Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 2, 2004

A Low Dynamics Fast Transversal Filter Adaptive Algorithm

Constantin Paleologu, Silviu Ciochină, Andrei Alexandru Enescu[1]

Abstract – This paper propose a low dynamics version of FTF adaptive algorithm using a modified form of the cost function, based on an asymptotically unbiased estimator of the mean square error. The reduced dynamics of the modified algorithm's parameters could lead to facility for fixed-point implementation.

Keywords: Adaptive filtering algorithms, FTF algorithm, dynamics of parameters, fixed-point implementation.

I. INTRODUCTION

The Recursive Least Squares (RLS) adaptive algorithm [1], [2] is one of the most popular adaptive algorithms, mainly due to its fast convergence rate. Nevertheless, there are some major drawbacks related to the high computational complexity and the large dynamic range of the algorithm's variables.

The first inconvenient could be solved by using a "fast" least squares (LS) algorithm, in sense that the computational cost increases linearly with the number of adjustable parameters. There are a variety of fast LS algorithms with widely varying properties. The most known are Recursive Least-Squares Lattice (RLSL), QR-Decomposition based Least-Squares Lattice (QRD-LSL) and Fast Transversal Filter (FTF) algorithms [1], [2]. Of these, only the FTF algorithm generates the adaptive filter weights. Unfortunately, FTF algorithm suffers from the numerical instability problem under a finite precision implementation [1].

The second drawback concerning the large dynamics of parameters could cause unwanted finite precision effects. Especially in a fixed-point arithmetic context, overflow or stalling phenomena could occur mainly due to the inherent scaling operations. The "guiltiest" parameter for that large dynamic range is the algorithm cost function. In the case of any LS adaptive algorithm the cost function produces a large biased estimate of the mean square error. In this paper we propose a version of FTF algorithm based on a modified form of the cost function, using an asymptotically unbiased estimator of the mean square error [3], [4]. The reduced dynamics of the modified algorithm's parameters could lead to facility for fixed-point implementation.

The organization of this paper is as follows. The theoretical background of our problem is introduced in the next section. The derivation of the low dynamics FTF adaptive algorithm is performed in section III. Some simulation results are presented in section IV. Finally, the conclusion remarks are given in section V.

II. THEORETICAL BACKGROUND

The LS cost function is defined as an estimate of the mean square-error:

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

where $0<\lambda\ [illegible]\ 1$ is the exponential weighting factor and $e(i)$ is the estimation error at time i. This estimate of the cost function induces similar estimates for the correlation matrix $\mathbf{\Phi}(n)$ and the cross-correlation vector $\mathbf{\theta}(n)$:

$$\mathbf{\Phi}(n)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)=\lambda\mathbf{\Phi}(n-1)+\mathbf{x}(n)\mathbf{x}^H(n) \quad (2)$$

$$\mathbf{\theta}(n)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)d^*(i)=\lambda\mathbf{\theta}(n-1)-\mathbf{x}(n)d^*(n) \quad (3)$$

where $\mathbf{x}(i)$ is the tap-input vector and $d(i)$ is the desired response, both at time i. The superscript H denotes Hermitian transposition (transposition and complex conjugation) and the superscript $*$ denotes the complex conjugation.

The expectations of these functions are

$$E\{J(n)\}\cong\frac{1-\lambda^n}{1-\lambda}E\left\{\left|e(n)\right|^2\right\} \quad (4)$$

$$E\{\mathbf{\Phi}(n)\}\cong\frac{1-\lambda^n}{1-\lambda}\mathbf{R} \quad (5)$$

[1] "Politehnica" University of Bucharest, Electronics and Telecommunications Faculty, 1-3, Iuliu Maniu Bvd., Bucharest, Romania, e-mail: pale@comm.pub.ro, silviu@comm.pub.ro, aenescu@comm.pub.ro

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

$$E\{J(n)\}\Big|_{n\to\infty} \cong \frac{1}{1-\lambda} E\left\{\left|e(n)\right|^2\right\} \qquad (6)$$

$$E\{\mathbf{\Phi}(n)\}\Big|_{n\to\infty} \cong \frac{1}{1-\lambda}\mathbf{R} \qquad (7)$$

Some classes of applications [5] require a high memory algorithm, which means that the value of the exponential weighting factor λ is very close to unity. In this case very large values for these parameters can result, causing unwanted finite precision effects in a practical implementation.
Taking into account the previous discussion we propose an unbiased estimator of the matrix $\mathbf{\Phi}(n)$. So that, we will modify the cost function from equation (1) as follows:

$$\begin{aligned} J(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\left|e(i)\right|^2 = \\ &= \lambda J(n-1)+(1-\lambda)\left|e(n)\right|^2 \end{aligned} \qquad (8)$$

In this case

$$E\{J(n)\} \cong \left(1-\lambda^n\right)E\left\{\left|e(n)\right|^2\right\} \qquad (9)$$

is an asymptotically unbiased estimator of the mean square-error.
Following this idea we have to perform the same modification in equations (2) and (3) obtaining

$$\begin{aligned} \mathbf{\Phi}(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i) = \\ &= \lambda\mathbf{\Phi}(n-1)+(1-\lambda)\mathbf{x}(n)\mathbf{x}^H(n) \end{aligned} \qquad (10)$$

$$\begin{aligned} \mathbf{\theta}(n) &= \sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)d^*(i) = \\ &= \lambda\mathbf{\theta}(n-1)+(1-\lambda)\mathbf{x}(n)d^*(n) \end{aligned} \qquad (11)$$

According.

$$E\{\mathbf{\Phi}(n)\} \cong \left(1-\lambda^n\right)\mathbf{R} \qquad (12)$$

is an asymptotically unbiased estimator of the correlation matrix.
Most of the expressions in the following section may look familiar to readers acquainted with the theory of least-squares transversal filters. However, the derivation that follows is developed according to the new approach.

III. LOW DYNAMICS FTF ADAPTIVE ALGORITHM

Let us consider a forward linear predictor of order N with the vector coefficients at time n denoted by $\mathbf{a}_N(n)$. The forward a posteriori prediction error produced at the output is

$$e_N^f(i) = \mathbf{a}_N^H(n)\mathbf{x}_{N+1}(i) \qquad (13)$$

where $\mathbf{x}_{N+1}(i)$ is the N+1-by-1 the tap-input vector, with $1 \le i \le n$.
The cost function is the sum of weighted forward a posteriori prediction-error squares in the modified form according to (8):

$$\begin{aligned} J_N^f(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\left|e_N^f(i)\right|^2 = \\ &= \lambda J_N^f(n-1)+(1-\lambda)\left|e_N^f(n)\right|^2 \end{aligned} \qquad (14)$$

The corresponding backward linear prediction-error filter with the vector coefficients denoted by $\mathbf{c}_N(n)$ will produced the backward a posteriori prediction error:

$$e_N^b(i) = \mathbf{c}_N^H(n)\mathbf{x}_{N+1}(i) \qquad (15)$$

In this case, the cost function is the sum of weighted backward a posteriori prediction-error squares:

$$\begin{aligned} J_N^b(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\left|e_N^b(i)\right|^2 = \\ &= \lambda J_N^b(n-1)+(1-\lambda)\left|e_N^b(n)\right|^2 \end{aligned} \qquad (16)$$

Let $\mathbf{\Phi}_{N+1}(n)$ denote the N+1-by-N+1 correlation matrix of the tap-input vector $\mathbf{x}_{N+1}(i)$, where $1 \le i \le n$, $\mathbf{\theta}^f(n)$ denote the m-by-1 cross-correlation vector between $x(i)$ and $\mathbf{x}_N(i-1)$, and $\mathbf{\theta}^b(n)$ denote the N-by-1 cross-correlation vector between $\mathbf{x}_N(i)$ and $x(i-N)$. According to (10) and (11) these parameters have the following forms:

$$\begin{aligned} \mathbf{\Phi}_{N+1}(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}_{N+1}(i)\mathbf{x}_{N+1}^H(i) = \\ &= \lambda\mathbf{\Phi}_{N+1}(n-1)+(1-\lambda)\mathbf{x}_{N+1}(n)\mathbf{x}_{N+1}^H(n) \end{aligned} \qquad (17)$$

$$\begin{aligned} \mathbf{\theta}^f(n) &= (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}_m(i-1)x^*(i) = \\ &= \lambda\mathbf{\theta}^f(n-1)+(1-\lambda)\mathbf{x}_m(n-1)x^*(n) \end{aligned} \qquad (18)$$

$$\boldsymbol{\theta}^{b}(n) = (1-\lambda)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}_N(i)x^*(i-N) = \\ = \lambda\boldsymbol{\theta}^{b}(n-1) + (1-\lambda)\mathbf{x}_N(n)x^*(n-N) \qquad (19)$$

In the case of forward linear prediction the normal equation is:

$$\boldsymbol{\Phi}_N(n-1)\mathbf{w}(n) = \boldsymbol{\theta}^{f}(n) \qquad (20)$$

where $\mathbf{w}(n)$ is the tap-weight vector of the forward linear predictor, or

$$\boldsymbol{\Phi}_{N+1}(n)\mathbf{a}_N(n) = \begin{bmatrix} J^{f}_{N\min} \\ 0 \end{bmatrix} \qquad (21)$$

where $J^{f}_{N\min}$ is the minimum value of the sum of weighted forward a posteriori prediction-error squares.

Following the classical procedure it is easy to deduce the following recursion for updating the tap-weight vector of the predictor:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}_N(n-1)\alpha^{f*}_N(n) \qquad (22)$$

where $\alpha^{f}_N(n)$ is the forward a priori prediction error:

$$\alpha^{f}_N(n) = \mathbf{a}^{H}_N(n-1)\mathbf{x}_{N+1}(n) \qquad (23)$$

and $\mathbf{k}_N(n-1)$ is the modified gain vector:

$$\mathbf{k}_N(n-1) = (1-\lambda)\boldsymbol{\Phi}^{-1}_N(n-1)\mathbf{x}_N(n-1) \qquad (24)$$

Taking these into account we may write the recursion for updating the tap-weight vector of the prediction-error filter:

$$\mathbf{a}_N(n) = \mathbf{a}_N(n-1) - \begin{bmatrix} 0 \\ \mathbf{k}^{*}_N(n-1) \end{bmatrix}\alpha^{f}_N(n) \qquad (25)$$

Finally, we get the following recursion for updating the minimum value of the sum of weighted forward prediction-error squares:

$$J^{f}_{N\min}(n) = \lambda J^{f}_{N\min}(n-1) + (1-\lambda)\alpha^{f}_N(n)e^{f*}_N(n) \qquad (26)$$

In a similar manner we obtain a set of relations for the backward prediction part of the algorithm:

$$\boldsymbol{\Phi}_{N+1}(n)\mathbf{c}_N(n) = \begin{bmatrix} 0 \\ J^{b}_{N\min}(n) \end{bmatrix} \qquad (27)$$

$$\mathbf{g}(n) = \mathbf{g}(n-1) + \mathbf{k}_N(n)\alpha^{b*}_N(n) \qquad (28)$$

$$\mathbf{k}_N(n) = (1-\lambda)\boldsymbol{\Phi}^{-1}_N(n)\mathbf{x}_N(n) \qquad (29)$$

$$\alpha^{b}_N(n) = \mathbf{c}^{H}_N(n-1)\mathbf{x}_{N+1}(n) \qquad (30)$$

$$\mathbf{c}_N(n) = \mathbf{c}_N(n-1) - \begin{bmatrix} \mathbf{k}^{*}_N(n) \\ 0 \end{bmatrix}\alpha^{b}_N(n) \qquad (31)$$

$$J^{b}_{N\min}(n) = \lambda J^{b}_{N\min}(n-1) + (1-\lambda)\alpha^{b}_N(n)e^{b*}_N(n) \qquad (32)$$

where:

- $J^{b}_{N\min}$ is the minimum value of the sum of weighted backward a posteriori prediction-error squares;
- $\mathbf{c}_N(n)$ is the tap-weight vector of the backward prediction-error filter;
- $\mathbf{g}(n)$ is the tap-weight vector of the backward predictor;
- $\mathbf{k}_N(n)$ is the modified gain vector;
- $\alpha^{b}_N(n)$ is the backward a priori prediction error.

The next step is to define a modified extended gain vector:

$$\mathbf{k}_{N+1}(n) = (1-\lambda)\boldsymbol{\Phi}^{-1}_{N+1}(n)\mathbf{x}_{N+1}(n) \qquad (33)$$

It can be demonstrated that the inverse of the correlation matrix may be expressed as follows:

$$\boldsymbol{\Phi}^{-1}_{N+1}(n) = \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\Phi}^{-1}_N(n-1) \end{bmatrix} + \frac{1}{J^{f}_{N\min}(n)}\mathbf{a}_N(n)\mathbf{a}^{H}_N(n) \qquad (34)$$

Using the previous relation we get the following recursion for the modified extended gain vector:

$$\mathbf{k}_{N+1}(n) = \begin{bmatrix} 0 \\ \mathbf{k}_N(n-1) \end{bmatrix} + (1-\lambda)\mathbf{a}_N(n)\frac{e^{f}_N(n)}{J^{f}_{N\min}(n)} \qquad (35)$$

Similarly, using an alternative expression for the inverse of the correlation matrix:

$$\boldsymbol{\Phi}^{-1}_{N+1}(n) = \begin{bmatrix} \boldsymbol{\Phi}^{-1}_N(n) & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{J^{b}_{N\min}(n)}\mathbf{c}_N(n)\mathbf{c}^{H}_N(n) \qquad (36)$$

we get the second recursion for the modified extended gain vector:

$$\mathbf{k}_{N+1}(n) = \begin{bmatrix} \mathbf{k}_N(n) \\ 0 \end{bmatrix} + (1-\lambda)\mathbf{c}_N(n)\frac{e^{b}_N(n)}{J^{b}_{N\min}(n)} \qquad (37)$$

The definition of the modified gain vector from relation (29) may also be viewed as the solution of a special case of the normal equations. It defines the tap-weight vector of a transversal filter that contains N taps and that operates of the input data $\mathbf{x}_N(n)$ to produce a least-squares estimate of a special desired response:

$$d(i)=\begin{cases}1, & i=n\\ 0, & 0<i<n\end{cases} \tag{38}$$

The estimation error (modified conversion factor) is defined as follows:

$$\gamma_N(n)=1-(1-\lambda)\mathbf{x}_N^H(n)\mathbf{\Phi}_N^{-1}(n)\mathbf{x}_N(n) \tag{39}$$

Taking into account the expression of the inverse of the correlation matrix from the standard recursive least-squares estimation problem [1], [2] we get:

$$\gamma_N(n)=\frac{1}{1+(1-\lambda)\lambda^{-1}\mathbf{x}_N^H(n)\mathbf{\Phi}_N^{-1}(n)\mathbf{x}_N(n)} \tag{40}$$

Three useful interpretations of the conversion factor are known [1], [2]:

- for recursive least-squares estimation:

$$\gamma_N(n)=\frac{e_N(n)}{\alpha_N(n)} \tag{41}$$

where $e_N(n)$ is the a posteriori estimation error and $\alpha_N(n)$ is the a priori estimation error;

- for adaptive forward linear prediction:

$$\gamma_N(n-1)=\frac{e_N^f(n)}{\alpha_N^f(n)} \tag{42}$$

- for adaptive backward linear prediction:

$$\gamma_N(n)=\frac{e_N^b(n)}{\alpha_N^b(n)} \tag{43}$$

Taking these into account, the following recursions for updating the conversion factor can be obtained:

$$\gamma_{N+1}(n)=\gamma_N(n-1)-(1-\lambda)\frac{\left|e_N^f(n)\right|^2}{J_{N\min}^f(n)} \tag{44}$$

$$\gamma_{N+1}(n)=\gamma_N(n)-(1-\lambda)\frac{\left|e_N^b(n)\right|^2}{J_{N\min}^b(n)} \tag{45}$$

$$\gamma_{N-1}(n)=\lambda\gamma_N(n-1)\frac{J_{N\min}^f(n-1)}{J_{N\min}^f(n)} \tag{46}$$

$$\gamma_{N+1}(n)=\lambda\gamma_N(n)\frac{J_{N\min}^b(n-1)}{J_{N\min}^b(n)} \tag{47}$$

Finally, we have to put together four distinct tasks (forward linear prediction, backward linear prediction, computation of the gain vector and estimation of the desired response) in order to obtain our modified FTF adaptive algorithm.

First, let us define the normalized gain vector:

$$\underline{\mathbf{k}}_N(n)=\frac{\mathbf{k}_N(n)}{\gamma_N(n)} \tag{48}$$

According, some simplified recursions can be obtained:

$$\underline{\mathbf{k}}_{N+1}(n)=\begin{bmatrix}0\\ \underline{\mathbf{k}}_N(n-1)\end{bmatrix}+\frac{(1-\lambda)\alpha_N^f(n)}{\lambda J_{N\min}^f(n-1)}\mathbf{a}_N(n-1) \tag{49}$$

$$\mathbf{a}_N(n)=\mathbf{a}_N(n-1)-\begin{bmatrix}0\\ \underline{\mathbf{k}}_N^*(n-1)\end{bmatrix}e_N^f(n) \tag{50}$$

$$\alpha_N^b(n)=\frac{\lambda}{(1-\lambda)}J_{N\min}^b(n-1)\underline{k}_{N+1,N+1}(n) \tag{51}$$

$$\gamma_N(n)=\frac{\gamma_{N+1}(n)}{1-\alpha_N^b(n)\gamma_{N+1}(n)\underline{k}_{N+1,N+1}^*(n)} \tag{52}$$

where $\underline{k}_{N+1,N+1}^*(n)$ is the last element of the vector $\underline{\mathbf{k}}_{N+1}(n)$.

Similarly, in the case of backward prediction we get:

$$\begin{bmatrix}\underline{\mathbf{k}}_N(n)\\ 0\end{bmatrix}=\underline{\mathbf{k}}_{N+1}(n)-(1-\lambda)\underline{k}_{N+1,N+1}(n)\mathbf{c}_N(n-1) \tag{53}$$

$$\mathbf{c}_N(n)=\mathbf{c}_N(n-1)-e_N^b(n)\begin{bmatrix}\underline{\mathbf{k}}_N^*\\ 0\end{bmatrix} \tag{54}$$

In order to complete the algorithm it is necessary to update the tap-weight vector of the adaptive filter as follows:

$$\alpha_N(n)=d(n)-\mathbf{w}_N^H(n-1)\mathbf{x}_N(n) \tag{55}$$

$$\begin{aligned}\mathbf{w}_N(n)&=\mathbf{w}_N(n-1)+\alpha_N^*(n)\mathbf{k}_N(n)=\\ &=\mathbf{w}_N(n-1)+e_N^*(n)\underline{\mathbf{k}}_N(n)\end{aligned} \tag{56}$$

In this manner we obtain our Low-Dynamics FTF (LD-FTF) adaptive algorithm. It is summarized below.

LD-FTF adaptive algorithm

Predictions

$$\alpha_N^f(n)=\mathbf{a}_N^H(n-1)\mathbf{x}_{N+1}(n)$$

$$e_N^f(n)=\gamma_N(n-1)\alpha_N^f(n)$$

$$J_{N\min}^f(n)=\lambda J_{N\min}^f(n-1)+(1-\lambda)\alpha_N^f(n)e_N^{f*}(n)$$

$$\gamma_{N+1}(n)=\lambda\gamma_N(n-1)\frac{J_{N\min}^f(n-1)}{J_{N\min}^f(n)}$$

$$\underline{\mathbf{k}}_{N+1}(n)=\begin{bmatrix}0\\ \underline{\mathbf{k}}_N(n-1)\end{bmatrix}+\frac{(1-\lambda)\alpha_N^f(n)}{\lambda J_{N\min}^f(n-1)}\mathbf{a}_N(n-1)$$

$$\mathbf{a}_N(n)=\mathbf{a}_N(n-1)-\begin{bmatrix}0\\ \underline{\mathbf{k}}_N^*(n-1)\end{bmatrix}e_N^f(n)$$

$$\alpha_N^b(n)=\frac{\lambda}{(1-\lambda)}J_{N\min}^b(n-1)\underline{k}_{N+1,N+1}(n)$$

$$\gamma_N(n)=\frac{\gamma_{N+1}(n)}{1-\alpha_N^b(n)\gamma_{N+1}(n)\underline{k}_{N+1,N+1}^*(n)}$$

$$e_N^b(n)=\gamma_N(n)\alpha_N^b(n)$$

$$J_{N\min}^b(n)=\lambda J_{N\min}^b(n-1)+(1-\lambda)\alpha_N^b(n)e_N^{b*}(n)$$

$$\begin{bmatrix}\underline{\mathbf{k}}_N(n)\\ 0\end{bmatrix}=\underline{\mathbf{k}}_{N+1}(n)-(1-\lambda)\underline{k}_{N+1,N+1}(n)\mathbf{c}_N(n-1)$$

$$\mathbf{c}_N(n)=\mathbf{c}_N(n-1)-e_N^b(n)\begin{bmatrix}\underline{\mathbf{k}}_N^*\\ 0\end{bmatrix}$$

Filtering

$$\alpha_N(n)=d(n)-\mathbf{w}_N^H(n-1)\mathbf{x}_N(n)$$

$$e_N(n)=\gamma_N(n)\alpha_N(n)$$

$$\mathbf{w}_N(n)=\mathbf{w}_N(n-1)+e_N^*(n)\underline{\mathbf{k}}_N(n)$$

The initialization of the algorithm, i.e. $1\le n\le N+1$ period, is quite complex and requires a lot of paper space in order to be deduced. The most common initialization is for the case when the initial condition is zero. At time n = N, initialization of both the gain vector and the adaptive filter is completed. However, the forward and backward prediction-error filters are both one unit longer. So, their initialization is completed at time $n = N + 1$. We have introduced our modifications into the standard initialization procedure presented in [1] and [2] and we obtain the algorithm as follows.

Initialization of LD-FTF adaptive algorithm

```
$\mathbf{a}_0(1)=\mathbf{c}_0(1)=1,\ \mathbf{k}_0(1)=0,\ \gamma_0(1)=1$

$\mathbf{w}_1(1)=\frac{d^*(1)}{x^*(1)}$

$J_{0\min}^f(n)=(1-\lambda)|x(1)|^2,\quad x(1)\neq 0$

for $n = 2:N+1$
    $\alpha_{n-2}^f(n)=\mathbf{a}_{n-2}^T(n-1)\mathbf{x}_{n-1}(n)$
    $\mathbf{a}_{n-1}(n)=\begin{bmatrix}\mathbf{a}_{n-2}(n-1)\\ -\frac{\alpha_{n-2}^f(n)}{x(1)}\end{bmatrix}$
    $e_{n-2}^f(n)=\gamma_{n-2}(n-1)\alpha_{n-2}^f(n)$
    $J_{n-1\min}^f(n)=\lambda J_{n-2\min}^f(n-1)$
    $J_{n-2\min}^f(n)=J_{n-1\min}^f(n)+(1-\lambda)\alpha_{n-2}^f(n)e_{n-2}^{f*}(n)$
    $\gamma_{n-1}(n)=\frac{J_{n-1\min}^f(n-1)}{J_{n-2\min}^f(n)}\gamma_{n-2}(n-1)$
    $\underline{\mathbf{k}}_{n-1}(n)=\begin{bmatrix}0\\ \underline{\mathbf{k}}_{n-2}(n-1)\end{bmatrix}+\frac{(1-\lambda)\alpha_{n-2}^f}{\lambda J_{n-2\min}^f(n-1)}\mathbf{a}_{n-2}(n-1)$
    if $n = N+1$
        $\mathbf{c}_{n-1}(n)=\begin{bmatrix}-x(1)\gamma_{n-1}(n)\underline{\mathbf{k}}_{n-1}^*(n)\\ 1\end{bmatrix}$
        $J_{n-1\min}^b(n)=(1-\lambda)\gamma_{n-1}(n)|x(1)|^2$
    end
    $\alpha_{n-1}(n)=d(n)-\mathbf{w}_{n-1}^H(n-1)\mathbf{x}_{n-1}(n)$
    $e_{n-1}(n)=\gamma_{n-1}(n)\alpha_{n-1}(n)$
    if $n = N+1$
        $\mathbf{w}_{n-1}(n)=\mathbf{w}_{n-1}(n-1)+\underline{\mathbf{k}}_{n-1}(n)e_{n-1}^*(n)$
    else
        $\mathbf{w}_n(n)=\begin{bmatrix}\mathbf{w}_{n-1}(n-1)\\ \frac{\alpha_n^*(n)}{x^*(1)}\end{bmatrix}$
    end
end
```

IV. SIMULATION RESULTS

For the experimental results we consider a "system identification" configuration. In this class of applications dealing with system identification, 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.

In our experiments we compare the classical FTF algorithm and the proposed LD-FTF algorithm. The input signal is a random sequence with an uniform distribution on the interval $(-1;1)$. The length of the adaptive filter is $N = 5$. The results are presented in Fig. 1 and Fig. 2, using an exponential weighting factor $\lambda = 0.999$.

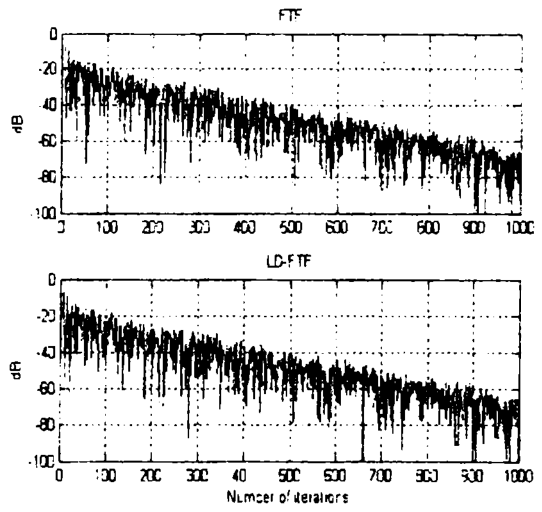

Fig.1. Convergence rate for FTF and LD-FTF adaptive algorithms

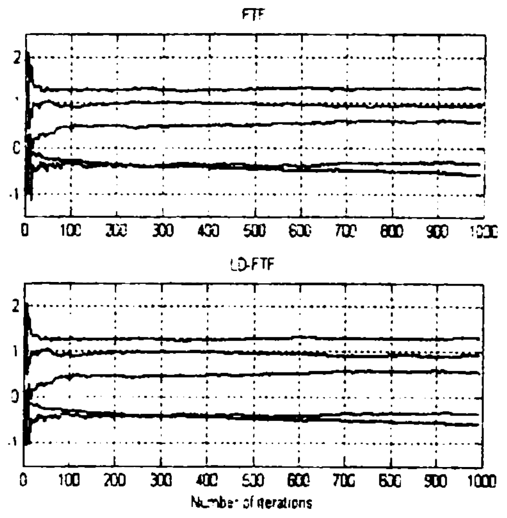

Fig.2. Evolution of adaptive filter coefficients for FTF and LD-FTF adaptive algorithms

It can be noticed that the performances of both adaptive algorithms are the same. Hence, the LD-FTF algorithm keeps the fast rate of convergence and specific to the family of fast LS algorithms. Moreover, the reduced dynamics of the modified algorithm's parameters could lead to facility for fixed-point implementation.

V. CONCLUSIONS AND PERSPECTIVES

In this paper we have proposed a modified version of the FTF adaptive algorithm, named LD-FTF, with low dynamics of the parameters, as a result of a different approach of the least squares estimation problem.

The basic idea was to use a modified form for the algorithm's cost functions in order to obtain asymptotically unbiased estimators for the mean square errors. In this manner we reduce the dynamic range of the algorithm parameters, preventing the unwanted overflow or stalling phenomena which may appear when such a algorithm is implemented using fixed-point arithmetic.

The simulation results prove that LD-FTF adaptive algorithm keeps the fast rate of convergence specific to the family of fast LS algorithms.

This paper represents only the first step of our research. Future work will focus on fixed-point DSP implementation of this algorithm. Also, a careful analysis of numerical stability of LD-FTF algorithm could be considered in perspective.

REFERENCES

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

[2] S. Ciochină, C. Negrescu, *Adaptive Systems*, Ed. Tehnică, Bucharest, 1999.

[3] S. Ciochină, C. Paleologu, 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*, Iaşi, Romania, 2003, vol. 1, pp. 57-60.

[4] C. Paleologu, S. Ciochină, A.A. Enescu, "A Low Dynamics Recursive Least-Squares Lattice Adaptive Algorithm", *Proc. of IEEE ICSES'04*, Poznan, Poland, 2004, pp. 195-198.

[5] C. Paleologu, S. Ciochină, A.A. Enescu, "A Network Echo Canceler Based on a SRF QRD-LSL Adaptive Algorithm Implemented on Motorola StarCore SC140 DSP", *Proc. of IEEE ICT2004*, Fortaleza, Brasil, 2004, LNCS Springer-Verlag, vol. 3124, pp. 560-567.