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

Research Report N° 94-014

The Fast Subsampled-Updating Fast Newton Transversal Filter (FSU FNTF) Algorithm for Adaptive Filtering Based on a Schur Procedure and the FFT

Karim Maouche Dirk T.M. Slock

November, 1994

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 Fast Newton Transversal Filter (FNTF) algorithm starts from the Recursive Least-Squares algorithm for adapting an FIR filter of length N. The FNTF algorithm approximates the Kalman gain by replacing the sample covariance matrix inverse by a banded matrix of total bandwidth 2M+1 (AR(M) assumption for the input signal). In this algorithm, 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. 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. Because of its low computational complexity, the prediction part of the FNTF algorithm is kept. A Schur type algorithm is used to compute various filter outputs at the intermediate time instants, while some products of vectors with Toeplitz matrices are carried out with the FFT. This leads to the Fast Subsampled-Updating FNTF (FSU FNTF) algorithm which has a relatively low computational complexity for large N while presenting good convergence and tracking performances. This renders the FSU FNTF algorithm very interesting for applications such as acoustic echo cancellation.

Contents

1 Introduction

Hands-free communications have become popular these last years. In these new systems, standard telephone is replaced by an audio system with microphone(s) and loudspeaker(s). This allows hands-free operations that can be critical for safety reasons in applications such as mobile telephone or leads to convenient and ergonomic communications in conferencing applications such as audioconference and teleconference. Unfortunately, these new systems suffer from a major drawback which is the acoustic echo phenomenon (see [4] for bibliography). The acoustic echo is a disturbing signal which comes from the sound propagation between the loudspeaker and the microphone of one audio terminal. In a communication situation as shown in Fig.1, the speech signal which comes from the distant speaker is reinjected into the microphone. This forms the acoustic echo which is fed back to the distant user. In the case where the round trip delay is larger than 20-30 milliseconds, the acoustic echo becomes perceivable for the distant speaker and hence disturbs the communication. Moreover, when such an open acoustic system is installed on both ends of the telephone connection, then a closed loop exists for the sounds which can lead to instability of the closed loop (Larsen effect). A simple solution for acoustic echo control is the switching of gains or the insertion of losses in the audio terminal but this kind of solutions does not yield to satisfactory performances. Actual solutions are based on the real-time identification of the acoustic impulse response. Real-time processing is necessary since the acoustic channel is time-varying. The principle (see Fig.2) is to synthetize an estimate $\widehat{y}_k$ of the real echo y_k by using an adaptive filter W_k (which modelize the acoustic path) whose input is the loudspeaker signal x_k. The echo estimate is substracted from the microphone signal which is the sum of the acoustic echo y_k and the output noise system n_k. This substraction gives the filtering error ϵ_k which is sent to the far speaker and which controls the adaptation mechanism of W_k. In the situation of acoustic echo cancellation, adaptive filtering becomes a very difficult task because of the hostility of the acoustic environment. This is essentially due to three characteristics of the problem:

- The acoustic impulse response has a relatively long duration. This translates into thousands of samples for sampling frequencies that lead to satisfactory quality audio. The number of degrees of freedom is close to the number of samples of the acoustic impulse response.
- The echo path can quickly varies in the case of movement of people or object in the room. It varies also with ambient temperature, pressure and humidity.
- The input and disturbance signals in the system identification are speech signals. They are very correlated and nonstationary. Moreover, silent periods of the input signal renders the problem ill-conditionned.

There are two major classes of adaptive algorithms. One is the Least-Mean-Square (LMS) algorithm which is based on a stochastic-gradient method [14]. The LMS algorithm has a computational complexity of $2N$ (N is the filter order). This renders the LMS algorithm very popular. Nevertheless, its performances becomes worst when the adaptive filter is relatively large or when the input signal is correlated as is the case in acoustic echo cancellation. The other class of adaptive algorithm is the Recursive Least-Squares (RLS) algorithm which minimizes a deterministic sum of squared errors [5]. RLS Algorithm shows a complexity of $O(N^2)$. However, for Finite Impulse Response (FIR) filtering, consecutive regression vectors are re-

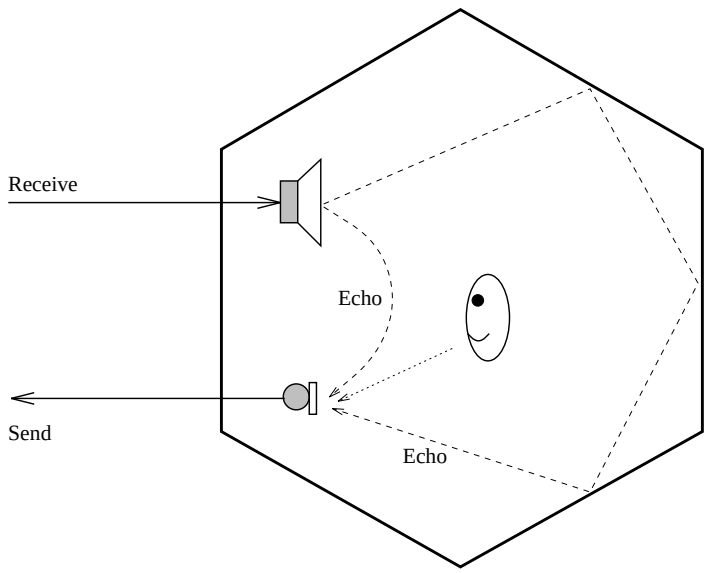

Figure 1: Acoustic echo phenomenon in hands-free communications.

lated through a shift operation. This allows for the derivation of fast RLS algorithms with a complexity of $O(N)$. The most popular between them is the Fast Transversal Filter (FTF) algorithm because of its lowest computational complexity which is equal to $7N$ [1]. However, these fast versions suffers from round-off error accumulation that leads to instability. Recently, a stabilized version of the FTF algorithm was derived which shows a computational complexity of $8N$ [10]. RLS algorithm is much more efficient than LMS algorithm since its superiority in convergence and tracking [3] but its complexity (even in the Fast versions) disqualify it from being used in an acoustic echo cancellation system.

In this report, we present a new fast algorithm for RLS adaptive filtering that constitutes a good solution to the acoustic echo cancellation problem. 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 FTF algorithm and uses the approximation that when dealing with Auto-Regressive (AR) 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 [8],[9]. 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 allows the reduction of the complexity to $\mathcal{O}(2N)$. The FNTF algorithm has been implemented successfully in a radio-mobile hands-free system [9]. It exhibited performances that are near to those of the RLS algorithm. However, when one deals with longer filters than those used in the mobile-radio context ($N > 256$), the FNTF algorithm and even the LMS algorithm can not be implemented because of todays technological limitations.

In [11],[12], 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

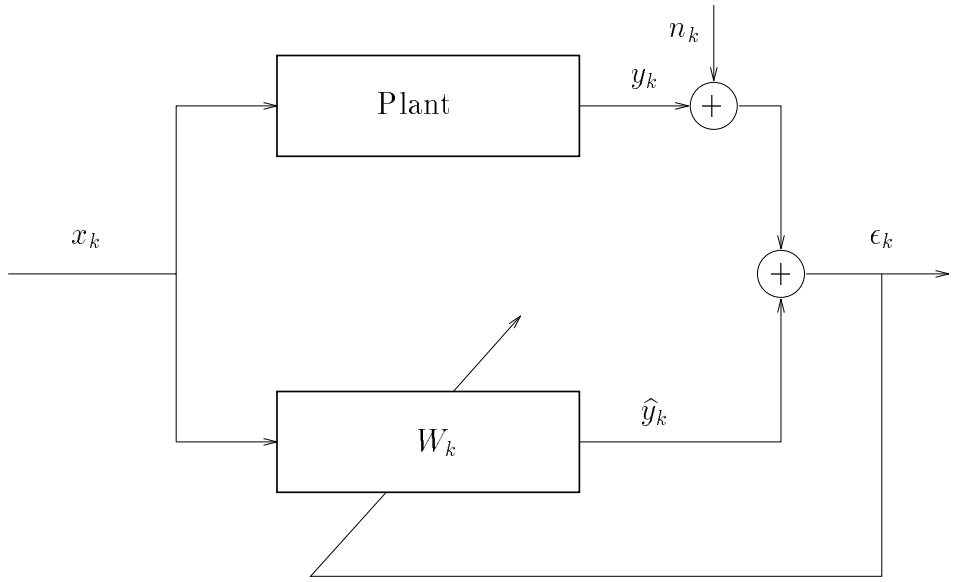

Figure 2: The identification scheme of an unknown plant.

algorithms which present a reduced complexity when dealing with long FIR filters. Here, we apply the subsampled-updating strategy (SUS) 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.
The rest of this report is organized as follows. In sections 2 and 3, we briefly recall the RLS and FNTF algorithms. In section 4, we introduce the Schur-FNTF algorithm that allows the computation of the filtering errors in a SUS. Section 5 deals with the fast computation of convolutions using the FFT in order to reduce the computational complexity of the Schur-FNTF procedure and the update of the filter estimate. This technique leads to the FSU FNTF algorithm which is given in section 6. Finally, some conclusions are presented in section 7.
In order to formulate the problem and to fix notation, we shall first recall the RLS algorithm and the FNTF algorithm. We shall mostly stick to the notation introduced in [1],[10],[2], except that the ordering of the rows in data vectors will be reversed (to transform a Hankel data matrix into a Toeplitz one) and some extra notation will be introduced.

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

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

where $X_N(i) = \left[x^H(i)\, x^H(i-1) \cdots x^H(i-N+1)\right]^H$ is the regression 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

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

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

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

where

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

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 [6, page 656] for $R_{N,k}^{-1}$, we obtain the RLS algorithm:

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

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

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

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

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

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

where $\epsilon_N^p(k)$ and $\epsilon_N(k)$ are the a priori and a posteriori error signals. They are related by the likelihood variable $\gamma_N(k)$ as in (9). The overnormalized Kalman gain $\tilde{C}_{N,k}$ is related to the unnormalized Kalman gain $C_{N,k}$:

$$C_{N,k} \;=\; -X_N^H(k) R_{N,k}^{-1} \;=\; \gamma_N(k) \tilde{C}_{N,k} \tag{11}$$

$$\gamma_N(k) \;=\; 1 + C_{N,k} X_N(k) \quad , \tag{12}$$

and the term overnormalized stems from the relation $\tilde{C}_{N,k} = \gamma_N^{-1}(k) C_{N,k}$. Equations (8)-(10) constitute the joint-process or filtering part of the RLS algorithm. Its computational complexity is $2N+1$. The role of the prediction part (5)-(7) is to produce the Kalman gain $\tilde{C}_{N,k}$ and the likelihood variable $\gamma_N(k)$ for the joint-process part. In the conventional RLS algorithm, this is done via the Riccati equation (7) which requires $\mathcal{O}(N^2)$ computations. Fast RLS algorithms exploit a certain shift invariance structure in $X_N(k)$ to avoid the Riccati equation in the prediction part and reduce its computational complexity to $\mathcal{O}(N)$. 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 $7N$. We now recall the FNTF algorithm, from which our new algorithm will be derived.

3 The FNTF Algorithm

In the FTF algorithm, the Kalman gain and the likelihood variable are computed in the prediction part of the algorith. The update of the sample inverse covariance is replaced by the update of its generators [7] 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 [8] suffers from the need for a significant data storage. In [9], this problem was overcome by using two prediction part FTF algorithms that run in parallel. The input signal of one prediction part being a delayed version of the input signal of the other part with a delay of $N-M$ samples.
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:

$$
\begin{aligned}
\widetilde{P}_k &= \Phi^p(k)\ P_{k-1} \\
\widetilde{P}_{k_d} &= \Phi^p(k_d)\ P_{k_d-1} \quad , \quad k_d = k - N + M \\
\begin{bmatrix} [\ \widetilde{C}_{N,M,k}\ 0\ ] \\ [\ W_{N,k}\ 0\ ] \end{bmatrix} &= \Phi(k) \begin{bmatrix} [\ P_{k-1}\ 0_{N-M}\ ] \\ [\ 0_{N-M}\ P_{k_d-1}\ ] \\ [\ 0\ \widetilde{C}_{N,M,k-1}\ ] \\ [\ W_{N,k-1}\ 0\ ] \end{bmatrix} \\
n = k, k_d\ :& \\
e_M^p(n) &= A_{M,n-1} X_{M+1}(n) \\
e_M(n) &= e_M^p(n)\gamma_M(n-1) \\
\gamma_{M+1}^{-s}(n) &= \gamma_M^{-1}(n-1) - \widetilde{C}_{M+1,n}^0 e_M^p(n) \\
\gamma_M^{-s}(n) &= \gamma_{M+1}^{-s}(n) + \widetilde{C}_{M+1,n}^M r_M^{pf}(n) \\
r_M^{ps}(n) &= -\lambda\beta_M(n-1)\widetilde{C}_{M+1,n}^{MH} \\
r_M^{pf}(n) &= B_{M,n-1} X_{M+1}(n) \\
\alpha_M^{-1}(n) &= \lambda^{-1}\alpha_M^{-1}(n-1) - \widetilde{C}_{M+1,n}^{0H}\ \gamma_{M+1}^s(n)\ \widetilde{C}_{M+1,n}^0 \qquad (13) \\
j = 1,2\ :& \qquad r_M^{p(j)}(n) = K_j r_M^{pf}(n) + (1-K_j) r_M^{ps}(n) \\
& \qquad r_M^{(j)}(n) = r_M^{p(j)}(n)\gamma_M^s(n) \\
\beta_M(n) &= \lambda\beta_M(n-1) + r_M^{(2)}(n)\, r_M^{p(2)^H}(n) \\
\gamma_M(n) &= \lambda^M \beta_M(n)\, \alpha_M^{-1}(n)
\end{aligned}
$$

$$\begin{aligned}
\widetilde{S}_{M+1,n} &= e_M^p(n)\lambda^{-1}\alpha_M^{-1}(n-1)A_{M,n-1} \\
\widetilde{U}_{M+1,n} &= r_M^{pf}(n)\lambda^{-1}\beta_M^{-1}(n-1)B_{M,n-1}
\end{aligned}$$

$$\begin{aligned}
\gamma_{N,M}(k) &= \gamma_{N,M}(k-1) + \widetilde{S}^0_{M+1,k}e_M^p(k) - \widetilde{U}^M_{M+1,k_d}r_M^{pf}(k_d) \\
\epsilon_N(k) &= \gamma_{N,M}(k)\epsilon_N^p(k) \ ,
\end{aligned}$$

where

$$\widetilde{P}_k = \begin{bmatrix} \begin{bmatrix} \widetilde{C}_{M,k} \ 0 \end{bmatrix} \\ A_{M,k} \\ B_{M,k} \end{bmatrix} , \ P_k = \begin{bmatrix} \begin{bmatrix} 0 \ \widetilde{C}_{M,k} \end{bmatrix} \\ A_{M,k} \\ B_{M,k} \end{bmatrix} , \tag{14}$$

$$\begin{aligned}
\Phi^p(k) &= \Phi_2^p(k)\ \Phi_1^p(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ r_M^1(k) & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & a & -\widetilde{C}^M_{M+1,k} \\ e_M(k) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
a &= -e_M^p(k)\lambda^{-1}\alpha_M^{-1}(k) \ ,
\end{aligned} \tag{15}$$

$$\begin{aligned}
\Phi(k) &= \Phi_2(k)\Phi_1(k) = \begin{bmatrix} 1 & 0 \\ \epsilon_N(k) & 1 \end{bmatrix} \begin{bmatrix} 0 & -\widetilde{C}^0_{M+1,k} & 0 & 0 & 0 & b & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
b &= r_M^{pf}(k_d)\lambda^{-1}\beta_M^{-1}(k_d-1) \ .
\end{aligned} \tag{16}$$

$A_{M,k}$ 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 [10].

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 the SUS. 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. The SUS does not necessarily involve approximations and the key ingredient is that even the adaptive filter is not updated all the time, it can be possible to compute efficiently the filtering errors that should be given if the filter estimate was updated at the sampling signal rate. The SUS leads to SU algorithms that are equivalent to the original algorithms, except for the fact that some quantities like the filter estimate are not provided at all the time instants. Moreover, fast convolution techniques can help to reduce the complexity of the SU algorithms and will give FSU algorithms. In the FNTF algorithm, the prediction part is not computationally demanding. Hence, the SUS is

applied to the filtering part of the FNTF. 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 filter estimate at these instants.

4 The Schur-FNTF algorithm

Let us introduce the negative of the filter output

$$\widehat{d}_N^p(k) = d(k) - \epsilon_N^p(k) \quad , \quad \widehat{d}_N(k) = d(k) - \epsilon_N(k) \ , \tag{17}$$

the reversed complex-conjugate regressor vector

$$x_{N,k} = [\ x_{k-N+1}\ x_{k-N+2} \cdots\ x_k\ ]^H \quad , \tag{18}$$

and the $L \times (N+1)$ Toeplitz input data matrix

$$X_{N+1,L,k} = \begin{bmatrix} X_{N+1}^H(k-L+1) \\ \vdots \\ X_{N+1}^H(k) \end{bmatrix} = [\ x_{L,k}\ x_{L,k-1} \cdots x_{L,k-N}\ ] \quad . \tag{19}$$

Now, consider the following set of filtering operations

$$F_L(k) \triangleq \begin{bmatrix} g_L^H(k) \\ g_L^H(k_d) \\ \eta_{N,L,k}^H \\ -\widehat{d}_{N,L,k}^{p\ H} \end{bmatrix} \triangleq \begin{bmatrix} [\ P_{k-L}\ 0_{N-M}\ ] \\ [\ 0_{N-M}\ P_{k_d-L}\ ] \\ \left[\ 0\ \widetilde{C}_{N,M,k-L}\ \right] \\ [\ W_{N,k-L}\ 0\ ] \end{bmatrix} X_{N+1,L,k}^H \quad , \quad g_L^H(k) \triangleq \begin{bmatrix} \eta_{M,L,k}^H \\ e_{M,L,k}^{p\ H} \\ r_{M,L,k}^{pf\ H} \\ \end{bmatrix} \quad . \tag{20}$$

$F_L(k)$ 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

$$\widehat{d}_{N,L,k}^p = d_{L,k} - \epsilon_{N,L,k}^p = \begin{bmatrix} d^H(k-L+1) \\ \vdots \\ d^H(k) \end{bmatrix} - \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-L) \end{bmatrix} = \begin{bmatrix} \widehat{d}_N^H(k-L+1|k-L) \\ \vdots \\ \widehat{d}_N^H(k|k-L) \end{bmatrix} \ . \tag{21}$$

The first column of $F_L(k)$ is

$$\begin{aligned} F_L(k)\, u_{L,1} &= \left[\ p^T(k)\ \ p^T(k_d)\ \ 1-\gamma_N^{-1}(k-L)\ \ -\widehat{d}_N^p(k-L+1)\ \right]^T \\ p^T(k) &= \left[\ 1-\gamma_M^{-1}(k-L)\ \ e_M^p(k-L+1)\ \ r_M^{pf}(k-L+1)\ \right] \ . \end{aligned} \tag{22}$$

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

$$\begin{bmatrix} \left[\, \widetilde{P}_k \; 0_{N-M} \,\right] \\ \left[\, 0_{N-M} \; \widetilde{P}_{k_d} \,\right] \\ \left[\, \widetilde{C}_{N,M,k} \; 0 \,\right] \\ \left[\, W_{N,k} \; 0 \,\right] \end{bmatrix} = \Theta_k \begin{bmatrix} \left[\, P_{k-1} \; 0_{N-M} \,\right] \\ \left[\, 0_{N-M} \; P_{k_d-1} \,\right] \\ \left[\, 0 \; \widetilde{C}_{N,M,k-1} \,\right] \\ \left[\, W_{N,k-1} \; 0 \,\right] \end{bmatrix} , \tag{23}$$

where Θ_k is a 8×8 matrix given by

$$\Theta_k = \begin{bmatrix} I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & \Phi_2(k) \end{bmatrix} \begin{bmatrix} \Phi_2^p(k) & 0 & 0 \\ 0 & \Phi_2^p(k_d) & 0 \\ 0 & 0 & I_2 \end{bmatrix} \begin{bmatrix} \Phi_1^p(k) & 0 & 0 \\ 0 & \Phi_1^p(k_d) & 0 \\ \Phi_1(k) & & I_2 \end{bmatrix} . \tag{24}$$

Hence, if we rotate both expressions for $F_L(k)$ in (20) with Θ_{k-L+1}, we obtain $\Theta_{k-L+1} F_L(k)$ which equals

$$\begin{bmatrix} \left[\, \widetilde{P}_{k-L+1} \; 0_{N-M} \,\right] \\ \left[\, 0_{N-M} \; \widetilde{P}_{k_d-L+1} \,\right] \\ \left[\, \widetilde{C}_{N,M,k-L+1} \; 0 \,\right] \\ \left[\, W_{N,k-L+1} \; 0 \,\right] \end{bmatrix} X^H_{N+1,L,k} = \begin{bmatrix} q_L(k) \\ q_L(k_d) \\ \boxed{\eta^H_{N,L-1,k}} \quad * \\ -\widehat{d}_N\,(k-L+1) \quad \boxed{-\widehat{d}^{\,p\,H}_{N,L-1,k}} \end{bmatrix} \tag{25}$$

$$q_L(k) = \begin{bmatrix} \boxed{\eta^H_{M,L-1,k}} \quad * \\ e_M(k-L+1) \quad \boxed{e^{p\,H}_{M,L-1,k}} \\ r^f_M(k-L+1) \quad \boxed{r^{pf\,H}_{M,L-1,k}} \end{bmatrix} . \tag{26}$$

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

$$\mathcal{S}\left(\Theta_{k-L+1}\, F_L(k)\right) = F_{L-1}(k) , \tag{27}$$

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 can be computed over the block data without updating the filter estimate. Now, since it is possible to compute the parameters of the successive matrices Θ_k, it suffices to accumulate the successive Θ_k and apply the resulting matrix to the filters in order to update them. In fact, it turns out that the accumulated matrix is a polynomial matrix. This is due to the shift operation in the update of the Kalman gain. Hence, the updating of the filters are done via convolutions. This is the Schur-FNTF algorithm, which contrasts with the Levinson-style FNTF algorithm in (13). This technique has already been applied to the FTF algorithm leading to what we have called the Schur-FTF algorithm [12]. 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 only be used for the computation of the successive a priori error filtering. Thus, the two prediction part FTF algorithms of order M are kept in order to compute the parameters of the successive matrices Θ_k. This avoids the accumulation of the successive rotation matrices and the update of the FNTF filters by mean of convolutions. 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 (22). This fact has a big importance for the real time implementation of the algorithm.

Taking into account that Θ_k in its factorized form (24) has 11 non-trivial elements, the Schur-FNTF procedure as given by (27) takes only $5.5L$ multiplications per sample. Inner products that represent filtering operations are needed for the initialization (computation of $F_L(k)$ in (20)). At this point, the Schur-FNTF procedure is computational demanding because the products in (20) requires $\mathcal{O}(N^2)$ multiplications per L incoming samples. We now consider the FFT technique to reduce the amount of operations.

5 Fast computation using the FFT

It is possible to reduce the computational complexity of the Schur-FNTF procedure by introducing FFT techniques as explained in [13]. In what follows, we shall often assume for simplicity that L is a power of two and that $N_L = (N+1)/L$ is an integer. To get $F_L(k)$ in (20), we need to compute products of the form $v_{N+1,k}\ X^H_{N+1,L,k}$ where $v_{N+1,k}$ is a row vector of $N+1$ elements.

Consider a partitioning of $v_{N+1,k}$ in N_L subvectors of length L:

$$v_{N+1,k} = \left[\ v^1_{N+1,k}\ \cdots\ v^{N_L}_{N+1,k}\ \right] \quad , \tag{28}$$

and a partitioning of $X_{N+1,L,k}$ in N_L submatrices of order $L \times L$:

$$X_{N+1,L,k} = [\ X_{L,L,k}\ X_{L,L,k-L}\ \cdots X_{L,L,k-N+L-1}\ ] \quad , \tag{29}$$

then

$$v_{N+1,k}\ X^H_{N+1,L,k} = \sum_{j=1}^{N_L} v^j_{N+1,k} X^H_{L,L,k-(j-1)L} \quad . \tag{30}$$

In other words, we have essentially N_L times the product of a a vector of length L with a $L \times L$ Toeplitz matrix. Such a product can be efficiently computed in basically two different ways [13]. One way is to use fast convolution algorithms , which are interesting for moderate values of L. Another way is to use the overlap-save method. We can embed the $L \times L$ Toeplitz matrix $X_{L,L,k}$ into a $2L \times 2L$ circulant matrix, viz.

$$\underline{X}^H_{L,L,k} \ = \ \begin{bmatrix} * & X^H_{L,L,k} \\ X^H_{L,L,k} & * \end{bmatrix} \ = \ \mathcal{C}\left(x^H_{2L,k}\right) \tag{31}$$

where $\mathcal{C}(c^H)$ is a right shift circulant matrix with c^H as first row. Then we get for the vector-matrix product

$$v^j_{N+1,k} X^H_{L,L,k-(j-1)L} = \left[\ 0_{1\times L}\ v^j_{N+1,k}\ \right]\ \mathcal{C}\left(x^H_{2L,k-(j-1)L}\right) \begin{bmatrix} I_L \\ 0_{L\times L} \end{bmatrix} \quad . \tag{32}$$

Now consider the Discrete Fourier Transform (DFT) $\mathcal{V}^j_{N+1,k}$ of $v^j_{N+1,k}$

$$\mathcal{V}^j_{N+1,k} = v^j_{N+1,k}\ F_L \quad , \tag{33}$$

F_L is the $L \times L$ DFT matrix whose generic element is $(F_L)_{p,q} = e^{-i2\pi\frac{(p-1)(q-1)}{L}}$, $i^2 = -1$. The inverse of F_L is $\frac{1}{L}F^H_L$. It defines the inverse DFT transformation (IDFT)

$$v^j_{N+1,k} = \mathcal{V}^j_{N+1,k}\ \frac{1}{L}F^H_L \quad . \tag{34}$$

The product of a row vector v with a circulant matrix $\mathcal{C}(c^H)$ where v and c are of length m can be computed efficiently as follows. Using the property that a circulant matrix can be diagonalized via a similarity transformation with a DFT matrix, we get

$$v\ \mathcal{C}(c^H) \ = \ v\ F_m\ \mathrm{diag}\left(c^H F_m\right)\frac{1}{m}F^H_m \ = \ \left[\,(vF_m)\ \ \mathrm{diag}\left(c^H F_m\right)\,\right]\ \frac{1}{m}F^H_m \quad , \tag{35}$$

where $\mathrm{diag}(w)$ is a diagonal matrix with the elements of the vector w as diagonal elements. So the computation of the vector in (32) requires the padding of v with L zeros, the DFT of the resulting vector, the DFT of $x_{2L,k-(j-1)L}$, the product of the two DFTs, and the (scaled) IDFT of this product. When the FFT is used to perform the DFTs, this leads to a computationally more efficient procedure than the straightforward matrix-vector product which would require L^2 multiplications. Note that at time k, only the FFT of $x_{2L,k}$ needs to be computed; the FFTs of $x_{2L,k-jL}, j = 1, \ldots, M-1$ have been computed at previous time instants. The above procedure will only be used for the product of $\left[\ 0\ \widetilde{C}_{N,M,k-L}\ \right]$ and $W_{N,k-L}$ with the data matrix. since the other vectors are of length M which is relatively small. This reduces the $(2N+6M)$ computations per sample that are needed for the initialization of the schur-FNTF procedure to

$$2N\ \left[\frac{\mathrm{FFT}(2L)}{L^2} + \frac{2}{L}\right] + \frac{3\mathrm{FFT}(2L)}{L} + 6M \tag{36}$$

computations per sample ($\mathrm{FFT}(L)$ signifies the computational complexity associated with a FFT of length L) or basically $\mathcal{O}\left(N\ \frac{\log_2(L)}{L}\right)$ operations.

6 The FSU FNTF algorithm

The issue now is the computation of the filter estimate and the Kalman gain at time k from their value L instants before. One finds straightforwardly for the estimate filter

$$[\ W_{N,k}\ 0\ ] = [\ W_{N,k-L}\ 0\ ] + \underline{\epsilon}^H_{N,L,k}\ \left[\ \underline{\widetilde{C}}_{N,M,k}\ 0\ \right]\ , \tag{37}$$

where

$$\underline{\epsilon}^H_{N,L,k} = [\ \epsilon_N(k-L+1)\ \cdots\ \epsilon_N(k)\ ] \tag{38}$$

$$\underline{\widetilde{C}}_{N,M,k} = \begin{bmatrix} \widetilde{C}_{N,M,k-L+1} \\ \vdots \\ \widetilde{C}_{N,M,k} \end{bmatrix} . \tag{39}$$

$\underline{\epsilon}^H_{N,L,k}$ is the vector of L successive a posteriori (one step ahead) output errors. It is obtained from $\underline{\epsilon}^{p\ H}_{N,L,k}$ which is computed efficiently with the Schur-FNTF algorithm and the FFT as shown in previous sections. $\underline{\widetilde{C}}_{N,M,k}$ is the $L \times N$ Kalman gain matrix where the rows are the L successive FNTF Kalman gains. In order to compute this matrix, we come back to the FNTF Kalman gain update

$$\left[\ \widetilde{C}_{N,M,k}\ 0\ \right] = \left[\ 0\ \widetilde{C}_{N,M,k-1}\ \right] - \left[\ \widetilde{S}_{M+1,k}\ 0_{N-M}\ \right] + \left[\ 0_{N-M}\ \widetilde{U}_{M+1,k_d}\ \right]\ , \tag{40}$$

definitions of $\widetilde{S}_{M+1,k}$ and $\widetilde{U}_{M+1,k_d}$ are given in (13). From the above equation, it is easy to see that each row of the Kalman gain matrix can be expressed as follows

$$\begin{aligned} \text{for } i &= 1,\ldots,L \\ \left[\ \widetilde{C}_{N,M,k-L+i}\ \ 0\ \right] &= \left[\ 0_i\ \ \widetilde{C}^{0:N-i}_{N,M,k-L}\ \right] - \sum_{l=0}^{i-1} \left[\ \widetilde{S}_{M+1,k_l}\ 0_{N-M}\ \right] Z^l + \sum_{l=0}^{i-1} \left[\ 0_{N-M}\ \widetilde{U}_{M+1,k_l}\ \right] Z^l \\ k_l &= k-L+i-l\ , \end{aligned} \tag{41}$$

where

$$Z = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{42}$$

is the $(N+1) \times (N+1)$ right shift matrix.
By replacing (41) into (39) for every row, we obtain the expression of the Kalman gain matrix $\underline{\widetilde{C}}_{N,M,k}$ in term of the Kalman gain vector $\widetilde{C}_{N,M,k-L}$

$$\begin{aligned} \left[\ \underline{\widetilde{C}}_{N,M,k}\ 0\ \right] &= \left[\left[\ 0\ \widetilde{C}_{N,M,k-L}\ \right] Z^i\right]_{i=1:L} - \left[\sum_{l=0}^{i-1} \left[\ \widetilde{S}_{M+1,k_l}\ 0_{N-M}\ \right] Z^l\right]_{i=1:L} \\ &\qquad + \left[\sum_{l=0}^{i-1} \left[\ 0_{N-M}\ \widetilde{U}_{M+1,k_l}\ \right] Z^l\right]_{i=1:L} \\ &= \mathcal{T}\left(\left[\ 0\ \widetilde{C}_{N,M,k-L}\ \right]\right) - \widetilde{S}_{N+1,L,k} + \widetilde{U}_{N+1,L,k}\ , \end{aligned} \tag{43}$$

the notation $[a_i]_{i=1:L}$ stands for the matrix in which the ith row is a_i. The first term of the right hand side of (43) is a $L \times (N+1)$ Toeplitz matrix whose first row is $\left[\, 0 \; \widetilde{C}_{N,M,k-L} \,\right]$ and the two others are $L \times (N+1)$ sparse matrices and have the following structures

$$\widetilde{S}_{N+1,L,k} = \begin{bmatrix} * & \cdots & * & 0 & & \cdots & \cdots & 0 \\ \vdots & & \vdots & \ddots & \ddots & & & \vdots \\ * & \cdots & * & \cdots & * & 0 & \cdots & 0 \end{bmatrix}$$

$$\widetilde{U}_{N+1,L,k} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \end{bmatrix} \quad , \tag{44}$$

the $*$ standing for nonzero elements. First rows of $\widetilde{S}_{N+1,L,k}$ and $\widetilde{U}_{N+1,L,k}$ have $M+1$ nonzero elements. The matrices in (44) are efficiently computed by applying the following recursions to their rows

$$\left(\widetilde{S}_{N+1,L,k}\right)_1 = \left[\, \widetilde{S}_{M+1,k-L+1} \; 0_{N-M} \,\right] \quad , \quad \left(\widetilde{U}_{N+1,L,k}\right)_1 = \left[\, 0_{N-M} \; \widetilde{U}_{M+1,k-L+1} \,\right] \quad ,$$

for $i = 2, \ldots, L$

$$\begin{aligned} \left(\widetilde{S}_{N+1,L,k}\right)_i &= Z\left(\widetilde{S}_{N+1,L,k}\right)_{i-1} + \left[\, \widetilde{S}_{M+1,k-L+i} \; 0_{N-M} \,\right] \\ \left(\widetilde{U}_{N+1,L,k}\right)_i &= Z\left(\widetilde{U}_{N+1,L,k}\right)_{i-1} + \left[\, 0_{N-M} \; \widetilde{U}_{M+1,k-L+i} \,\right] \quad . \end{aligned} \tag{45}$$

The recursions in (45) need $2(L-1)M$ additions per L samples. On the other hand, $\widetilde{C}_{N,M,k}$ is updated by using the last row of (43)

$$\left[\, \widetilde{C}_{N,M,k} \; 0 \,\right] = \left[\, 0_L \; \widetilde{C}^{0:N-L}_{N,M,k-L} \,\right] - \left(\widetilde{S}_{N+1,L,k}\right)_L + \left(\widetilde{U}_{N+1,L,k}\right)_L \quad , \tag{46}$$

$\left(\widetilde{S}_{N+1,L,k}\right)_L$ has its first $M+L$ elements that are nonzero while the nonzero elements of $\left(\widetilde{U}_{N+1,L,k}\right)_L$ are in the last M positions. Henceforth, the update of the Kalman gain vector (46) needs $2M$ additions per L samples (we consider the case where nonzero elements of $\left(\widetilde{S}_{N+1,L,k}\right)_L$ and $\left(\widetilde{U}_{N+1,L,k}\right)_L$ are in disjoint portions. This happens when $L+2M < N+1$).
By using the decompositon of the Kalman gain matrix in (43), we split the product in (37) as follows

$$\underline{\epsilon}^H_{N,L,k} \left[\, \underline{\widetilde{C}}_{N,M,k} \; 0 \,\right] = \underline{\epsilon}^H_{N,L,k} \, \mathcal{T}\left(\left[\, 0 \; \widetilde{C}_{N,M,k-L} \,\right]\right) - \underline{\epsilon}^H_{N,L,k} \, \widetilde{S}_{N+1,L,k} + \underline{\epsilon}^H_{N,L,k} \, \widetilde{U}_{N+1,L,k} \quad . \tag{47}$$

Since $\left(\widetilde{S}_{N+1,L,k}\right)_L$ and $\left(\widetilde{U}_{N+1,L,k}\right)_L$ are sparce matrices, second and third products in (47) are straightforwardly done. These products take respectively $ML + 0.5L(L-1)$ and ML multiplications per L samples. First matrix being Toeplitz, we can further reduce the NL^2 multiplications per L samples which are needed for the computation of $\underline{\epsilon}^H_{N,L,k} \, \mathcal{T}\left(\left[\, 0 \; \widetilde{C}_{N,M,k-L} \,\right]\right)$ by using the FFT technique.
Consider the following decomposition of $\mathcal{T}\left(\left[\, 0 \; \widetilde{C}_{N,M,k-L} \,\right]\right)$ in N_L submatrices of orde $L \times L$

$$\mathcal{T}\left(\left[\, 0 \; \widetilde{C}_{N,M,k-L} \,\right]\right) = \left[\, \mathcal{T}^1_{L,L} \; \cdots \; \mathcal{T}^{N_L}_{L,L} \,\right] \quad , \tag{48}$$

and hence, the product becomes

$$\underline{\epsilon}_{N,L,k}^{H}\,\mathcal{T}\left(\left[\begin{array}{c} 0\ \widetilde{C}_{N,M,k-L} \end{array}\right]\right) = \left[\begin{array}{c} \underline{\epsilon}_{N,L,k}^{H}\,\mathcal{T}_{L,L}^{1}\ \cdots\ \underline{\epsilon}_{N,L,k}^{H}\,\mathcal{T}_{L,L}^{N_L} \end{array}\right] \quad , \tag{49}$$

every subproduct in (49) is computed as follows:

$$\begin{aligned} \underline{\epsilon}_{N,L,k}^{H}\,\mathcal{T}_{L,L}^{j} &= \left[\begin{array}{c} 0_L\ \underline{\epsilon}_{N,L,k}^{H} \end{array}\right] \overline{\mathcal{T}}_{L,L}^{j} \left[\begin{array}{c} I_L \\ 0_{L\times L} \end{array}\right] \\ &= \left[\left(\left[\begin{array}{c} 0_L\ \underline{\epsilon}_{N,L,k}^{H} \end{array}\right] F_{2L}\right) \mathrm{diag}(\tau^{j\,H} F_{2L})\right] \frac{1}{2L} F_{2L}^{H} \left[\begin{array}{c} I_L \\ 0_{L\times L} \end{array}\right] \quad , \end{aligned} \tag{50}$$

where $\tau^{j\,H}$ is the first row of the $2L \times 2L$ shift right matrix obtained by embedding $\mathcal{T}_{L,L}^{j}$ in the same manner as was done for the data Toeplitz matrix $X_{L,L,k}^{j}$ in section 5. As it is shown in (50), the product $\underline{\epsilon}_{N,L,k}^{H}\,\mathcal{T}\left(\left[\begin{array}{c} 0\ \widetilde{C}_{N,M,k-L} \end{array}\right]\right)$ is done by adding L zeros to $\underline{\epsilon}_{N,L,k}^{H}$, computing the corresponding DFT, computing the DFT of $\tau^{j\,H}$ (there is N_L vectors like this), computing the product of the two DFTs , applying the IDFT to the product and finally taking the first L elements of the result. This is done in $\left(2\frac{N+1}{L}+1\right)\frac{FFT(2L)}{L}+2\frac{N+1}{L}$ multiplications per sample. The resulting FSU FNTF algorithm is summarized in Table I.

7 Conclusions

We have derived a new algorithm that is equivalent to the FNTF algorithm. The FSU FNTF (and the FNTF) algorithm exhibit the same performances that are almost those of the RLS algorithm, especially when the input signal is an AR process. The computational complexity of the FSU FNTF algorithm is $\mathcal{O}(4\frac{N}{L}\frac{FFT(2L)}{L}+6\frac{N}{L}+6L+16M)$ 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 ($FFT(2m) = m log_2(2m)$ real multiplications for real signals) the multiplicative complexity is respectively $0.63N$ and $0.40N$, compared to $7N$ for the FTF algorithm, the currently fastest RLS algorithm, and $2N$ for the FNTF algorithm. The cost we pay is a processing delay which is of the order of L samples. Moreover, the FSU FNTF algorithm removes the long-term round-off error instability of the likelihood variable that appears in the FNTF algorithm. 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.

Table I : FSU FNTF Algorithm

#	Computations	Cost per L samples
1	Two Stabilized FTF prediction parts	$12ML$
2	$\begin{bmatrix} g_L^H(k) \\ g_L^H(k_d) \\ \eta_{N,L,k}^{p\,H} \\ -\widehat{d}_{N,L,k}^{p\,H} \end{bmatrix} = \begin{bmatrix} [\,P_{k-L}\;0_{N-M}\,] \\ [\,0_{N-M}\;P_{k_d-L}\,] \\ \left[\,0\;\widetilde{C}_{N,M,k-L}\,\right] \\ [\,W_{N,k-L}\;0\,] \end{bmatrix} X_{N+1,L,k}^H$	$(3+2\frac{N+1}{L})FFT(2L)+4(N+1)+6M$
3	Schur-FNTF Algorithm : Input: $g_L(k),\ g_L(k_d),\ \eta_{N,L,k}^p,\ -\widehat{d}_{N,L,k}^p$ Output: $\Theta_{k-L+i} \quad i=1,\cdots,L \quad , \quad \underline{\epsilon}_{N,L,k}$	$5.5L^2$
4	$\left[\,\underline{\widetilde{C}}_{N,M,k}\;0\,\right] = \mathcal{T}\left(\left[\,0\;\widetilde{C}_{N,M,k-L}\,\right]\right) - \widetilde{S}_{N+1,L,k} + \widetilde{U}_{N+1,L,k}$	$2M(L-1)$
5	$\left[\,\widetilde{C}_{N,M,k}\;0\,\right] = \left[\,0_L\;\widetilde{C}_{N,M,k-L}^{0:N-L}\,\right] - \left(\widetilde{S}_{N+1,L,k}\right)_L + \left(\widetilde{U}_{N+1,L,k}\right)_L$	$2M$
6	$[\,W_{N,k}\;0\,] = [\,W_{N,k-L}\;0\,] + \underline{\epsilon}_{N,L,k}^H\left[\,\underline{\widetilde{C}}_{N,M,k}\;0\,\right]$	$(1+2\frac{N+1}{L})FFT(2L)+2(N+1)+.5L^2+2ML$
Total cost per sample		$4(1+\frac{N+1}{L})\frac{FFT(2L)}{L}+6\frac{N+1}{L}+6L+16M$

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] 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.

[3] 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.

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

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

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

[7] 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.).

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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