The Fast Subsampled-Updating Fast Newton Transversal Filter (FSU FNTF) for Adapting Long FIR Filters

Karim Maouche and Dirk T. M. Slock
Institut EURECOM, 2229 route des Crêtes,
B.P. 193, 06904, Sophia Antipolis Cedex, FRANCE

Abstract

The FNTF algorithm starts from the RLS algorithm for adapting FIR filters. The FNTF algorithm approximates the Kalman gain by replacing the sample covariance matrix inverse by a banded matrix (AR(M) assumption for the input signal). The approximate Kalman gain can still be computed using an exact recursion that involves the prediction parts of two Fast Transversal Filter (FTF) algorithms of order M. Here we introduce the Subsampled Updating (SU) approach in which the FNTF filter estimate and Kalman gain are provided at a subsampled rate, say every L samples. The low-complexity prediction part is kept and a Schur type algorithm is used to compute a priori filtering errors at the intermediate time instants, while some convolutions are carried out with the FFT. This leads to the FSU FNTF algorithm which has a low computational complexity for relatively long filters.

1 Introduction

We present a new fast algorithm for Recursive Least Squares (RLS) adaptive filtering. The Fast Subsampled Updating Fast Newton Transversal Filter (FSU FNTF) algorithm is derived from the Fast Newton Transversal Filter (FNTF) algorithm. The FNTF algorithm departs from the Fast Transversal filter (FTF) algorithm which efficiently exploits the displacement structure of the inverse of the sample covariance matrix appearing in the RLS algorithm to reduce its computational complexity from $\mathcal{O}(N^2)$ for the RLS algorithm (N is the length of the adaptive filter) to $7N$ in the most efficient version [1]. The FNTF algorithm uses the approximation that when dealing with Auto-Regressive signals, the prediction part of the FTF algorithm can be limited to prediction filters and Kalman gain of length M, the order of the AR model [3]. In fact, in the FNTF algorithm the inverse of the sample covariance matrix is approximated by a banded matrix of total bandwidth $2M+1$. This fact allows the reduction of the complexity to $\mathcal{O}(2N)$. However, this first version suffers from the need for significant data storage. In [4], this problem was overcome by using two prediction part FTF algorithms. In [5],[6], 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 and FSU SFTF algorithms which present a reduced complexity when dealing with long filters. Here, we apply the subsampled updating strategy to the FNTF algorithm. In this approach, we keep the prediction part of the FNTF algorithm and compute the FNTF Kalman gain and the filter estimate from quantities that were available L instants before. It turns out that the successive a priori filtering errors can be computed efficiently by using a Schur type algorithm while some convolutions are done with the Fast Fourier Transform (FFT) technique. This leads to a new algorithm with a reduced computational complexity, rendering it especially suited for adapting very long filters such as in the acoustic echo cancellation problem.

2 The FSU FNTF Algorithm

2.1 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

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

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 Hermi-

tian (complex conjugate) transpose. In the RLS algorithm, the set of N transversal filter coefficients $W_{N,k} = \left[W_{N,k}^0 \cdots W_{N,k}^{N-1}\right]$ are adapted so as to minimize recursively the following LS criterion

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

where $\lambda \in (0,1]$ is the exponential forgetting factor. The RLS algorithm is given by

$$\begin{aligned}\widetilde{C}_{N,k} &= -X_N^H(k)\lambda^{-1}R_{N,k-1}^{-1} \qquad (3)\\ \gamma_N^{-1}(k) &= 1-\widetilde{C}_{N,k}\,X_N(k) \qquad (4)\\ R_{N,k}^{-1} &= \lambda^{-1}R_{N,k-1}^{-1} - \widetilde{C}_{N,k}^H\gamma_N(k)\widetilde{C}_{N,k} \qquad (5)\\ \epsilon_N^p(k) &= \epsilon_N(k|k-1) = d(k)+W_{N,k-1}\,X_N(k) \qquad (6)\\ \epsilon_N(k) &= \epsilon_N(k|k) = \epsilon_N^p(k)\,\gamma_N(k) \qquad (7)\\ W_{N,k} &= W_{N,k-1}+\epsilon_N(k)\widetilde{C}_{N,k}\ , \qquad (8)\end{aligned}$$

where $\epsilon_N^p(k)$ and $\epsilon_N(k)$ are the a priori and a posteriori filtering errors, which are related by the likelihood variable $\gamma_N(k)$ (7) and $\widetilde{C}_{N,k}$ is the Kalman gain. The updating of the $(N\times N)$ inverse covariance matrix $R_{N,k}^{-1}$ is done via the Ricatti equation (5) in $\mathcal{O}(N^2)$ operations. Eqs. (3),(4) and (5) constitute the prediction part whereas the set of eqs. (6),(7) and (8) are called the joint-process.

Fast versions of the RLS algorithm have been derived by using the displacement structure of the covariance matrix and leads to algorithms such as the FTF which computational complexity is $7N$.

2.2 The FNTF Algorithm

In the FTF algorithm, the Kalman gain and the likelihood variable are computed in the prediction part of the algorithm where the update of the sample inverse covariance is replaced by the update of its generators which are the optimal forward and backward prediction filters and the Kalman gain of order N (plus the update of other scalar quantities). The FNTF algorithm is an approximation to the FTF algorithm. It uses the fact that for Auto-Regressive signals of order M, the inverse covariance matrix is a banded matrix of bandwith $2M+1$, This fact allows to use prediction filters of order M in the FTF scheme and Kalman gain and the likelihood variable of order N are computed by using the property that for AR(M) processes, forward and backward prediction filters of order N are obtained from those of order M by filling with zeros. This is interesting when the input signal is speech as is the case for acoustic echo cancellation applications. The key ingredient of the FNTF algorithm is the extrapolatation of the Kalman gain $\widetilde{C}_{N,M,k}$ and the likelihood variable $\gamma_{N,M}(k)$ of order N from quantities computed in the prediction part of order M. The first version of the FNTF algorithm [3] sufferred from the need for a significant data storage. This drawback was overcome by using two FTF prediction parts, the second one being the delayed version of the first one with a delay of $N-M$ samples [4] . In what follows, we will consider this last version which uses two FTF prediction parts of order M running in parallel. The FNTF algorithm can be described in the following way which emphasizes its rotational structure:

$$\begin{aligned}\widetilde{P}_k &= \Phi^p(k)\,P_{k-1}\\ k_d &= k-N+M\\ \widetilde{P}_{k_d} &= \Phi^p(k_d)\,P_{k_d-1}\\ \begin{bmatrix}\left[\widetilde{C}_{N,M,k}\ 0\right]\\ \left[W_{N,k}\ 0\right]\end{bmatrix} &= \Phi(k)\begin{bmatrix}\left[P_{k-1}\ 0_{N-M}\right]\\ \left[0_{N-M}\ P_{k_d-1}\right]\\ \left[0\ \widetilde{C}_{N,M,k-1}\right]\\ \left[W_{N,k-1}\ 0\right]\end{bmatrix}\\ i=k,k_d:\ e_M^p(i) &= A_{M,i-1}X_{M+1}(i)\\ e_M(i) &= e_M^p(i)\gamma_M(i-1)\\ \gamma_{M+1}^{-s}(i) &= \gamma_M^{-1}(i-1)-\widetilde{C}_{M+1,i}^0 e_M^p(i)\\ \gamma_M^{-s}(i) &= \gamma_{M+1}^{-s}(i)+\widetilde{C}_{M+1,i}^M r_M^{pf}(i)\\ r_M^{ps}(i) &= -\lambda\beta_M(i-1)\widetilde{C}_{M+1,i}^{M\,H}\\ r_M^{pf}(i) &= B_{M,i-1}X_{M+1}(i)\\ \alpha_M^{-1}(i) &= \lambda^{-1}\alpha_M^{-1}(i-1)-\widetilde{C}_{M+1,i}^{0\,H}\,\gamma_{M+1}^{s}(i)\,\widetilde{C}_{M+1,i}^0 \qquad (9)\\ r_M^{p(j)}(i) &= K_j r_M^{pf}(i)+(1-K_j)r_M^{ps}(i)\\ r_M^{(j)}(i) &= r_M^{p(j)}(i)\gamma_M^s(i) \qquad j=1,2\\ \beta_M(i) &= \lambda\beta_M(i-1)+r_M^{(2)}(i)\,r_M^{p(2)^H}(i)\\ \gamma_M(i) &= \lambda^M\beta_M(i)\,\alpha_M^{-1}(i)\\ \widetilde{S}_{M+1,i} &= e_M^p(i)\lambda^{-1}\alpha_M^{-1}(i-1)A_{M,i-1}\\ \widetilde{U}_{M+1,i} &= r_M^{pf}(i)\lambda^{-1}\beta_M^{-1}(i-1)B_{M,i-1}\\ \gamma_{N,M}(k) &= \gamma_{N,M}(k-1)+\widetilde{S}_{M+1,k}^0 e_M^p(k)-\widetilde{U}_{M+1,k_d}^M r_M^{pf}(k_d)\\ \epsilon_N(k) &= \gamma_{N,M}(k)\epsilon_N^p(k)\end{aligned}$$

where

$$\widetilde{P}_k = \begin{bmatrix}\left[\widetilde{C}_{M,k}\ 0\right]\\ A_{M,k}\\ B_{M,k}\end{bmatrix}\ ,\ P_k = \begin{bmatrix}\left[0\ \widetilde{C}_{M,k}\right]\\ A_{M,k}\\ B_{M,k}\end{bmatrix} \qquad (10)$$

$$\begin{aligned}\Phi^p(k) &= \Phi_2^p(k)\,\Phi_1^p(k)\\ &= \begin{bmatrix}1&0&0\\0&1&0\\ r_M^1(k)&0&1\end{bmatrix}\begin{bmatrix}1&a&-\widetilde{C}_{M+1,k}^M\\ e_M(k)&1&0\\0&0&1\end{bmatrix}\\ a &= -e_M^p(k)\lambda^{-1}\alpha_M^{-1}(k) \qquad (11)\end{aligned}$$

$$\Phi(k)=\Phi_2(k)\,\Phi_1(k)=$$

$$\begin{bmatrix}1&0\\ \epsilon_N(k)&1\end{bmatrix}\begin{bmatrix}0&-\widetilde{C}_{M+1,k}^0&0&0&0&b&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}$$

$$b = r_M^{pf}(k_d)\lambda^{-1}\,\beta_M(k_d-1)\ . \qquad (12)$$

$A_{M,k}$ and $B_{M,k}$ are the forward and backward prediction filters, $e_M^p(k)$ and $e_M(k)$ are the a priori and a

posteriori forward prediction errors, $r_M^p(k)$ and $r_M(k)$ are the a priori and a posteriori backward predition errors, $\widetilde{C}_{M+1,k} = \left[\widetilde{C}^0_{M+1,k} \cdots \widetilde{C}^M_{M+1,k}\right]$ and $\alpha_M(k)$ and $\beta_M(k)$ are the forward and backward prediction error variances. $K_1 = 1.5$ and $K_2 = 2.5$ are the optimal feedback gains that ensure the stability of the dynamics of the accumulated round-off errors in the prediction part [2].
The prediction part of the FNTF has a computational complexity of $12M$ and the joint-process takes $2N$ operations. In order to reduce the amount of computations, we use a subsampled updating strategy. This idea comes from the fact that when one deals with relatively long adaptive filters, it is not necessary to update the filter at each new input sample because there is not a significant change in the filter coefficients after one update. Hence, the subsampled updating strategy combined with fast convolution techniques can help to reduce the complexity. In the FNTF algorithm, the prediction part is not computationally demanding. Hence, we will apply the subsampled updating just for the filtering part of the FNTF and the prediction part remains unchanged. Henceforth, the objective is to compute at time k the extrapolated Kalman gain $\widetilde{C}_{N,M,k}$ of order N and the filter $W_{N,k}$ from their values at time $k-L$. First, a Schur procedure can be used to compute the different filter outputs appearing in the FNTF algorithm, and in particular, the L successive a priori filtering errors without updating the adaptive filters at these instants.

2.3 The Schur-FNTF algorithm

Let us introduce the negative of the filter output

$$\widehat{d}_N^p(k) = d(k) - \epsilon_N^p(k) \;\;,\;\; \widehat{d}_N(k) = d(k) - \epsilon_N(k) \;, \qquad (13)$$

and consider the following set of filtering operations

$$F_L(k) \triangleq \begin{bmatrix} g_L^H(k) \\ g_L^H(k_d) \\ \eta_{N,L,k}^H \\ -\widehat{d}_{N,L,k}^{\,p\,H} \end{bmatrix} \triangleq \begin{bmatrix} [P_{k-L}\ 0_{N-M}] \\ [0_{N-M} P_{k_d-L}] \\ [0\ \widetilde{C}_{N,M,k-L}] \\ [W_{N,k-L}\ 0] \end{bmatrix} X^H_{N+1,L,k}$$

$$g_L^H(k) \triangleq \begin{bmatrix} \eta_{M,L,k}^H \\ e_{M,L,k}^{p\,H} \\ r_{M,L,k}^{pf\,H} \end{bmatrix} . \qquad (14)$$

where

$$X_{N+1,L,k} = [X_{N+1}(k-L+1) \cdots X_{N+1}(k)]^H \;, \qquad (15)$$

is the $(L \times (N+1))$ Toeplitz input data matrix. $F_L(k)$ is a $(8 \times L)$ matrix whose rows are the output of the different filters appearing in the FNTF algorithm. In particular the last row of $F_L(k)$ corresponds to the (multi-step ahead predicted) adaptive filter outputs

$$\widehat{d}^{\,p}_{N,L,k} = d_{L,k} - \epsilon^p_{N,L,k} = \begin{bmatrix} d^H(k-L+1) \\ \vdots \\ d^H(k) \end{bmatrix} - \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-L) \end{bmatrix} = \begin{bmatrix} \widehat{d}_N^H(k-L+1|k-L) \\ \vdots \\ \widehat{d}_N^H(k|k-L) \end{bmatrix} . \qquad (16)$$

The first column of $F_L(k)$ is

$$F_L(k) u_{L,1} = \left[\, p^T(k)\ \ p^T(k_d)\ \ 1-\gamma_N^{-1}(k-L)\ \ -\widehat{d}_N^{\,p}(k-L+1) \,\right]^T$$
$$p^T(k) = \left[\, 1-\gamma_M^{-1}(k-L)\ \ e_M^p(k-L+1)\ \ r_M^{pf}(k-L+1) \right] . \qquad (17)$$

where $u_{L,n}$ is the $L \times 1$ vector with 1 at the n^{th} position and 0 elsewhere.
The updating scheme of the FNTF algorithm can be written as

$$\begin{bmatrix} [\widetilde{P}_k\ 0_{N-M}] \\ [0_{N-M}\ \widetilde{P}_{k_d}] \\ [\widetilde{C}_{N,M,k}\ 0] \\ [W_{N,k}\ 0] \end{bmatrix} = \Theta_k \begin{bmatrix} [P_{k-1}\ 0_{N-M}] \\ [0_{N-M}\ P_{k_d-1}] \\ [0\ \widetilde{C}_{N,M,k-1}] \\ [W_{N,k-1}\ 0] \end{bmatrix} , \qquad (18)$$

where Θ_k is a (8×8) matrix given by

$$\Theta_k = \begin{bmatrix} I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & \Phi_2(k) \end{bmatrix} \begin{bmatrix} \Phi_2^p(k) & 0 & 0 \\ 0 & \Phi_2^p(k_d) & 0 \\ 0 & 0 & I_2 \end{bmatrix} \begin{bmatrix} \Phi_1^p(k) & & & 0 & 0 \\ 0 & & & \Phi_1^p(k_d) & 0 \\ & & \Phi_1(k) & & I_2 \end{bmatrix} . \qquad (19)$$

Hence, if we rotate both expressions for $F_L(k)$ in (14) with Θ_{k-L+1}, we obtain $\Theta_{k-L+1} F_L(k)$ which equals

$$\begin{bmatrix} [\widetilde{P}_{k-L+1}\ 0_{N-M}] \\ [0_{N-M}\ \widetilde{P}_{k_d-L+1}] \\ [\widetilde{C}_{N,M,k-L+1}\ 0] \\ [W_{N,k-L+1}\ 0] \end{bmatrix} X^H_{N+1,L,k} = \begin{bmatrix} q_L(k) \\ q_L(k_d) \\ \boxed{\eta^H_{N,L-1,k}} \quad * \\ -\widehat{d}_N(k-L+1) \quad \boxed{-\widehat{d}^{\,p\,H}_{N,L-1,k}} \end{bmatrix} \qquad (20)$$

$$\text{where } q_L(k) = \begin{bmatrix} \boxed{\eta^H_{M,L-1,k}} & * \\ e_M(k-L+1) & \boxed{e^{p\,H}_{M,L-1,k}} \\ r^f_M(k-L+1) & \boxed{r^{pf\,H}_{M,L-1,k}} \end{bmatrix} . \qquad (21)$$

We can see from the above that the transformation of $F_L(k)$ by the application of the matrix Θ_{k-L+1} provides quantities (in boxes) that are the different rows of $F_{L-1}(k)$. This can be written more compactly as

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

where the operator $\mathcal{S}(M)$ stands for: shift the first, the fourth and the seventh rows 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 application instance, the different parameters of the next transformation matrix can be calculated from the first column of $F_l(k)$. In particular, the successive a priori error filtering are hence computed over the block data without updating the adaptive filter. Now, since it is possible to compute the parameters of the successive matrices Θ_k, it suffices to accumulate them and apply the resulting matrix to the filters in order to update them. In fact, it turns out that because of the shift operation of the Kalman gain, the accumulated matrix is a polynomial matrix and hence, the updating of the filters is done via convolutions. This technique has already been applied to the FTF algorithm leading to what we have called the Schur-FTF algorithm [6]. This procedure was the key ingredient for the derivation of the FSU SFTF algorithm that presents a reduced computational complexity when adapting long FIR filters. In the present case where the complexity of the prediction part is small ($12\ M$), the Schur-FNTF procedure will be only used for the computation of the successive a priori error filtering. Thus, the two prediction part FTF algorithms of order M are kept. A remarkable property of the Schur-FNTF procedure is the removal of the long-term round-off error instability due to the recursive computation of the likelihood variable $\gamma_{N,M}(k)$. The recursions are interrupted since the likelihood variable is computed at each new block of data via an inner product (17). This fact has a big importance for the real time implementation of the algorithm.

Taking into account that Θ_k in its factorized form (19) has 11 non-trivial elements, the FNTF-Schur procedure takes only $5.5L$ multiplications per sample. Inner products (filtering operations) are only needed for the initialization (computation of $F_L(k)$). This is the FNTF-Schur algorithm, which contrasts with the Levinson-style FNTF algorithm in (9).

2.4 The FSU FNTF algorithm

Having computed the successive a priori error filtering, the issue now is the computation of the estimated filter at time k from its value L instants before.

One finds straightforwardly

$$\left[W_{N,k}\ 0\right] = \left[W_{N,k-L}\ 0\right] + \underline{\epsilon}^{H}_{N,L,k}\ \left[\underline{\widetilde{C}}_{N,M,k}\ 0\right]\ , \qquad (23)$$

where

$$\begin{aligned} \underline{\epsilon}^{H}_{N,L,k} &= [\epsilon_N(k-L+1)\cdots\epsilon_N(k)] \\ \underline{\widetilde{C}}_{N,M,k} &= \begin{bmatrix} \widetilde{C}_{N,M,k-L+1} \\ \vdots \\ \widetilde{C}_{N,M,k} \end{bmatrix}\ . \end{aligned} \qquad (24)$$

As was shown in the previous section, $\underline{\epsilon}^{H}_{N,L,k}$ is computed with the Schur-FNTF algorithm. On the other hand, the extrapolated Kalman gain is given by

$$\begin{aligned} \left[\ \underline{\widetilde{C}}_{N,M,k}\ 0\ \right] &= \left[\ \underline{\widetilde{C}}_{N,M,k-1}\ 0\ \right] - \left[\ \widetilde{S}_{M+1,k}\ 0_{N-M}\ \right] \\ &+ \left[\ 0_{N-M}\ \widetilde{U}_{M+1,k_d}\ \right]\ . \end{aligned} \qquad (25)$$

From the above equation, it is easy to see that for $i = 1, \ldots, L$:

$$\begin{aligned} \left[\widetilde{C}_{N,M,k-L+i}\ \ 0\right] &= \left[0_i\ \ \widetilde{C}^{0:N-i}_{N,M,k-L}\right] \\ &- \sum_{l=0}^{i}\left[\widetilde{S}_{M+1,k_l}\ 0_{N-M}\right]Z^l + \sum_{l=0}^{i}\left[0_{N-M}\ \widetilde{U}_{M+1,k_l}\ \right]Z^l\ , \end{aligned} \qquad (26)$$

where Z is the lower $(M \times M)$ shift matrix (ones on the first subdiagonal and zeros elsewhere) and $k_l = k-L+i-l$. In particular, we see from (26) that the Kalman gain $\left[\widetilde{C}_{N,M,k}\ 0\right]$ can be obtained from $\left[0_L\ \widetilde{C}^{0:N-L}_{N,M,k-L}\right]$ and the $\widetilde{S}_{M+1,k-l}, \widetilde{U}_{M+1,k-l}$, for $l = 0, \ldots, L$. Rewriting (26) for every row gives the Kalman gain matrix $\underline{\widetilde{C}}_{N,M,k}$ in term of the Kalman gain vector $\widetilde{C}_{N,M,k-L}$

$$\begin{aligned} \left[\underline{\widetilde{C}}_{N,M,k}\ 0\right] &= \left[\left[\widetilde{C}_{N,M,k-L}\ 0\right]Z^i\right]_{i=1:L} \\ &- \left[\sum_{l=0}^{i}\left[\widetilde{S}_{M+1,k_l}\ 0_{N-L}\right]Z^l\right]_{i=1:L} \\ &+ \left[\sum_{l=0}^{i}\left[0_{N-M}\ \widetilde{U}_{M+1,k_l}\right]Z^l\right]_{i=1:L}\ , \end{aligned} \qquad (27)$$

where the notation $[a_i]_{i=1:L}$ stands for the matrix in which the i^{th} row is a_i. The first term of the right hand side of (27) is a $(L \times (N+1))$ Toeplitz matrix and the two others are $(L \times (N+1))$ sparse matrices. Hence the product in (23) decomposes in three parts: the first one can be done efficiently by using fast convolution techniques. When the filter to adapt is relatively large, the convolution is implemented with the FFT Technique (Overlap-Save method). This is achieved in $\left(2\frac{N+1}{L}+1\right)\frac{FFT(2L)}{L} + 2\frac{N+1}{L}$ multiplications per sample, where $FFT(m)$ is the computational complexity of the Fourier transform of a sequence of length m. The product of $\underline{\epsilon}_{N,L,k}$ with the second and third matrix of the right hand side of (27) is done straighforwardly with $L + 3M$ multiplications per sample.

Table : FSU FNTF Algorithm

#	Computations	Cost per L samples
1	$\begin{bmatrix} g_L^H(k) \\ g_L^H(k_d) \\ \eta^{p\,H}_{N,L,k} \\ -\widehat{d}^{\,p\,H}_{N,L,k} \end{bmatrix} = \begin{bmatrix} [P_{k-L}\ 0_{N-M}] \\ [0_{N-M}\, P_{k_d-L}] \\ \begin{bmatrix} 0 & \widetilde{C}_{N,M,k-L} \end{bmatrix} \\ [W_{N,k-L}\ 0] \end{bmatrix} X^H_{N+1,L,k}$	$(3+2\frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 4\mathrm{(N+1)}$
2	Schur-FNTF Algorithm : Input: $g_L(k),\ g_L(k_d),\ \eta^p_{N,L,k},\ -\widehat{d}^{\,p}_{N,L,k}$ Output: $\Theta_{k-L+i}\ i=1,\cdots,L\ ,\ \underline{\epsilon}_{N,L,k}$	$5.5\mathrm{L}^2$
3	Computation of: $\begin{bmatrix} \underline{\widetilde{C}}_{N,M,k} & 0 \end{bmatrix}$, see (27)	$2\mathrm{(M+1)L}$
4	$[W_{N,k}\ 0] = [W_{N,k-L}\ 0] + \underline{\epsilon}^H_{N,L,k} \begin{bmatrix} \underline{\widetilde{C}}_{N,M,k} & 0 \end{bmatrix}$	$(1+2\frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 2\mathrm{(N+1)} + .5\mathrm{L}^2 + 2\mathrm{ML}$
Total cost per sample		$4(1+\frac{\mathrm{N+1}}{\mathrm{L}})\frac{\mathrm{FFT(2L)}}{\mathrm{L}} + 6\frac{\mathrm{N+1}}{\mathrm{L}} + 6\mathrm{L} + 16\mathrm{M}$

The computation of the two sums in(27) is done with $2(M+1)L$ additions. The Overlap-Save method is also used to compute the product (14) which is needed for the initialization of the Schur procedure. The resulting FSU FNTF algorithm is summarized in Table I.

3 Conclusions

The computational complexity of the FSU FNTF algorithm is $\mathcal{O}(4\frac{N+1}{L}\frac{FFT(2L)}{L} + 6\frac{N}{L} + 6L + 16M)$ operations per sample. This can be very interesting for long filters. For example, when $(N, L, M) = (4095, 256, 16); (8191, 256, 16)$ and the FFT is done via the split radix ($FFT(2m) = mlog_2(2m)$ real multiplications for real signals) the multiplicative complexity is respectively $0.6N$ and $0.4N$, compared to $7N$ for the FTF algorithm, the currently fastest RLS algorithm, and $2N$ for the FNTF algorithm. The number of additions is somewhat higher. The cost we pay is a processing delay which is of the order of L samples. The low computational complexity of the FSU FNTF when dealing with long filters and its performance capabilities render it very interesting for applications such as acoustic echo cancellation.

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] 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, October 1991.

[4] 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.

[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.

[6] Dirk 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.