the same as that of Fig. 4. The distinction is the using of the vectors $\boldsymbol{\alpha}_F^{b(n)}(k)$, $\mathbf{e}_F^{b(n)}(k)$, $\boldsymbol{\alpha}_F^{(n)}(k)$ and $\mathbf{e}_F^{(n)}(k)$ instead of scalar values.

Initialization : $E^{f(n)}(0)=\delta^2, E^{b(n)}(0)=\delta^2\lambda^{-(n-1)}$,

0) $\Delta^{(n)}(0)=0, \boldsymbol{\kappa}^{(n)}(0)=0, \boldsymbol{\alpha}_F^{b(n)}(0)=\mathbf{0}_F, \mathbf{e}_F^{b(n)}(0)=\mathbf{0}_F$,

$\Gamma^{f(n)}(0)=0, \Gamma^{b(n)}(0)=0, \gamma^{(n)}(0)=0, n=1,2,\ldots,N$

For $k=1,2,\ldots,K$

$\boldsymbol{\alpha}_F^{f(1)}(k)=\mathbf{x}_F(k), \boldsymbol{\alpha}_F^{b(1)}(k)=\mathbf{x}_F(k),$

1) $\mathbf{e}_F^{f(1)}(k)=\mathbf{x}_F(k)\mathbf{S}_F, \mathbf{e}_F^{b(1)}(k)=\mathbf{x}_F(k)\mathbf{S}_F,$

$\boldsymbol{\alpha}_F^{(1)}(k)=\mathbf{d}_F(k), \mathbf{e}_F^{(1)}(k)=\mathbf{d}_F(k)\mathbf{S}_F$

For $n=1,2,\ldots,N$

2) $E^{f(n)}(k)=\lambda E^{f(n)}(k-1)+\mathbf{e}_F^{f(n)}(k)\boldsymbol{\alpha}_F^{f(n)H}(k)$

3) $E^{b(n)}(k)=\lambda E^{b(n)}(k-1)+\mathbf{e}_F^{b(n)}(k)\boldsymbol{\alpha}_F^{b(n)H}(k)$

4) $\Delta^{(n)}(k)=\lambda\Delta^{(n)}(k-1)+\mathbf{e}_F^{b(n)}(k-1)\boldsymbol{\alpha}_F^{f(n)H}(k)$

5) $\boldsymbol{\kappa}^{(n)}(k)=\lambda\boldsymbol{\kappa}^{(n)}(k-1)+\boldsymbol{\alpha}_F^{b(n)}(k)\mathbf{e}_F^{(n)H}(k)$

6) $\boldsymbol{\alpha}_F^{f(n+1)}(k)=\boldsymbol{\alpha}_F^{f(n)}(k)+\Gamma^{f(n)*}(k-1)\boldsymbol{\alpha}_F^{b(n)}(k-1)$

7) $\boldsymbol{\alpha}_F^{b(n+1)}(k)=\boldsymbol{\alpha}_F^{b(n)}(k-1)+\Gamma^{b(n)*}(k-1)\boldsymbol{\alpha}_F^{f(n)}(k)$

8) $\boldsymbol{\alpha}_F^{(n+1)}(k)=\boldsymbol{\alpha}_F^{(n)}(k)+\gamma^{(n)*}(k-1)\boldsymbol{\alpha}_F^{b(n)}(k)$

9) $\Gamma^{f(n)}(k)=-\Delta^{(n)}(k)/E^{b(n)}(k-1)$

10) $\Gamma^{b(n)}(k)=-\Delta^{(n)*}(k)/E^{f(n)}(k)$

11) $\gamma^{(n)}(k)=-\boldsymbol{\kappa}^{(n)}(k)/E^{b(n)}(k)$

12) $\mathbf{e}_F^{f(n+1)}(k)=\mathbf{e}_F^{f(n)}(k)+\Gamma^{f(n)*}(k)\mathbf{e}_F^{b(n)}(k-1)$

13) $\mathbf{e}_F^{b(n+1)}(k)=\mathbf{e}_F^{b(n)}(k-1)+\Gamma^{b(n)*}(k)\mathbf{e}_F^{f(n)}(k)$

14) $\mathbf{e}_F^{(n+1)}(k)=\mathbf{e}_F^{(n)}(k)+\gamma^{(n)*}(k)\mathbf{e}_F^{b(n)}(k)$

End for n

End for k

The parallel algorithm allows implementing any of three modified algorithms: regularized PW, SW and regularized SW. The algorithms diversity is defined by the number of components in the vectors, i.e. two (F=2) or four (F=4), and by the structure of the vectors of processed data $\mathbf{x}_F(k)$ and $\mathbf{d}_F(k)$, which are the same as those of [6] for a case of a single-channel filter. Output error signal of the adaptive filter with parallel computations is the element of the vector $\boldsymbol{\alpha}_F^{(n)}(k)$, which depends on the $x_U(k)=x(k)$ signal.

The considered algorithms can be also based on the using of only a priori or only a posteriori errors. Such algorithms use the known relationships for the errors, listed below for sequential and parallel computations: $e_{x_U}^{f(n)}(k)=\alpha_{x_U}^{f(n)}(k)\varphi_{x_U}^{(n)}(k-1)$, $e_{x_U}^{b(n)}(k)=\alpha_{x_U}^{b(n)}(k)\times\varphi_{x_U}^{(n)}(k)$, $e_{x_U}^{(n)}(k)=\alpha_{x_U}^{(n)}(k)\varphi_{x_U}^{(n)}(k)$, $e_{x_D}^{f(n)}(k)=\alpha_{x_D}^{f(n)}(k)\varphi_{x_D}^{(n)}(k-1)$, $e_{x_D}^{b(n)}(k)=\alpha_{x_D}^{f(n)}(k)\varphi_{x_D}^{(n)}(k)$, $e_{x_D}^{(n)}(k)=\alpha_{x_D}^{(n)}(k)\varphi_{x_D}^{(n)}(k)$, $\mathbf{e}_F^{f(n)}(k)=\boldsymbol{\alpha}_F^{f(n)}(k)\times\boldsymbol{\Phi}_F^{(n)}(k-1)$, $\mathbf{e}_F^{b(n)}(k)=\boldsymbol{\alpha}_F^{b(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k)$, $\mathbf{e}_F^{(n)}(k)=\boldsymbol{\alpha}_F^{(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k)$, where φ and $\boldsymbol{\Phi}$ are so-called likehood ratio scalar and matrix.

The lattice algorithms with feedbacks use the recursive calculations of reflection coefficients instead of steps 10) … 12), 23) …, 25) in sequential algorithms and steps 9)…11) in parallel algorithms. In case of parallel algorithms, the coefficients are calculated as

$$\Gamma^{f(n)}(k)=\Gamma^{f(n)}(k-1)-\mathbf{e}_F^{b(n)}(k-1)\times$$
$$\times\boldsymbol{\alpha}_F^{f(n+1)H}(k)/E^{b(n)}(k-1),$$

$$\Gamma^{b(n)}(k) = \Gamma^{b(n)}(k-1) - \mathbf{e}_F^{f(n)}(k)\boldsymbol{\alpha}_F^{b(n+1)H}(k)\big/E^{f(n)}(k),$$

$$\Gamma^{(n)}(k) = \Gamma^{(n)}(k-1) - \mathbf{e}_F^{b(n)}(k)\boldsymbol{\alpha}_F^{(n+1)H}(k)\big/E^{b(n)}(k).$$

4. Square Root Free QRD Lattice RLS Algorithms

The ideas of normalized and QRD lattice algorithms can be found in [10, 11]. They are applied directly to the considered SW RLS lattice algorithms, regularized PW and regularized SW RLS algorithms with sequential computations. However, the parallel computations cannot be applied to the three known versions of the algorithms, because some of variables appear as uncommutated matrices and some of the matrix products cannot be substituted by other variables as in sequential algorithms where the mentioned variables are scalar ones.

At the same time, the lattice algorithm with square-root free QRD can be viewed as a transformation of a priori error algorithm with feedback and have both sequential and parallel implementations. In the algorithm, reflection coefficients are calculated by means of the elements of Givens rotation matrix, that are sin and cos of some rotation angles, denoted as $\bar{s}(k)$, $\bar{c}(k)$ in sequential algorithms and as $\bar{\mathbf{s}}(k)$, $\bar{c}(k)$ in parallel algorithms.

Below, the sequential square root free QRD lattice SW RLS algorithm is listed.

Initialization : $E_{x_D}^{f(n)}(0) = \delta^2$, $E_{x_D}^{b(n)}(0) = \delta^2\lambda^{-(n-1)}$,

0) $E_{x_U}^{b(n)}(0) = \delta^2\lambda^{-(n-1)}$, $\varphi_{x_U}^{(n)}(0) = 1$, $\varphi_{x_D}^{(n)}(0) = 1$, $\bar{c}_{x_U}^{b(n)}(0) = 1$, $\bar{s}_{x_U}^{b(n)}(0) = 0$, $\bar{c}_{x_D}^{b(n)}(0) = 1$, $\bar{s}_{x_D}^{b(n)}(0) = 0$, $\alpha_{x_U}^{b(n)}(0) = 0$, $\alpha_{x_D}^{b(n)}(0) = 0$, $\Gamma_{x_D}^{f(n)}(0) = 0$, $\Gamma_{x_D}^{b(n)}(0) = 0$, $\gamma_{x_D}^{(n)}(0) = 0$, $n = 1,2,\ldots,N$

For $k = 1,2,\ldots,K$

1) $\varphi_{x_U}^{(1)}(k) = 1$, $\alpha_{x_U}^{f(1)}(k) = x(k)$, $\alpha_{x_U}^{b(1)}(k) = x(k)$, $\alpha_{x_U}^{(1)}(k) = d(k)$

2) $\varphi_{x_D}^{(1)}(k) = 1$, $\alpha_{x_D}^{f(1)}(k) = x(k-L)$, $\alpha_{x_D}^{b(1)}(k) = x(k-L)$, $\alpha_{x_D}^{(1)}(k) = d(k-L)$

For $n = 1,2,\ldots,N$

3) $E_{x_U}^{f(n)}(k) = \lambda E_{x_D}^{f(n)}(k-1) + \alpha_{x_U}^{f(n)}(k)\varphi_{x_U}^{(n)}(k-1) \times \alpha_{x_U}^{f(n)*}(k)$

4) $\bar{c}_{x_U}^{f(n)}(k) = \lambda E_{x_D}^{f(n)}(k-1)\big/E_{x_U}^{f(n)}(k)$

5) $\bar{s}_{x_U}^{f(n)}(k) = \alpha_{x_U}^{f(n)}(k)\varphi_{x_U}^{(n)}(k-1)\big/E_{x_U}^{f(n)}(k)$

6) $\alpha_{x_U}^{f(n+1)}(k) = \alpha_{x_U}^{f(n)}(k) + \Gamma_{x_D}^{f(n)*}(k-1)\alpha_{x_U}^{b(n)}(k-1)$

7) $\Gamma_{x_U}^{f(n)}(k) = \bar{c}_{x_U}^{b(n)}(k-1)\Gamma_{x_D}^{f(n)}(k-1) - \bar{s}_{x_U}^{b(n)}(k-1) \times \alpha_{x_U}^{f(n)*}(k)$

8) $E_{x_U}^{b(n)}(k) = \lambda E_{x_D}^{b(n)}(k-1) + \alpha_{x_U}^{b(n)}(k)\varphi_{x_U}^{(n)}(k)\alpha_{x_U}^{b(n)*}(k)$

9) $\bar{c}_{x_U}^{b(n)}(k) = \lambda E_{x_D}^{b(n)}(k-1)\big/E_{x_U}^{b(n)}(k)$

10) $\bar{s}_{x_U}^{b(n)}(k) = \alpha_{x_U}^{b(n)}(k)\varphi_{x_U}^{(n)}(k)\big/E_{x_U}^{b(n)}(k)$

11) $\alpha_{x_U}^{b(n+1)}(k) = \alpha_{x_U}^{b(n)}(k-1) + \Gamma_{x_D}^{b(n)*}(k-1)\alpha_{x_U}^{f(n)}(k)$

12) $\Gamma_{x_U}^{b(n)}(k) = \bar{c}_{x_U}^{f(n)}(k)\Gamma_{x_D}^{b(n)}(k-1) - \bar{s}_{x_U}^{f(n)}(k)\alpha_{x_U}^{b(n)*}(k)$

13) $\alpha_{x_U}^{(n+1)}(k) = \alpha_{x_U}^{(n)}(k) + \gamma_{x_D}^{(n)*}(k-1)\alpha_{x_U}^{b(n)}(k)$

14) $\gamma_{x_U}^{(n)}(k) = \bar{c}_{x_U}^{b(n)}(k)\gamma_{x_D}^{(n)}(k-1) - \bar{s}_{x_U}^{b(n)}(k)\alpha_{x_U}^{(n)*}(k)$

15) $\varphi_{x_U}^{(n+1)}(k) = \varphi_{x_U}^{(n)}(k)\bar{c}_{x_U}^{b(n)}(k)$

16) $E_{x_D}^{f(n)}(k) = E_{x_U}^{f(n)}(k) - \mu\alpha_{x_D}^{f(n)}(k)\varphi_{x_D}^{(n)}(k-1)\alpha_{x_D}^{f(n)*}(k)$

17) $\bar{c}_{x_D}^{f(n)}(k) = E_{x_U}^{f(n)}(k)\big/E_{x_D}^{f(n)}(k)$

18) $\bar{s}_{x_D}^{f(n)}(k) = \alpha_{x_D}^{f(n)}(k)\varphi_{x_D}^{(n)}(k-1)\big/E_{x_D}^{f(n)}(k)$

19) $\alpha_{x_D}^{f(n+1)}(k) = \alpha_{x_D}^{f(n)}(k) + \Gamma_{x_U}^{f(n)*}(k)\alpha_{x_D}^{b(n)}(k-1)$

20) $\Gamma_{x_D}^{f(n)}(k) = \bar{c}_{x_D}^{b(n)}(k-1)\Gamma_{x_U}^{f(n)}(k) + \mu\bar{s}_{x_D}^{b(n)}(k-1) \times \alpha_{x_D}^{f(n)*}(k)$

21) $E_{x_D}^{b(n)}(k) = E_{x_U}^{b(n)}(k) - \mu\alpha_{x_D}^{b(n)}(k)\varphi_{x_D}^{(n)}(k)\alpha_{x_D}^{b(n)*}(k)$

22) $\bar{c}_{x_D}^{b(n)}(k) = E_{x_U}^{b(n)}(k)\big/E_{x_D}^{b(n)}(k)$

23) $\bar{s}_{x_D}^{b(n)}(k) = \alpha_{x_D}^{b(n)}(k)\varphi_{x_D}^{(n)}(k)\big/E_{x_D}^{b(n)}(k)$

24) $\alpha_{x_D}^{b(n+1)}(k) = \alpha_{x_D}^{b(n)}(k-1) + \Gamma_{x_U}^{b(n)*}(k)\alpha_{x_D}^{f(n)}(k)$

25) $\Gamma_{x_D}^{b(n)}(k) = \bar{c}_{x_D}^{f(n)}(k)\Gamma_{x_U}^{b(n)}(k) + \mu\bar{s}_{x_D}^{f(n)}(k)\alpha_{x_D}^{b(n)*}(k-1)$

26) $\alpha_{x_D}^{(n+1)}(k) = \alpha_{x_D}^{(n)}(k) + \gamma_{x_U}^{(n)*}(k)\alpha_{x_D}^{b(n)}(k)$

27) $\gamma_{x_D}^{(n)}(k) = \bar{c}_{x_D}^{b(n)}(k)\gamma_{x_U}^{(n)}(k) + \mu\bar{s}_{x_D}^{b(n)}(k)\alpha_{x_D}^{(n)*}(k)$

28) $\varphi_{x_D}^{(n+1)}(k) = \varphi_{x_D}^{(n)}(k)\bar{c}_{x_D}^{b(n)}(k)$

End for n

End for k

A parallel form the above algorithm is shown below.

Initialization : $E^{f(n)}(0)=\delta^2, E^{b(n)}(0)=\delta^2\lambda^{-(n-1)},$

0) $\boldsymbol{\alpha}_F^{b(n)}(0)=\mathbf{0}_F, \boldsymbol{\Phi}_F^{(n)}(0)=\mathbf{S}_F, \bar{c}^{b(n)}(0)=1,$
$\bar{\mathbf{s}}_F^{b(n)}(0)=\mathbf{0}_F, \Gamma^{f(n)}(0)=0, \Gamma^{b(n)}(0)=0, \Gamma^{(n)}(0)=0,$
$n=1,2,\ldots,N$

For $k=1,2,\ldots,K$

1) $\boldsymbol{\Phi}_F^{(1)}(k)=\mathbf{S}_F, \boldsymbol{\alpha}_F^{f(1)}(k)=\mathbf{x}_F(k), \boldsymbol{\alpha}_F^{b(1)}(k)=\mathbf{x}_F(k),$
$\boldsymbol{\alpha}_F^{(1)}(k)=\mathbf{d}_F(k)$

For $n=1,2,\ldots,N$

2) $E^{f(n)}(k)=\lambda E^{f(n)}(k-1)+\boldsymbol{\alpha}_F^{f(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k-1)\times$
$\times\boldsymbol{\alpha}_F^{f(n)H}(k)$

3) $\bar{c}^{f(n)}(k)=\lambda E^{f(n)}(k-1)/E^{f(n)}(k)$

4) $\bar{\mathbf{s}}_F^{f(n)}(k)=\boldsymbol{\alpha}_F^{f(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k-1)/E^{f(n)}(k)$

5) $\boldsymbol{\alpha}_F^{f(n+1)}(k)=\boldsymbol{\alpha}_F^{f(n)}(k)+\Gamma^{f(n)*}(k-1)\boldsymbol{\alpha}_F^{b(n)}(k-1)$

6) $\Gamma^{f(n)}(k)=\bar{c}^{b(n)}(k-1)\Gamma^{f(n)}(k-1)-\bar{\mathbf{s}}_F^{b(n)}(k-1)\times$
$\times\boldsymbol{\alpha}_F^{f(n)H}(k)$

7) $E^{b(n)}(k)=\lambda E^{b(n)}(k-1)+\boldsymbol{\alpha}_F^{b(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k)\boldsymbol{\alpha}_F^{b(n)H}(k)$

8) $\bar{c}^{b(n)}(k)=\lambda E^{b(n)}(k-1)/E^{b(n)}(k)$

9) $\bar{\mathbf{s}}_F^{b(n)}(k)=\boldsymbol{\alpha}_F^{b(n)}(k)\boldsymbol{\Phi}_F^{(n)}(k)/E^{b(n)}(k)$

10) $\boldsymbol{\alpha}_F^{b(n+1)}(k)=\boldsymbol{\alpha}_F^{b(n)}(k-1)+\Gamma^{b(n)*}(k-1)\boldsymbol{\alpha}_F^{f(n)}(k)$

11) $\Gamma^{b(n)}(k)=\bar{c}^{f(n)}(k)\Gamma^{b(n)}(k-1)-\bar{\mathbf{s}}_F^{f(n)}(k)\times$
$\times\boldsymbol{\alpha}_F^{b(n)H}(k-1)$

12) $\boldsymbol{\alpha}_F^{(n+1)}(k)=\boldsymbol{\alpha}_F^{(n)}(k)+\gamma^{(n)*}(k-1)\boldsymbol{\alpha}_F^{b(n)}(k)$

13) $\gamma_n(k)=\bar{c}^{b(n)}(k)\gamma_n(k-1)-\bar{\mathbf{s}}_F^{b(n)}(k)\boldsymbol{\alpha}_F^{(n)H}(k)$

14) $\boldsymbol{\Phi}_F^{(n+1)}(k)=\boldsymbol{\Phi}_F^{(n)}(k)\left[\mathbf{I}_F-\boldsymbol{\alpha}_F^{b(n)H}(k)\boldsymbol{\alpha}_F^{b(n)}(k)\times\right.$
$\left.\times\boldsymbol{\Phi}_F^{(n)}(k)/E^{b(n)}(k)\right]$

End for n

End for k

Thus, the diversity of sequential algorithms includes the earlier mentioned ten basic algorithms with three possible modifications: regularized PW, SW and regularized SW. It produces 30 different lattice RLS algorithms. The same algorithms exist in parallel form excluding normalized and QRD ones. It produces 21 different parallel algorithms.

5. Simulation

In Fig. 6 there are the results of comparative simulation of the SW lattice algorithms and some of the fast SW RLS algorithms [1] in the same conditions: line 1 is a priori and a posteriori errors algorithm, line 2 is square root free QRD lattice algorithm, line 3 is stabilized Fast a Posteriori Error Sequential Technique (FAEST) algorithm, line 4 is Fast Kalman (FK) algorithm, line 5 is Fast Transversal Filter (FTF) algorithm.

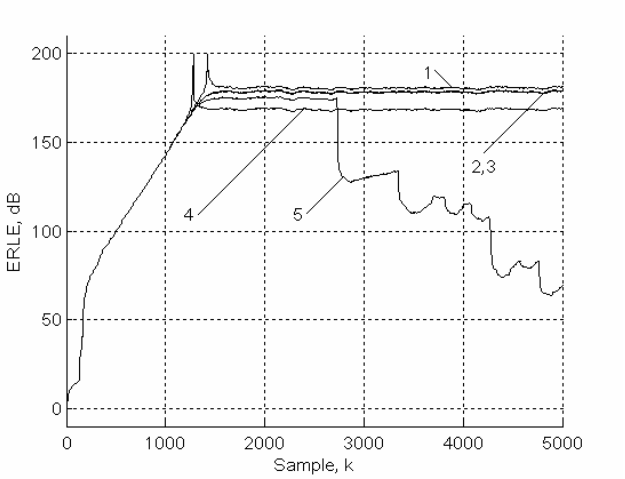

Fig. 6. Lattice filters simulation.

The simulation was conducted in floating point arithmetic. In the simulation, an identification of an N=32 weights filter was studied. Signal $x(k)$ was generated as white noise, and signal $d(k)$ was produced as the convolution of the noise signal with the identified impulse response. Additive noise in $d(k)$ signal was absent. Fig. 5 shows Echo Return Loss Enhancement (ERLE) parameter

$$ERLE(k)=10\log_{10}\left(\sum_{i=k-B+1}^{k}d_U^2(i)\Big/\sum_{i=k-B+1}^{k}\alpha_{x_U}^2(i)\right),$$

that is one of the quality parameters in identification tasks. The parameters B and L were selected as 256. If N is the same for identified and adaptive filters, then the noise due to the filters length mismatch is absent at $d(k)$ input of an adaptive filter. In the case, the steady-state ERLE is bounded by the roundoff errors of the algorithms.

As we can see, the ERLE of the considered lattice algorithms is about the same as that of the stabilized FAEST algorithm. It indirectly confirms the good properties of the lattice RLS algorithms: stability and low roundoff errors. At the same time, FTF algorithm becomes instable after about 2700 iterations.

6. Conclusion

Thus, the paper presents the descriptions of the some ideas for the modification of RLS lattice algorithms for

nonstationary signal processing. The examples of the modifications are presented by the computational procedures of a priori and a posteriori errors and square root free QRD lattice SW RLS algorithms based on sequential and parallel computations. Structures of adaptive filters that use the algorithms are also presented.

The number of arithmetic operations in modified algorithms is about two or four times grater that of PW algorithms. Computational load per processor is decreased by the same degree if parallel computations are used. The lattice filters are used in applications that do not require the explicit computation of weights, where signal of interest is output error signal only. Acoustic and electrical echo cancellations are the examples of the applications.

The considered and some other lattice algorithms for nonstationary signal processing were also presented in [14-16]. They are the part of the permanently developed and updated applied library of adaptive filtering algorithms [17].

References

[1] DJIGAN, V. I. Multichannel RLS and fast RLS adaptive filtering algorithms. *Successes of Modern Radioelectronics,* 2004, no. 11, p. 48 - 77 (in Russian).

[2] WIDROW, B. Thinking about thinking: the discovery of the LMS algorithm – DSP history. *IEEE Signal Processing Magazine,* 2005, vol. 22, no. 1, p. 100 - 106.

[3] DJIGAN, V. I. Efficiency of adaptive signal processing algorithms implementation on basis of MULTICORE SoC. In *Proceedings of the All–Russian Scientific and Technical Conference on Problems of Prospective Microelectronic Systems Development*. Moscow (Russia), 2005, p. 453 - 460 (in Russian).

[4] GAY, S. L. Dynamically regularized fast RLS with application to echo cancellation. In *Proceedings of the International Conference on Acoustic Speech and Signal Processing*. Atlanta (USA), 1996, p. 957 - 960.

[5] DJIGAN V. I. Unified approach to the fast time recursive least square adaptive filtering algorithms development. In *Proceedings of the 3-rd International Conference on Antennas, Radiocommunication Systems & Means (ICARSM–97)*. Voronezh (Russia), 1997, vol. 3, p. 33 - 42.

[6] DJIGAN, V. I. RLS adaptive filtering algorithms based on parallel computations. *Radioengineering: Proceedings of Czech and Slovak Technical Universities and URSI Committees,* 2005, vol. 14, no. 3, p. 28 - 36.

[7] ZELNIKER, G., TAYLOR, F. J. *Advanced Digital Signal Processing: Theory and Applications*. New York: Marcel Dekker, Inc., 1994.

[8] CARAYANNIS, G., MANOLAKIS, D., KALOUPTSIDIS, N. A unified view of parametric processing algorithms for prewindowed signals. *Signal Processing,* 1986, vol. 10, p. 335 - 368.

[9] HAYKIN, S. *Adaptive Filter Theory.* 4-th edition. Prentice Hall, 2001.

[10] SAYED, A. H. *Fundamentals of Adaptive Filtering*. Hoboken, NJ: John Wiley and Sons, Inc., 2003.

[11] DJIGAN, V. I. Lattice RLS adaptive filtering algorithms diversity. *Digital Signal Processing,* 2005, no. 3, p. 2 -12 (in Russian).

[12] ZHAO, K., LING, F., LEV-ARI, H., PROAKIS, J. G. Sliding window order-recursive least-squares algorithms. *IEEE Trans. Signal Processing,* 1994, vol. 42, no. 8, p. 1961 - 1972.

[13] BAYKAL, B., CONSTANTINIDES, A. G. Sliding window adaptive fast QR and QR-lattice algorithms. *IEEE Trans. Signal Processing,* 1998, vol. 46, no. 11, p. 2864 - 2876.

[14] DJIGAN, V. I. Sliding window in lattice RLS algorithms of adaptive filtering. In *Proceedings of the 13-th International Conference on Information Systems and Technologies*. Moscow (Russia), 2005, vol. 1, p. 151 - 154 (in Russian).

[15] DJIGAN, V. I. Regularization of lattice RLS algorithms. In *Proceedings of the 5-th International Scientific-Technical Conference on Electronics and Informatics 2005*. Moscow (Russia), 2005, vol. 2, p. 131 - 132 (in Russian).

[16] DJIGAN, V. I. A family of lattice adaptive filtering algorithms with parallel computations. In *Proceedings of the 14-th International Scientific and Technical Conference on Problem of Information Transmitting and Processing in Telecommunication Networks and Systems.* Ryazan (Russia), 2005, p. 203 - 204 (in Russian).

[17] DJIGAN, V.I. Library of adaptive filtering algorithms. In *Proceedings of the 6-th International Conference on Digital Signal Processing and its Applications (DSPA–2004)*. Moscow (Russia), 2004, vol. 1, pp. 88 - 94.

About Author...

Victor I. DJIGAN was born in 1958 (Ukraine). He received a B.S. degree with honors from the Vinnitsa Technical College of Electronic Devices, Ukraine, in 1978; M.S. degree with honors from the Moscow Institute of Electronic Engineering (MIEE) - Technical University, Russia, in 1984, and Ph.D. degree from MIEE in 1990, all in radio engineering. From 1984 to the present he has been working in the fields of digital signal processing; communication; adaptive antenna arrays; acoustic echo and noise cancellation; transmitting line analysis, simulation and reflectometry. Presently, he is a Principal Researcher in ELVEES R&D Center, Moscow, Russia. His scientific interests include adaptive signal processing, speech processing and digital communication. He is the author of about 130 papers in the above-mentioned fields and a Member of Russian A.S. Popov Society for Radioengineering, Electronics & Communications and IEEE Senior Member (M'96, SM'2004).