The Fast Subsampled-Updating Stabilized Fast Transversal Filter (FSU SFTF) RLS Algorithm

Dirk T.M. SLOCK† and Karim MAOUCHE ††

†, †† Institut EURECOM, 2229 route des Crêtes
B.P. 193, 06904, Sophia Antipolis Cedex, FRANCE

† *Tel/Fax: +33 93002606/ 93002627, E-Mail: slock@eurecom.fr*

†† *Tel/Fax: +33 93002632/ 93002627, E-Mail: maouche@eurecom.fr*

Abstract. We present a new fast algorithm for Recursive Least-Squares (RLS) adaptive filtering that uses displacement structure and subsampled updating. The FSU SFTF algorithm is based on the Stabilized Fast Transversal Filter (SFTF) algorithm, which is a numerically stabilized version of the classical FTF algorithm. The FTF algorithm exploits the shift invariance that is present in the RLS adaptation of a FIR filter. The FTF algorithm is in essence the application of a rotation matrix to a set of filters and in that respect resembles the Levinson algorithm. In the subsampled updating approach, we accumulate the rotation matrices over some time interval before applying them to the filters. It turns out that the successive rotation matrices themselves can be obtained from a Schur type algorithm which, once properly initialized, does not require inner products. The various convolutions that thus appear in the algorithm are done using the Fast Fourier Transform (FFT). For relatively long filters, the computational complexity of the new algorithm is smaller than the one of the well-known LMS algorithm, rendering it especially suitable for applications such as acoustic echo cancellation.

1 Introduction

Fast Recursive Least Squares (RLS) algorithms such as the Fast Transversal Filter (FTF) algorithm [1] exploit a certain shift invariance structure in the input data vector to reduce the computational complexity from $\mathcal{O}(N^2)$ for RLS to $\mathcal{O}(N)$ for FTF (N being the FIR filter length). In [3],[4],[5], we have pursued an alternative way to reduce the complexity of RLS adaptive filtering algorithms. The approach consists of subsampling the filter adaptation, i.e. the LS filter estimate is no longer provided every sample but every $L \geq 1$ samples (subsampling factor L). This strategy has led us to derive two new RLS algorithms that are the FSU RLS and FSU FTF algorithms which present a reduced complexity when dealing with long filters.

Here, we extend the FSU FTF idea to the Stabilized FTF algorithm (SFTF) which is a numerically stabilized version of the FTF. The starting point is an interpretation of the SFTF algorithm as a rotation applied to the vectors of filter coefficients. Using the filter estimates at a certain time instant, we compute the filter outputs over the next L time instants. Using what we shall call a SFTF-Schur algorithm, it will be possible to compute from these multi-step ahead predicted filter outputs the one step ahead predicted filter outputs in an efficient way. These quantities will allow us to compute the successive rotation matrices of the SFTF algorithm for the next L time instants. Because of the presence of a shift operation in the SFTF algorithm, it turns out to be most convenient to work with the z-transform of the rotation matrices and the filters. Applying the L rotation matrices to the filter vectors becomes an issue of multiplying polynomials, which can be efficiently carried out using the FFT. The subsampled updating technique turns out to be especially applicable in the case of very long filters such as occur in the acoustic echo cancellation problem. The computational gain it offers is obtained in exchange for some processing delay, as is typical of block processing.

In order to formulate the RLS adaptive filtering problem and to fix notation, we shall first recall the RLS algorithm.

2 The RLS Algorithm

An adaptive transversal filter $W_{N,k}$ forms a linear combination
of N consecutive input samples $\{x(i-n), n=0,\ldots,N-1\}$ to approximate (the negative of) the desired-response signal $d(i)$. The resulting error signal is given by

$$\epsilon_N(i|k) = d(i) + W_{N,k}\, X_N(i) = d(i) + \sum_{n=0}^{N-1} W_{N,k}^{n+1}\, x(i-n) \quad (1)$$

where $X_N(i) = \left[x^H(i)\; x^H(i-1) \cdots x^H(i-N+1)\right]^H$ is the input data vector and superscript ^{H} denotes Hermitian (complex conjugate) transpose. In the RLS algorithm, the set of N transversal filter coefficients $W_{N,k} = \left[W_{N,k}^1 \cdots W_{N,k}^N\right]$ are adapted so as to minimize recursively the following LS criterion

$$\begin{aligned} \xi_N(k) &= \min_{W_N} \left\{ \sum_{i=1}^{k} \lambda^{k-i} \left\| d(i) + W_N\, X_N(i) \right\|^2 \right\} \\ &= \sum_{i=1}^{k} \lambda^{k-i} \left\| \epsilon_N(i|k) \right\|^2 \end{aligned} \quad (2)$$

where $\lambda \in (0,1]$ is the exponential weighting factor, $\|v\|_\Lambda^2 = v\Lambda v^H$, $\|.\| = \|.\|_I$. Minimization of the LS criterion leads to the following minimizer

$$W_{N,k} = -P_{N,k}^H R_{N,k}^{-1} \quad (3)$$

where

$$\begin{aligned} R_{N,k} &= \lambda R_{N,k-1} + X_N(k) X_N^H(k) \\ P_{N,k} &= \lambda P_{N,k-1} + X_N(k) d^H(k) \end{aligned} \tag{4}$$

are the sample second order statistics. Substituting the time recursions for $R_{N,k}$ and $P_{N,k}$ from (4) into (3) and using the matrix inversion lemma for $R_{N,k}^{-1}$, we obtain the RLS algorithm:

$$\widetilde{C}_{N,k} = -X_N^H(k)\lambda^{-1}R_{N,k-1}^{-1} \tag{5}$$

$$\gamma_N^{-1}(k) = 1 - \widetilde{C}_{N,k} X_N(k) \tag{6}$$

$$R_{N,k}^{-1} = \lambda^{-1}R_{N,k-1}^{-1} - \widetilde{C}_{N,k}^H \gamma_N(k) \widetilde{C}_{N,k} \tag{7}$$

$$\epsilon_N^p(k) = \epsilon_N(k|k-1) = d(k) + W_{N,k-1}X_N(k) \tag{8}$$

$$\epsilon_N(k) = \epsilon_N(k|k) = \epsilon_N^p(k)\,\gamma_N(k) \tag{9}$$

$$W_{N,k} = W_{N,k-1} + \epsilon_N(k)\widetilde{C}_{N,k} \tag{10}$$

where $\epsilon_N^p(k)$ and $\epsilon_N(k)$ are the a priori and a posteriori error signals (resp. predicted and filtered errors in the Kalman filtering terminology) and one can verify (or see [1]) that they are related by the likelihood variable $\gamma_N(k)$ as in (9). $\widetilde{C}_{N,k}$ is the Kalman gain of order N at time k.

3 The Stabilized Fast Transversal Filter Algorithm

The computational complexity of the FTF algorithm is $7N$ in its most efficient form [1]. However, the FTF suffers from numerical instabilities that are due to long-term round-off error propagation. In [2], a stabilization procedure has been introduced to overcome the instability problem. This has led to the Stabilized FTF algorithm (SFTF) with computational complexity $8N$. In what follows we shall consider the single-channel case. However the generalization to the multichannel case can easily be done. The SFTF algorithm can be described in the following way, which emphasizes its rotational structure:

$$\begin{bmatrix} \left[\widetilde{C}_{N,k}\ 0\right] \\ A_{N,k} \\ B_{N,k} \\ [W_{N,k}\ 0] \end{bmatrix} = \Theta_k \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-1}\right] \\ A_{N,k-1} \\ B_{N,k-1} \\ [W_{N,k-1}\ 0] \end{bmatrix}$$

$$\begin{aligned} e_N^p(k) &= A_{N,k-1}X_{N+1}(k) \\ e_N(k) &= e_N^p(k)\gamma_N(k-1) \\ \gamma_{N+1}^{-s}(k) &= \gamma_N^{-1}(k-1) - \widetilde{C}_{N+1,k}^0 e_N^p(k) \\ \gamma_N^{-s}(k) &= \gamma_{N+1}^{-s}(k) + \widetilde{C}_{N+1,k}^N r_N^{pf}(k) \\ r_N^{ps}(k) &= -\lambda\beta_N(k-1)\widetilde{C}_{N+1,k}^{N\,H} \\ r_N^{pf}(k) &= B_{N,k-1}X_{N+1}(k) \\ \alpha_N^{-1}(k) &= \lambda^{-1}\alpha_N^{-1}(k-1) - \widetilde{C}_{N+1,k}^{0\,H}\gamma_{N+1}^s(k)\widetilde{C}_{N+1,k}^0 \\ r_N^{p(j)}(k) &= K_j r_N^{pf}(k) + (1-K_j) r_N^{ps}(k) \qquad j=1,2 \\ r_N^{(j)}(k) &= r_N^{p(j)}(k)\gamma_N^s(k) \qquad j=1,2 \\ \beta_N(k) &= \lambda\beta_N(k-1) + r_N^{(2)}(k)\, r_N^{p(2)^H}(k) \\ \gamma_N(k) &= \lambda^N \beta_N(k)\,\alpha_N^{-1}(k) \end{aligned} \tag{11}$$

where $A_{N,k}$ and $B_{N,k}$ are the forward and backward prediction filters, $e_N^p(k)$ and $e_N(k)$ are the a priori and a posteriori forward prediction errors, $r_N^p(k)$ and $r_N(k)$ are the a priori and a posteriori backward predition errors, $\widetilde{C}_{N+1,k} = \left[\widetilde{C}_{N+1,k}^0 \cdots \widetilde{C}_{N+1,k}^N\right]$ and $\alpha_N(k)$ and $\beta_N(k)$ are the forward and backward prediction error variances. $K_1 = 1.5$ and $K2 = 2.5$ are the optimal feedback gains that ensure the stability of the dynamics of the accumulated round-off errors [2]. Θ_k is a 4×4 rotation matrix given by

$$\Theta_k = \Theta_k^4\ \Theta_k^3\ \Theta_k^2\ \Theta_k^1 \tag{12}$$

where the four 4×4 matrices $\Theta_k^i \ \ i = 1,2,3,4$ are

$$\Theta_k^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & 0 & 0 & 1 \end{bmatrix} \ \Theta_k^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Theta_k^2 = \begin{bmatrix} 1 & 0 & c & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ \Theta_k^1 = \begin{bmatrix} 1 & e & 0 & 0 \\ d & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

with $a = \epsilon_N(k)$, $b = r_N^{(1)}(k)$, $c = -\widetilde{C}_{N+1,k}^N$, $d = e_N(k)$ and $e = -e_N^p(k)\,\lambda^{-1}\alpha_N^{-1}(k-1)$.
In order to compute the rotation matrices, one must obtain the a priori errors $e_N^p(k), r_N^{pf}(k)$ and $\epsilon_N^p(k)$ which are the outputs at time k of the filters $A_{N,k-1}, B_{N,k-1}$ and $W_{N,k-1}$.

4 The SFTF-Schur Algorithm

Now we introduce subsampled updating and from the filters at time instant $k-L$, we want to obtain the filters at time instant k. This will require the rotation matrices and hence the a priori errors in that time range. We shall show that these quantities can be computed without generating (completely) the intermediate filter estimates using a SFTF-Schur algorithm. Let us introduce the negative of the filter output

$$\widehat{d}_N^p(k) = d(k) - \epsilon_N^p(k) \quad , \quad \widehat{d}_N(k) = d(k) - \epsilon_N(k) \quad . \tag{14}$$

Consider now the following set of filtering operations

$$F_L(k) \triangleq \begin{bmatrix} \eta_{N,L,k}^H \\ e_{N,L,k}^{p\ H} \\ r_{N,L,k}^{pf\ H} \\ -\widehat{d}_{N,L,k}^{p\ H} \end{bmatrix} \triangleq \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-L}\right] \\ A_{N,k-L} \\ B_{N,k-L} \\ [W_{N,k-L}\ 0] \end{bmatrix} X_{N+1,L,k}^H \tag{15}$$

where

$$X_{N+1,L,k} = \left[X_{N+1}(k-L+1) \cdots X_{N+1}(k)\right]^H \tag{16}$$

is the $L \times (N+1)$ Toeplitz input data matrix. $F_L(k)$ is a $4 \times L$ matrix, the rows of which are the result of the filtering of the data sequence $\{x(j)\ ,\ j = k-N-L+1, \ldots, k\}$ by the four filters of the SFTF algorithm. $\eta_{N,L,k}$ is the output of the Kalman gain and $e_{N,L,k}^p$ and $r_{N,L,k}^{pf}$ are respectively the vectors of forward and backward prediction errors

$$e_{N,L,k}^p = \begin{bmatrix} e_N^H(k-L+1|k-L) \\ \vdots \\ e_N^H(k|k-L) \end{bmatrix}, r_{N,L,k}^{pf} = \begin{bmatrix} r_N^{f\ H}(k-L+1|k-L) \\ \vdots \\ r_N^{f\ H}(k|k-L) \end{bmatrix} \tag{17}$$

The last row of $F_L(k)$ corresponds to the (multi-step ahead predicted) adaptive filter outputs

$$\widehat{d}^{\,p}_{N,L,k} = d_{L,k} - \epsilon^p_{N,L,k} = \begin{bmatrix} \widehat{d}^{\,H}_N(k-L+1|k-L) \\ \vdots \\ \widehat{d}^{\,H}_N(k|k-L) \end{bmatrix} . \quad (18)$$

The first column of $F_L(k)$ is

$$F_L(k)\; u_{L,1} \;=\; \begin{bmatrix} 1-\gamma_N^{-1}(k-L) \\ e^p_N(k-L+1) \\ r^{pf}_N(k-L+1) \\ -\widehat{d}^{\,p}_N(k-L+1) \end{bmatrix} \quad (19)$$

where $u_{L,n}$ is the $L \times 1$ vector with 1 at the n^{th} position and 0 elsewhere. In order to obtain Θ_{k-L+1}, one needs to compute $r^{ps}_N(k-L+1)$ and hence $\widetilde{C}^N_{N+1,k-L+1}$. In fact, it turns out that the different $\widetilde{C}^N_{N+1,k-L+j}$ for $j = 1,\ldots,L$ can be obtained by carrying out the SFTF recursions on the last $L-j$ entries of the filters.
For $j = 1,\ldots,L$ do

$$\begin{aligned}
m &= N-L+j \\
K &= k-L+j \\
\widetilde{C}^{m:N}_{N+1,K} &= \widetilde{C}^{m-1:N-1}_{N,K-1} - \lambda^{-1}\alpha^{-1}(K-1)\, e^{p\,H}_N(K)\, A^{m:N}_{N,K-1} \\
\left[\widetilde{C}^{m:N-1}_{N,K}\;\; 0\right] &= \widetilde{C}^{m:N}_{N+1,K} - \widetilde{C}^{N+1}_{N+1,K}\, B^{m:N}_{N,K-1} \\
A^{m+1:N}_{N,K} &= A^{m+1:N}_{N,K-1} + e_N(K)\, \widetilde{C}^{m:N-1}_{N,K-1} \\
B^{m+1:N}_{N,K} &= B^{m+1:N}_{N,K-1} + r^{(1)}_N(K)\left[\widetilde{C}^{m+1:N-1}_{N,K}\;\; 0\right] \\
r^{ps}_N(K) &= -\lambda\beta_N(K-1)\, \widetilde{C}^{N\,H}_{N+1,K} \; .
\end{aligned} \quad (20)$$

Counting only the most significant term as we often do, the computational complexity of these recursions is $2L^2$. So with the quantities in $F_L(k)\; u_{L,1}$ and the recursions (11) and (20), it is possible to construct Θ_{k-L+1}. Now we rotate both expressions for $F_L(k)$ in (15) with Θ_{k-L+1} to obtain $\Theta_{k-L+1}F_L(k)$ which equals

$$\begin{bmatrix} \left[\widetilde{C}_{N,k-L+1}\;\; 0\right] \\ A_{N,k-L+1} \\ B_{N,k-L+1} \\ \left[W_{N,k-L+1}\;\; 0\right] \end{bmatrix} X^H_{N+1,L,k} \;=$$

$$\begin{bmatrix} \multicolumn{2}{c}{\boxed{\eta^H_{N,L-1,k}} \quad *} \\ e_N(k-L+1) & \boxed{e^{p\,H}_{N,L-1,k}} \\ r^f_N(k-L+1) & \boxed{r^{pf\,H}_{N,L-1,k}} \\ -\widehat{d}_N(k-L+1) & \boxed{-\widehat{d}^{\,p\,H}_{N,L-1,k}} \end{bmatrix} \quad (21)$$

Or we can write more compactly

$$\mathcal{S}\left(\Theta_{k-L+1}\, F_L(k)\right) \;=\; F_{L-1}(k) \quad (22)$$

where the operator $\mathcal{S}(M)$ stands for: shift the first row of the matrix M one position to the right and drop the first column of the matrix thus obtained. Now this process can be repeated until we get $F_0(k)$ which is a matrix with no dimensions. So the same rotations that apply to the filters at times $k-l$, $l = L-1,\ldots,0$, also apply to the set of filtering error vectors $F_l(k)$ over the same time span. Furthermore, at each rotation instance, the rotation parameters can be calculated from the first column of $F_l(k)$, the recursions (11) and (20). Inner products (filtering operations) are only needed for the initialization (computation of $F_L(k)$). This is the SFTF-Schur algorithm, which contrasts with the Levinson-style SFTF algorithm in (11).

5 The FSU SFTF Algorithm

Once we have computed the L consecutive rotation matrices with the SFTF-Schur algorithm, we want to apply them all at once to obtain the filters at time k from the filters at time $k-L$. Due to the shift of the Kalman gain in (11), we need to work in the z-transform domain. So we shall associate polynomials with the filter coefficients as follows

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(z) \\ W_k(z) \end{bmatrix} = \begin{bmatrix} \left[\widetilde{C}_{N,k}\; 0\right] \\ A_{N,k} \\ B_{N,k} \\ \left[W_{N,k}\; 0\right] \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N} \end{bmatrix} . \quad (23)$$

Hence (11) can be written in the z-transform domain as

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ W_k(z) \end{bmatrix} = \Theta_k \begin{bmatrix} z^{-1} & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \widetilde{C}_{k-1}(z) \\ A_{k-1}(z) \\ B_{k-1}(z) \\ W_{k-1}(z) \end{bmatrix} . \quad (24)$$

It appears natural to introduce

$$\Theta_k(z) \;=\; \Theta_k \begin{bmatrix} z^{-1} & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} . \quad (25)$$

Now, in order to adapt the filters at time k from the ones at time $k-L$, we get straightforwardly

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ W_k(z) \end{bmatrix} \;=\; \Theta_{k,L}(z) \begin{bmatrix} \widetilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ W_{k-L}(z) \end{bmatrix} \quad (26)$$

where

$$\Theta_{k,L}(z) = \Theta_k(z)\,\Theta_{k-1}(z)\cdots\Theta_{k-L+1}(z) \;. \quad (27)$$

As mentioned before, the successive rotation matrices can be obtained via the SFTF-Schur algorithm with a computational complexity of $4.5L^2$ operations, which takes into account the fact that a rotation matrix in factored form as in (13) only contains five non-trivial entries. Now also remark that $\Theta_{k,L}(z)$ has the following structure

$$\Theta_{k,L}(z) = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 1 \end{bmatrix} \quad (28)$$

Table I: the FSU SFTF Algorithm

#	Computation	Cost per L samples
1	$\begin{bmatrix} \eta^{H}_{N,L,k} \\ e^{p\ H}_{N,L,k} \\ r^{pf\ H}_{N,L,k} \\ -\widehat{d}^{\,p\ H}_{N,L,k} \end{bmatrix} = \begin{bmatrix} [0\ \widetilde{C}_{N,k-L}] \\ A_{N,k-L} \\ B_{N,k-L} \\ [W_{N,k-L}\ 0] \end{bmatrix} X^{H}_{N+1,L,k}$	$(5+4\frac{N+1}{L})FFT(2L)+8N$
2	SFTF-Schur Algorithm: Input: $\eta_{N,L,k},\ e^{p}_{N,L,k},\ r^{pf}_{N,L,k},\ -\widehat{d}^{\,p}_{N,L,k}$ Output: $\Theta_{k-i}(z)\ ,\ i=L{-}1,\cdots,0$	$4.5L^2$
3	$\Theta_{k,L}(z) = \prod_{i=0}^{L-1}\Theta_{k-i}(z)$	$7.5L^2$
4	$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(z) \\ W_k(z) \end{bmatrix} = \Theta_{k,L}(z) \begin{bmatrix} \widetilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ W_{k-L}(z) \end{bmatrix}$	$(12+4\frac{N+1}{L})FFT(2L)+24N$
Total cost per sample		$(17+8\frac{N+1}{L})\frac{FFT(2L)}{L}+32\frac{N}{L}+12L$

where the stars stand for polynomials in z^{-1} of degree at most L. Taking into account these two remarks, the accumulation of the successive rotation matrices to form $\Theta_{k,L}(z)$ takes $7.5L^2$ operations. As a result of the structure displayed in (28), the product in (26) represents 12 convolutions of a polynomial of order L with a polynomial of order N. These convolutions can be done using fast convolution techniques. In the case we consider, in which the orders of the polynomials are relatively large, we will implement the convolutions using the FFT technique. In that case the complexity for the update of each one of the four filters is $3(1+2\frac{N+1}{L})FFT(2L)+2(N+1))$ (multiply/add) operations plus $6(N+1)$ additions ($FFT(m)$ denotes the computational complexity for computing a FFT of length m, and we assume that L is a power of 2 and that $\frac{N+1}{L}$ is an integer). The computation of $F_L(k)$ in (15) can also be done with the FFT and one should compute the FFTs of the filters only once. In the Overlap-Save method, the data matrix is decomposed into $\frac{N+1}{L}$ blocks of $L\times L$ Toeplitz matrices, which are then embedded into $2L\times 2L$ Toeplitz matrices. Note that at time k, only the most recent $2L$ samples of the input signal, corresponding to the new $L\times L$ block in the data matrix, have to be Fourier transformed. The other parts have been computed at previous instants (see [5] for more details). The resulting FSU SFTF algorithm is summarized in Table I.

6 Concluding Remarks

The complexity of the FSU SFTF is $\mathcal{O}((8\frac{N+1}{L}+17)\frac{FFT(2L)}{L}+32\frac{N}{L}+12L)$ operations per sample. This can be very interesting for long filters. For example, when $(N,L)=(4095,256)$, $(8191,256)$ and the FFT is done via the split radix ($FFT(2m)=mlog_2(2m)$ real multiplications for real signals) the multiplicative complexity is respectively $1.2N$ and $0.8N$ per sample. This should be compared to $8N$ for the SFTF algorithm, the currently fastest stable RLS algorithm, and $2N$ for the LMS algorithm. The number of additions is somewhat higher. The cost we pay is a processing delay which is of the order of L samples. We have simulated the algorithm and have verified that it works. In [3],[5], we have introduced the FSU RLS algorithm, an alternative algorithm with a very similar computational complexity, but a very different internal structure. These developments leads us to conjecture that perhaps a lower bound on computational complexity has been reached for RLS algorithms when the subsampled updating strategy is applied and when the filters to be adapted are relatively long.

REFERENCES

[1] J.M. Cioffi and T. Kailath. "Fast, recursive least squares transversal filters for adaptive filtering". *IEEE Trans. on ASSP*, ASSP-32(2):304–337, April 1984.

[2] D.T.M. Slock and T. Kailath. "Numerically Stable Fast Transversal Filters for Recursive Least-Squares Adaptive Filtering". *IEEE Trans. Signal Proc.*, ASSP-39(1):92–114, Jan. 1991.

[3] D.T.M. Slock and K. Maouche. "The Fast Subsampled-Updating Recursive Least-Squares (FSU RLS) and Fast Transversal Filter (FSU FTF) Algorithms for Adapting Long FIR Filters". In André Gilloire, editor, *Proc. Third International Workshop on Acoustic Echo Control*, pages 75–86, Plestin les Grèves, France, Sept. 7-8 1993.

[4] D.T.M. Slock and K. Maouche. "The Fast Subsampled-Updating Fast Transversal Filter (FSU FTF) Algorithm for Adapting Long FIR Filters". Submitted to Annals of Telecommunications.

[5] D.T.M. Slock and K. Maouche. "The Fast Subsampled-Updating Recursive Least-Squares (FSU RLS) Algorithm for Adaptive Filtering Based on Displacement Structure and the FFT". *Signal Processing*, 40(2), 1994.