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

Research Report N° 95-021

The Generalized Sliding Window Recursive Least-Squares (SGW RLS) Algorithm

Karim Maouche Dirk T.M. Slock

February, 1995

Telephone:	+33 93 00 26 26	E-mail:
Karim Maouche:	+33 93 00 26 32	maouche@eurecom.fr
Dirk T.M. Slock:	+33 93 00 26 06	slock@eurecom.fr
Fax:	+33 93 00 26 27	

Abstract

The Recursive Least-Squares algorithm with a Sliding Rectangular Window (SWCRLS) exhibits a better tracking ability than the RLS algorithm with an exponential window (WRLS). The exploitation of a certain shift-invariance property that is inherent to the adaptive filtering problem allows the derivation of fast versions and leads to the Fast Transversal Filter (FTF) and SWCFTF algorithms whose complexities are $\mathcal{O}(N)$, N being the filter length. The SWCRLS algorithm has a major drawback which is noise amplification whereas the WRLS algorithm is less sensible to noise amplification because of the larger memory of the exponential window. In our communication, we derive a new RLS algorithm that is the Generalized Sliding Window RLS (SGW RLS) algorithm. This algorithm uses a generalized window which consists of the superposition of an exponential window for the L' most recent data and the same but attenuated exponential window for the rest of the data. This new window reduces noise amplification in the SWCRLS algorithm. Moreover, we prove theoritically that the use of this window leads to a better compromise between estimation noise and lag noise. Furthermore, after providing a fast version to the GSW RLS algorithm that is the GSW FTF algorithm, we apply the Subsampled-Upadating technique to derive the FSU GSW FTF algorithm, a doubly-fast version of the GSW RLS algorithm.

Contents

1 Introduction

The Recursive Least-Squares algorithm with a Sliding Rectangular Window (SWCRLS) exhibits a better tracking ability than the RLS algorithm with an exponential window (WRLS). This is explained by the fact that the rectangular window allows to forget the past more abruptly than the exponential window does. The complexity of the RLS algorithm is $\mathcal{O}(N^2)$ operations, N being the filter length. Nevertheless, the exploitation of a certain shift-invariance property that is inherent to the adaptive filtering problem allows the derivation of fast versions and leads to the Fast Transversal Filter (FTF) and SWCFTF algorithms whose complexities are $7N$ and $14N$ respectively. Unfortunately, the SWCRLS algorithm has a major drawback which is noise amplification. This noise amplification is due to the fact that the estimation of the covariance matrix of order N is done using a rectangular window of a relatively short length L' ($L' > N$). With such window, the covariance matrix can be ill-conditionned if the input signal is not active in a significant portion of the window. The WRLS algorithm is less sensible to noise amplification because of the larger memory of the exponential window.
In our communication, we propose the GSW RLS algorithm, a new RLS algorithm that generalizes the WRLS and SWCRLS algorithms. This algorithm uses a generalized window which consists of the superposition of an exponential window for the L' most recent data and the same but attenuated exponential window for the rest of the data. This new window reduces noise amplification in the SWCRLS algorithm. Moreover, we prove theoritically that the use of this window leads to a better compromise between estimation noise and lag noise. The SGW RLS algorithm turns out to have the same structure and complexity as the SWC RLS algorithm. Fast Recursive Least Squares (RLS) algorithms such as the Fast Transversal Filter (FTF) algorithm [2] 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). The common structure of the GSW RLS and the SWC RLS algorithms allows to derive easily a fast version of the GSW RLS algorithm by using the structure of the SWC FTF algorithm. Moreover,he obtained GSW FTF algorithm has also the same complexity than the SWC FTF algorithm ($14N$).
In [?],[?],[16], 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 new RLS algorithms that are the FSU RLS, FSU SFTF and FSU FNTF algorithms which present a reduced complexity when dealing with long filters.
Here, we apply this technique to the SWC FTF algorithm. The starting point is an interpretation of the SWC FTF 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 SWC FTF-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 SWC FTF algorithm for the next L time instants. Because of the presence of a shift operation in the SWC FTF 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 $W_{N,k}$ combines linearly 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

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

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

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

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

$$W_{N,k} \;=\; -P_{N,k}^H R_{N,k}^{-1} \tag{3}$$

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 (MIL) for $R_{N,k}^{-1}$, we obtain the RLS algorithm:

$$\begin{aligned} \widetilde{C}_{N,k} &= -X_N^H(k) \lambda^{-1} R_{N,k-1}^{-1} \\ \gamma_N^{-1}(k) &= 1 - \widetilde{C}_{N,k} X_N(k) \\ R_{N,k}^{-1} &= \lambda^{-1} R_{N,k-1}^{-1} \;-\; \widetilde{C}_{N,k}^H \gamma_N(k) \widetilde{C}_{N,k} \\ \epsilon_N^p(k) &= \epsilon_N(k|k-1) \;=\; d(k) + W_{N,k-1} X_N(k) \\ \epsilon_N(k) &= \epsilon_N(k|k) \;=\; \epsilon_N^p(k)\, \gamma_N(k) \\ W_{N,k} &= W_{N,k-1} + \epsilon_N(k) \widetilde{C}_{N,k} \end{aligned} \tag{5}$$

(6)

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 [2]) that they are related by the likelihood variable $\gamma_N(k)$. $\widetilde{C}_{N,k}$ is the Kalman gain of order N at time k.

3 The GSW RLS Algorithm

The Sliding Window Covariance RLS (SWCRLS) algorithm minimizes recursively the following criterion:

$$\xi_{N,L'}(k) = \sum_{i=k-L'+1}^{k} \lambda^{k-i} \left\| \epsilon_N(i|k) \right\|^2 \quad , \tag{7}$$

where L', the length of the sliding window must be greater than the filter length: $L' > N$. Compared to the WRLS algorithm, the SWC RLS algorithm exhibits a better tracking but is more sensible to noise. In order to reduce this sensibility, we add to the rectangular window, an attenuated exponential window. This introduce a new degree of freedom for the choice of the window which is the attenuation of the exponential tail. Furtermore, it has appeared that the derivation of a recursive algorithm which uses the new window must use an exponential window instead of the rectangular window and with the same forgetting factor as the attenuated exponential window. Hence, consider the following criterion:

$$\xi_{N,L'}(k) = (1-\alpha) \sum_{i=1}^{k-L'} \lambda^{k-i} \left\| \epsilon_N(i|k) \right\|^2 + \sum_{i=k-L'+1}^{k} \lambda^{k-i} \left\| \epsilon_N(i|k) \right\|^2 \tag{8}$$

The new criterion generalizes the WRLS and SWC RLS criteria since the WRLS crierion (2) is obtained from (8) by setting $\alpha = 0$ and the SWC RLS criterion is the one given by the generalized criterion when $\alpha = 1$ and $\lambda = 1$.
Putting the gradient of the quadratic criterion (8) equal to zero leads to the normal equations for the optimal filter $W_{L',k}$:

$$W_{L',k} = -P^H_{L',k} \, R^{-1}_{L',k} \quad , \tag{9}$$

where

$$\begin{aligned} R_{L',k} &= (1-\alpha) \sum_{i=1}^{k-L'} \lambda^{k-i} X_N(i) X_N^H(i) + \sum_{i=k-L'+1}^{k} \lambda^{k-i} X_N(i) X_N^H(i) \\ P_{L',k} &= (1-\alpha) \sum_{i=1}^{k-L'} \lambda^{k-i} X_N(i) d^H(i) + \sum_{i=k-L'}^{k} \lambda^{k-i} X_N(i) d^H(i) \end{aligned} \tag{10}$$

The use of the same forgetting factor for the two windows allows the following recursions for the sample second moments

$$\begin{aligned} R_{L'+1,k} &= \lambda R_{L',k-1} + X_N(k) X_N^H(k) \\ &= R_{L',k} - \alpha \lambda^{L'} X_N(k-L') X_N^H(k-L') \end{aligned} \tag{11}$$

$$\begin{aligned} P_{L'+1,k} &= \lambda P_{L',k-1} + X_N(k) d^H(k) \\ &= P_{L',k} - \alpha \lambda^{L'} X_N(k-L') d^H(k-L') \quad . \end{aligned} \tag{12}$$

Hence, the new algorithm is derived by applying the strategy for the usual RLS algorithm twice. The first step will be devoted to the time and order update $(k-1, L') \rightarrow (k, L'+1)$, which is analogous to the update of the usual RLS algorithm while the second step will be the order downdate $(k, L'+1) \rightarrow (k, L')$. The downdate scheme is obtained as follows:
By using (12), one has

$$W_{L',k} R_{L',k} = W_{L'+1,k} R_{L'+1,k} - \alpha \lambda^{L'} d_{k-L'} X_N^H(k-L') \quad . \tag{13}$$

Using (11) for $R_{L'+1,k}$ in term of $R_{L',k}$, we get

$$W_{L',k} = W_{L'+1,k} + \alpha\lambda^{L'}\zeta^{p}_{N,L'}(k)D_{N,L',k} \quad , \tag{14}$$

where $\zeta^{p}_{N,L'}(k) = d(k-L') + W_{L'+1,k}X_N(k-L')$ and $D_{N,L',k} = -X^H_N(k-L')R^{-1}_{L',k}$ are the a priori error signal and the a posteriori Kalman gain of the downdate part. Applying the MIL to (11) gives

$$R^{-1}_{L',k} = R^{-1}_{L'+1,k} + \widetilde{D}^H_{N,L',k}\delta_{N,L'}(k)\widetilde{D}_{N,L',k} \quad , \tag{15}$$

with $\widetilde{D}_{N,L',k} = -X^H_N(k-L')R^{-1}_{L'+1,k}$ and $\delta^{-1}_{N,L'}(k) = \alpha^{-1}\lambda^{-L'} - \widetilde{D}_{N,L',k}X_N(k-L')$ respectively the Kalman gain and the likelihood variable associated with the downdate part. Now, it is straigthtforward to find that $D_{N,L',k} = \alpha^{-1}\lambda^{-L'}\delta_{N,L'}(k)\widetilde{D}_{N,L',k}$ and that the a posteriori error is $\zeta_{N,L'}(k) = d(k-L') + W_{L',k}X_N(k-L') = \alpha^{-1}\lambda^{-L'}\delta_{N,L'}(k)\zeta^{p}_{N,L'}(k)$.
Finally, by associating the update part, the GSW RLS algorithm is given by

$$\begin{array}{rcl}
\widetilde{C}_{N,L',k} & = & -\lambda^{-1}X^H_N(k)R^{-1}_{L',k-1} \\
\gamma^{-1}_{N,L'}(k) & = & 1 - \widetilde{C}_{N,L',k}X_N(k) \\
\epsilon^{p}_{N,L'}(k) & = & d(k) + W_{L',k-1}X_N(k) \\
\epsilon_{N,L'}(k) & = & \epsilon^{p}_{N,L'}(k)\gamma_{N,L'}(k) \\
W_{L'+1,k} & = & W_{k-1,L'} + \widetilde{C}_{N,L',k}\epsilon_{N,L'}(k) \\
R^{-1}_{L+1,k} & = & \lambda^{-1}R^{-1}_{L',k-1} - \widetilde{C}^H_{N,L',k}\gamma_{N,L'}(k)\widetilde{C}_{N,L',k} \\
 & & \\
\widetilde{D}_{N,L',k} & = & -X^H_N(k-L')R^{-1}_{L'+1,k} \\
\delta^{-1}_{N,L'}(k) & = & \alpha^{-1}\lambda^{-L'} - \widetilde{D}_{N,L',k}X_N(k-L') \\
\zeta^{p}_{N,L'}(k) & = & d(k-L') + W_{L'+1,k}X_N(k-L') \\
\zeta_{N,L'}(k) & = & \alpha^{-1}\lambda^{-L'}\delta_{N,L',k}\zeta^{p}_{N,L'}(k) \\
W_{L',k} & = & W_{L'+1,k} + \alpha\lambda^{L'}\widetilde{D}_{N,L',k}\zeta_{N,L'}(k) \\
R^{-1}_{L,k'} & = & R^{-1}_{L'+1,k} + \widetilde{D}^H_{N,L',k}\delta_{N,L'}(k)\widetilde{D}_{N,L',k}
\end{array} \tag{16}$$

The set of equations (16) constitute the complete time update of the algorithm. The algorithm is initialized with $R_{L',0} = \mu I$ where μ is a small scalar quantity. The GSWRLS shows a computational complexity of $O(N^2)$. One must notice that the GSWRLS algorithm has the same structure and complexity as the SWCRLS algorithm.

4 Performance analysis

For the purpose of analysis, we consider the following classical identification model for the desired signal

$$d(k) = W^o_{N,k-1}X_N(k) + \eta(k) \tag{17}$$

where $\eta(k)$ is a centered Gaussian i.i.d. sequence with variance σ^2_η ($\eta(k) \sim \mathrm{N}(0,\sigma^2_\eta)$ and $W^o_{N,k}$ is the unknown filter that is time-varying according to a random walk

$$W^o_{N,k} = W^o_{N,k-1} + Z(k) \quad , \quad Z(k) \text{ i.i.d.} \sim \mathrm{N}(0,Q) \tag{18}$$

The normal equations can be rewritten as

$$W_{N,k} = -\left(\sum_{i=0}^{\infty}\omega_i d(k-i)X^H_N(k-i)\right)R^{-1}_{N,k} \tag{19}$$

with the sample covariance matrix

$$R_{N,k} = \sum_{i=0}^{\infty} \omega_i X_N(k-i) X_N^H(k-i) \quad , \tag{20}$$

and the ω_i are the coefficient of the window to be considered, for example, when the WRLS algorithm is used: $\omega_i = \lambda^i$.

The unknown filter can be written as follows

$$W_{N,k}^o R_{N,k} R_{N,k}^{-1} = \left(\sum_{i=0}^{\infty} \omega_i W_{N,k}^O X_N(k-i) X_N(k-i) \right) R_{N,k}^{-1} \tag{21}$$

hence, the deviation filter $\widetilde{W}_{N,k} = W_{N,k} + W_{N,k}^o$ can be expressed as:

$$\widetilde{W}_{N,k} = \left(\sum_{i=0}^{\infty} \omega_i (W_{N,k}^o X_N(k-i) - d(k-i)) X_N^H(k-i) \right) R_{N,k}^{-1} \tag{22}$$

using the random walk equation (18), we easily find that:

$$W_{N,k}^o X_N(k-i) - d(k-i) = \sum_{i=k-i}^{k} Z(k) X_N(k-i) - \eta(k-i) \tag{23}$$

finally, the deviation filter is given by

$$\widetilde{W}_{N,k} = \left(\sum_{i=0}^{\infty} \omega_i \left(\sum_{j=k-i}^{\infty} Z(j) \right) X_N(k-i) X_N(k-i)^H - \sum_{i=0}^{\infty} \omega_i \eta(k-i) X_N(k-i) \right) R_{N,k}^{-1} \tag{24}$$

let's $C_{N,k}$ be the covariance matrix of the deviation filter: $C_{N,k} = \mathrm{E}\left(\widetilde{W}_{N,k} \widetilde{W}_{N,k}^H\right)$ Using the Bayes rule $E\left(\widetilde{W}_k \widetilde{W}_k^H\right) = E_X E_{\eta,Z|X}\left(\widetilde{W}_k \widetilde{W}_k^H\right)$ and assuming that k is big enough so that: $R_{N,k} \approx E R_{N,k} = \left(\sum_{i=0}^{\infty} \omega_i \right) R$, we get after some manipulations

$$C_{N,k} = \sigma_\eta^2 \left(\sum_{i=0}^{\infty} \widetilde{\omega}_i^2 \right) R_{N,k}^{-1} + \left(\sum_{i=0}^{\infty} \varpi_i^2 \right) Q \quad , \tag{25}$$

where

$$\widetilde{\omega}_i = \omega_i \left(\sum_{j=0}^{\infty} \omega_j \right)^{-1} \tag{26}$$

and

$$\varpi_i = \left(\sum_{j=i}^{\infty} \omega_j \right) \left(\sum_{j=0}^{\infty} \omega_j \right)^{-1} = 1 - \sum_{j=0}^{i-1} \widetilde{\omega}_j \tag{27}$$

We see that (25) is divided into two parts: the first one is the *estimation noise*. the second one is the *lag noise* and comes only for a random echo path response, it shows the error that comes with the variation of the echo path response (Q), i.e. the tracking. Now, assuming the

statistical independence between $\widetilde{W}_{N,k}$ and $X_N(k-1)$ the variance of the a priori error signal is

$$\begin{aligned} \mathrm{var}(\epsilon_N^p(k)) &= \sigma_\eta^2 + tr\left(RC_{k-1}\right) \\ &= \sigma_\eta^2 + N\left(\sum_{j=0}^{\infty}\widetilde{\omega}_j^2\right)\sigma_\eta^2 + \left(\sum_{j=0}^{\infty}\varpi_j^2\right)\mathrm{tr}\left(RQ\right) \end{aligned} \tag{28}$$

Now we can compute the *noise amplification* in the 3 cases: RLS, SWC RLS and GSW RLS:

$$\begin{aligned} RLS &: \sigma_\eta^2\left(1+N\tfrac{1-\lambda}{1+\lambda}\right) \\ GSWRLS &: \sigma_\eta^2\left(1+N\tfrac{1-\lambda}{1+\lambda}\tfrac{1+\alpha(\alpha-2)\lambda^{2L'}}{(1-\alpha\lambda^{L'})^2}\right) \\ SWCRLS &: \sigma_\eta^2\left(1+\tfrac{N}{L'}\right) \end{aligned} \tag{29}$$

We recognize the RLS and the SWC RLS case as particular cases of the GSW RLS: SWC RLS for $\alpha = 1$ and $\lambda \to 1$, RLS for $\alpha = 0$.
It is normal that the SWRLS is an intermediate case between RLS and SWCRLS. The RLS is the best in term of noise amplification. the results for the tracking part are

$$\begin{aligned} RLS &: N\sigma_X^2\sigma_Z^2\tfrac{1}{1-\lambda^2} \\ GSWRLS &: N\sigma_X^2\sigma_Z^2\tfrac{1-2\alpha\lambda^{L'}(1+\lambda-\lambda^{L'+1})+\alpha^2\lambda^{2L'}(1+L'-L'\lambda^2)}{(1-\lambda^2)(1-\alpha\lambda^{L'})^2} \\ SWCRLS &: N\sigma_X^2\sigma_Z^2\tfrac{(L'+1)(2L'+1)}{6L'} \end{aligned} \tag{30}$$

This time the RLS gives, of course, the worst results: the noise is $\sim 1/(1-\lambda^2)$, with λ close to 1. On figure (1), we show curves made from the computation made before: first we fix the value of the estimation noise for SWC RLS ($\frac{N}{L'}$), then for this value, we vary λ and α in GSW RLS in order to keep the same noise estimation in GSW RLS while minimizing the lag noise in GSW RLS. This shows that for the same value of the noise amplification, GSW RLS has a lower lag noise than SWC RLS, moreover GSW RLS has the same complexity than SWC RLS, so we can conclude that the GSW RLS algorithm is truly better than the SWC RLS algorithm.

Figure 1: Comparison of SWCRLS and GSW RLS algorithms ($N = 50$).

5 The GSW FTF algorithm

The GSW FTF algorithm can be described in the following way, which emphasizes its rotational structure:

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

$$\begin{aligned}
e^p_{N,L'}(k) &= A_{N,L',k-1} X_{N_p}(k) \\
e_{N,L'+1}(k) &= e^p_{N,L'}(k)\gamma_{N,L'}(k-1) \\
\gamma_{N_p,L'}(k) &= \gamma_{N,L'}(k-1) - e^H_{N,L'_p}(k)\alpha^{-1}_{N,L'_p}(k)e_{N,L'_p}(k) \\
\gamma_{N,L'}(k) &= \left(1+\gamma_{N_p,L'}(k)\widetilde{C}^N_{N_p,L',k} r^p_{N,L'}(k)\right)^{-1}\gamma_{N_p,L'}(k) \\
r^p_N(k) &= -\lambda\beta_N(k-1)\widetilde{C}'^{NH}_{N_p1,k} \\
\alpha^{-1}_N(k) &= \lambda^{-1}\alpha^{-1}_N(k-1) - \widetilde{C}'^{0H}_{N_p1,k}\gamma^s_{N_p1}(k)\widetilde{C}^0_{N_p1,k} \\
r_{N,L'_p}(k) &= r^p_{N,L}(k)\,\gamma_{N,L'}(k) \\
\beta_{N,L'_p}(k) &= \lambda\beta_{N,L'}(k-1) + r^p_{N,L}(k)\, r^H_{N,L'+1}(k) \\
a_{N,L'+1}(k) &= A_{N,L'_p1,k} X_{N_p}(k-L'+1) \\
a^s_{N,L'}(k) &= a_{N,L'_p}(k)\delta_{N,L'_p}(k-1) \\
\alpha_{N,L'}(k) &= \alpha_{N,L'_p}(k) - \alpha\lambda^{L'} a^s_{N,L'}(k) a^{s^H}_{N,L'}(k) \\
\delta_{N+1,L'_p}(k) &= \delta_{N,L'_p}(k-1) - a^H_{N,L'_p}(k)\alpha^{-1}_{N,L'_p}(k)a_{N,L'_p}(k) \\
b_{N,L'_p}(k) &= -\beta_{N,L'_p}(k)\widetilde{D}^{NH}_{N_p,L_p,k} \\
\delta_{N,L'}(k) &= \left(1+\delta_{N_p,L'}(k)\widetilde{D}^N_{N_p,L',k} b^p_{N,L'}(k)\right)^{-1}\delta_{N_p,L'}(k) \\
b^s_{N,L'}(k) &= b_{N,L'_p}(k)\delta_{N,L'_p}(k) \\
\beta_{N,L'}(k) &= \beta_{N,L'_p}(k) - \alpha\lambda^{L'} b^s_{N,L'}(k) b^{s^H}_{N,L'}(k)
\end{aligned} \tag{31}$$

where $L'_p = L'+1$ and $N_p = N+1$. $A_{N,L',k}$ and $B_{N,L',k}$ are the forward and backward prediction filters that are used in the update part, $e^p_{N,L'}(k)$ and $e_{N,L'}(k)$ are the a priori and a posteriori forward prediction errors, $r^p_{N,L'}(k)$ and $r_{N,L'}(k)$ are the a priori and a posteriori backward predition errors, $\widetilde{C}_{N+1,k} = \left[\widetilde{C}^0_{N+1,k} \cdots \widetilde{C}^N_{N+1,k}\right]$ and $\alpha_{N,L'}(k)$ and $\beta_{N,L'}(k)$ are the forward and backward prediction error variances. Θ_k is a 5×5 rotation matrix given by

$$\Theta_k = \Theta^6_k\ \Theta^5_k\ \Theta^4_k\ \Theta^3_k\ \Theta^2_k\ \Theta^1_k \tag{32}$$

where the 5×5 matrices Θ^i_k $i = 1,2,3,4,5,6$ are

$$\Theta^6_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & a & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & b & 1 \end{bmatrix} \quad \Theta^5_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & c & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Theta_k^4=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & d & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & e & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \Theta_k^3=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ f & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ g & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Theta_k^2=\begin{bmatrix} 1 & 0 & h & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \Theta_k^1=\begin{bmatrix} 1 & i & 0 & 0 & 0 \\ j & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

with $a=-b^s_{N,L'}(k)$, $b=\alpha\lambda^{L'}\zeta_{N,L'}(k)$, $c=-\widetilde{D}^N_{N,L'+1,k}$, $d=-a^s_{N,L'+1}(k)$, $e=-a^H_{N,L'+1}(k)\alpha^{-1}_{N,L+1}(k)$, $f=r_{N,L'+1}(k)$, $g=\epsilon_{N,L+1}(k)$, $h=-\widetilde{C}^N_{N,L'+1,k}$, $i=-p_{N,L'}(k)\alpha^{-1}_{N,L'}(k)$ and $j=e_{N,L+1}(k)$.
In order to compute the rotation matrices, one must obtain the a priori errors $\epsilon^p_N(k)$, $r^p_N(k)$ and $\epsilon^p_N(k)$ which are the outputs at time k of the filters $A_{N,k-1}$, $B_{N,k-1}$ and $W_{N,k-1}$.

6 The GSW FTF-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

$$\widehat{d}^p_N(k)=d(k)-\epsilon^p_N(k) \quad , \quad \widehat{d}_N(k)=d(k)-\epsilon_N(k) \quad . \tag{34}$$

Consider now the following set of filtering operations

$$F_L(k) \triangleq \begin{bmatrix} \eta^H_{N,L,k} \\ e^{pH}_{N,L,k} \\ r^{pH}_{N,L,k} \\ \kappa^H_{N,L,k} \\ -\widehat{d}^{pH}_{N,L,k} \end{bmatrix} \triangleq \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-L}\right] \\ A_{N,k-L} \\ B_{N,k-L} \\ \left[0\ \widetilde{D}_{N,k-L}\right] \\ \left[W_{N,k-L}\ 0\right] \end{bmatrix} X^H_{N+1,L,k} \tag{35}$$

where

$$X_{N+1,L,k} = \left[X_{N+1}(k-L+1)\cdots X_{N+1}(k)\right]^H \tag{36}$$

is the $L\times(N+1)$ Toeplitz input data matrix. $F_L(k)$ is a $5\times L$ 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 five filters of the GSW FTF algorithm. $\eta_{N,L,k}$ and $\kappa_{N,L,k}$ are the output of the Kalman gains corresponding to the update and downdate parts, $e^p_{N,L,k}$ and $r^{pf}_{N,L,k}$ are respectively the vectors of forward and backward prediction errors

$$e^p_{N,L,k}=\begin{bmatrix} e^H_N(k-L+1|k-L) \\ \vdots \\ e^H_N(k|k-L) \end{bmatrix}, r^p_{N,L,k}=\begin{bmatrix} r^H_N(k-L+1|k-L) \\ \vdots \\ r^H_N(k|k-L) \end{bmatrix} \tag{37}$$

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

$$\widehat{d}^{\,p}_{N,L,k} = d_{L,k} - \epsilon^{p}_{N,L,k} = \begin{bmatrix} \widehat{d}^{\,H}_N(k{-}L{+}1|k{-}L) \\ \vdots \\ \widehat{d}^{\,H}_N(k|k{-}L) \end{bmatrix} . \tag{38}$$

The first column of $F_L\,(k)$ is

$$F_L\,(k)\;\; u_{L,1} \;\; = \;\; \begin{bmatrix} 1-\gamma^{-1}_{L_\omega}(k{-}L) \\ e^{p}_N\,(k{-}L{+}1) \\ r^{p}_N\,(k{-}L{+}1) \\ 1-\delta^{-1}_{L_\omega}(k{-}L) \\ -\widehat{d}^{\,p}_N\,(k{-}L{+}1) \end{bmatrix} \tag{39}$$

where $u_{L,n}$ is the $L \times 1$ vector with 1 at the n^{th} position and 0 elsewhere.
So with the quantities in $F_L\,(k)\;\; u_{L,1}$ and the recursions (31) and (**??**), it is possible to construct Θ_{k-L+1}. Now we rotate both expressions for $F_L(k)$ in (35) with Θ_{k-L+1} to obtain $\Theta_{k-L+1}F_L(k)$ which equals

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

$$\begin{bmatrix} \boxed{\eta^{H}_{N,L-1,k}} & * \\ e_N(k{-}L{+}1) & \boxed{e^{p\,H}_{N,L-1,k}} \\ r^{f}_N(k{-}L{+}1) & \boxed{r^{pf\,H}_{N,L-1,k}} \\ \boxed{\kappa^{H}_{N,L-1,k}} & * \\ -\widehat{d}_N\,(k{-}L{+}1) & \boxed{-\widehat{d}^{\,p\,H}_{N,L-1,k}} \end{bmatrix} . \tag{40}$$

Or we can write more compactly

$$\mathcal{S}\,(\Theta_{k-L+1}\,F_L(k)) \;\; = \;\; F_{L-1}(k) \tag{41}$$

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 (31) and (**??**). 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 (31).

7 The FSU GSW FTF 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 gains in (31), we need to work in the z-transform domain. So we shall associate polynomials with the filter coefficients as follows

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(z) \\ \widetilde{D}_k(z) \\ W_k(z) \end{bmatrix} = \begin{bmatrix} \left[\widetilde{C}_{N,k}\ 0\right] \\ A_{N,k} \\ B_{N,k} \\ \left[\widetilde{D}_{N,k}\ 0\right] \\ \left[W_{N,k}\ 0\right] \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N} \end{bmatrix} . \tag{42}$$

Hence (31) can be written in the z-transform domain as

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ \widetilde{D}_k(z) \\ W_k(z) \end{bmatrix} = \Theta_k \begin{bmatrix} z^{-1} & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & z^{-1} & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} \widetilde{C}_{k-1}(z) \\ A_{k-1}(z) \\ B_{k-1}(z) \\ \widetilde{D}_{k-1}(z) \\ W_{k-1}(z) \end{bmatrix} . \tag{43}$$

It appears natural to introduce

$$\Theta_k(z) = \Theta_k \begin{bmatrix} z^{-1} & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & z^{-1} & \\ & & & & 1 \end{bmatrix} . \tag{44}$$

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

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ \widetilde{D}_k(z) \\ W_k(z) \end{bmatrix} = \Theta_{k,L}(z) \begin{bmatrix} \widetilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ \widetilde{D}_{k-L}(z) \\ W_{k-L}(z) \end{bmatrix} \tag{45}$$

where

$$\Theta_{k,L}(z) = \Theta_k(z)\,\Theta_{k-1}(z)\cdots\Theta_{k-L+1}(z) \quad . \tag{46}$$

As mentioned before, the successive rotation matrices can be obtained via the GSW FTF-Schur algorithm with a computational complexity of $5L^2$ operations, which takes into account the fact that a rotation matrix in factored form as in (33) only contains five non-trivial entries.

Table I: the FSU GSW FTF Algorithm

#	Computation	Cost per L samples
1	$\begin{bmatrix} \eta^H_{N,L,k} \\ e^{p\,H}_{N,L,k} \\ r^{p\,H}_{N,L,k} \\ \kappa^H_{N,L,k} \\ -\widehat{d}^{\,p\,H}_{N,L,k} \end{bmatrix} = \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-L}\right] \\ A_{N,k-L} \\ B_{N,k-L} \\ \left[0\ \widetilde{D}_{N,k-L}\right] \\ \left[W_{N,k-L}\ 0\right] \end{bmatrix} X^H_{N+1,L,k}$	$(6+5\frac{N+1}{L})FFT(2L)+10N$
2	GSW FTF-Schur Algorithm: Input: $\eta_{N,L,k},\ e^p_{N,L,k},\ r^p_{N,L,k},\ \kappa_{N,L,k},\ -\widehat{d}^{\,p}_{N,L,k}$ Output: $\Theta_{k-i}(z)\ ,\ i=L-1,\cdots,0$	$5L^2$
3	$\Theta_{k,L}(z) = \prod_{i=0}^{L-1}\Theta_{k-i}(z)$	$20L^2$
4	$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(z) \\ \widetilde{D}_k(z) \\ W_k(z) \end{bmatrix} = \Theta_{k,L}(z) \begin{bmatrix} \widetilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ \widetilde{C}_{k-L}(z) \\ W_{k-L}(z) \end{bmatrix}$	$5(4+\frac{N+1}{L})FFT(2L)+40N$
Total cost per sample		$(26+10\frac{N+1}{L})\frac{FFT(2L)}{L}+50\frac{N}{L}+25L$

Now also remark that $\Theta_{k,L}(z)$ has the following structure

$$\Theta_{k,L}(z) = \begin{bmatrix} * & * & * & * & 0 \\ * & * & * & * & 0 \\ * & * & * & * & 0 \\ * & * & * & * & 1 \end{bmatrix} \tag{47}$$

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 $20L^2$ operations. As a result of the structure displayed in (47), the product in (45) represents 16 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 (35) 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 [16] for more details). The resulting FSU SFTF algorithm is summarized in Table I.

8 Concluding Remarks

The complexity of the FSU SFTF is $\mathcal{O}((10\frac{N+1}{L}+26)\frac{FFT(2L)}{L} + 50\frac{N}{L} + 25L)$ 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 $2.2N$ and $1.4N$ per sample. This should be compared to $14N$ for the GSW FTF 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.

References

[1] J. Benesty, P. Duhamel, "A Fast Exact LMS Adaptive Algorithm". *IEEE Trans. on ASSP*, ASSP-29(3):2904–2920, Dec. 1992.

[2] 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.

[3] J.M. Cioffi and T. Kailath, "Windowed Fast Transversal Filters Adaptive Algorithms with Normalization". *IEEE Trans. on ASSP*, ASSP-33(3):607–625, June 1985.

[4] E. Eleftheriou and D. Falconer, "Tracking properties and steady state performance of RLS adaptive filter algorithms". *IEEE Trans. on ASSP*, ASSP-34(5):821–823, July 1987.

[5] S. L. Gay, "A Fast Converging, Low Complexity Adaptive Filtering Algorithm". *int. rep. AT&T*, 1993.

[6] E. Hänsler, "The hands-free telephone problem. An annotated bibliography" *Signal Processing*, Vol. 27, pp. 259–271.

[7] S. Haykin, "Adaptive Filter Theory", Second edition, Prentice Hall, Englewood Cliffs, NJ, 1991.

[8] T. Kailath, *Linear Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[9] T. Kailath, S.Y. Kung, and M. Morf, "Displacement ranks of matrices and linear equations". *J. Math. Anal. Appl.*, 68(2):295–407, 1979. (See also *Bull. Amer. Math. Soc.*, vol. 1, pp. 769–773, 1979.).

[10] K. Maouche and D. T. M. Slock, "The Fast Subsampled-Updating Fast Newton Transversal filter (FSU FNTF) Algorithm for Adaptive Filtering Based on on a Schur procedure and the FFT". Research Report N° 94-014, Institut Eurécom.

[11] M. Montazeri and P. Duhamel, "Fast Modified Projection Algorithms for Acoustic Echo Cancellation ". Submitted to *IEEE Trans. on ASSP*, 1994.

[12] G.V. Moustakides and S. Theodoridis, "Fast newton transversal filter –A new class of adaptive estimation algorithms". *IEEE Trans. on ASSP*, ASSP-39(10):2184–2193, Oct. 1991.

[13] C. B. Papadias and D. T. M. Slock, "New Adaptive Blind Equalization Algorithms for Constant Modulus Constellations". *International Conference on Acoustics, Speech and Signal Processing (ICASSP-94)*, pp. 321-324, Adelaide, Australia, April 19-22, 1994.

[14] T. Pétillon, A. Gilloire and S. Thedoridis, "The Fast Newton Transversal filter: An Efficient Scheme for acoustic Echo Cancellation in Mobile Radio ". *IEEE Trans. on ASSP*, ASSP-42(3):2184–2193, March 1994.

[15] 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.

[16] 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*, Vol. 40, No. 1, Oct. 1994, pp. 5–20.

[17] D.T.M. Slock and K. Maouche, "The Fast Subsampled-Updating Stabilized Fast Transversal Filter (FSU SFTF) RLS Algorithm". In *Proc. EUSIPCO 94, VIIth European Signal Processing Conference,* pages 740-743, Edinburgh, Scotland, U.K. Sep. 13-16 1992.

[18] M. Vetterli, "Fast Algorithms for Signal Processing". In M. Kunt, editor, *Techniques modernes de traitement numérique des signaux*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1991. ISBN 2-88074-207-2.

[19] B. Widrow et al., "Stationary and nonstationary learning characteristics of the LMS adaptive filter", *Proc. IEEE*, vol. 64, No. 8, August 1976, pp. 1151–1162.