The Fast Subsampled-Updating Fast Newton Transversal Filter (FSU FNTF) Algorithm for Adaptive Filtering

Karim Maouche Dirk T.M. Slock

Institut EURECOM
B.P. 193, 06904 Sophia Antipolis Cedex, FRANCE

Abstract

The Fast Newton Transversal Filter (FNTF) algorithm starts from the Recursive Least-Squares algorithm for adapting an FIR filter. 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 procedure 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, an algorithm that is mathematically equivalent to the FNTF algorithm in the sense that exactly the same filter output is produced. However, it shows a significantly smaller computational complexity for large filter lengths at the expense of some relatively small delay. The FSU FNTF algorithm (like the FNTF algorithm) has good convergence and tracking properties. This renders the FSU FNTF algorithm very interesting for applications such as acoustic echo cancellation.

Corresponding author: Karim Maouche
Institut eurecom, 2229 route des Crêtes, B.P. 193, 06904 Sophia Antipolis Cedex, France
Tel: +33 4 93 00 26 32, Fax: +33 4 93 00 26 27, E-mail: maouche@eurecom.fr

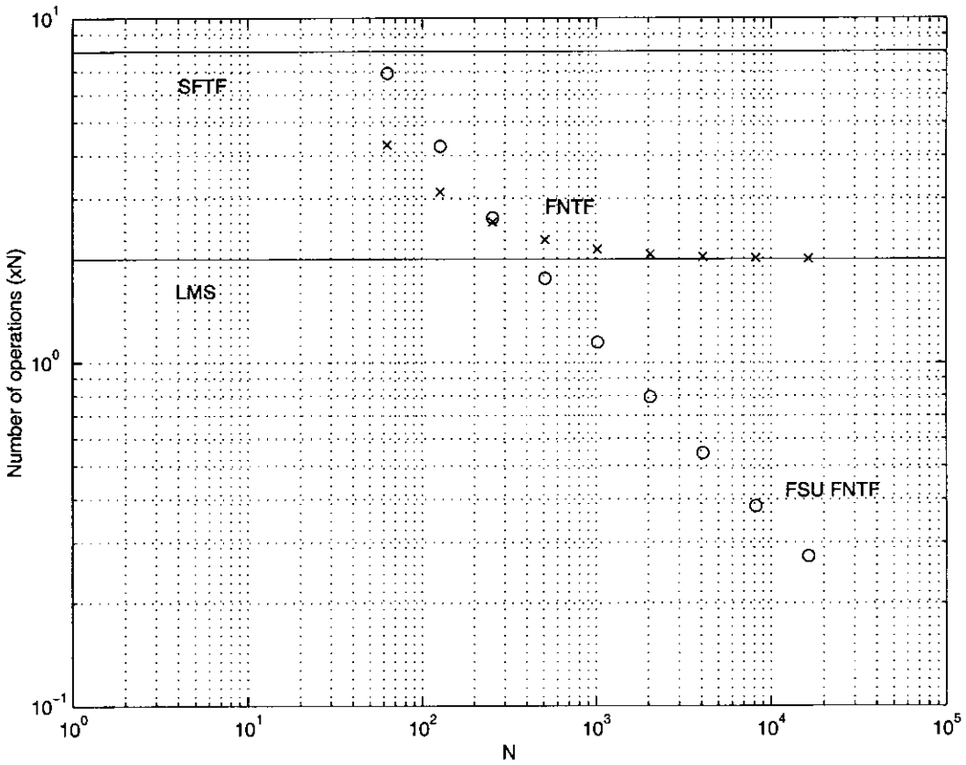

Figure 3: Optimal complexity of the FSU FNTF algorithm *vs.* filter length N.

The Fast Subsampled-Updating Fast Newton Transversal Filter (FSU FNTF) Algorithm for Adaptive Filtering

Karim Maouche Dirk T.M. Slock

Institut EURECOM
B.P. 193, 06904 Sophia Antipolis Cedex, FRANCE

Abstract

The Fast Newton Transversal Filter (FNTF) algorithm starts from the Recursive Least-Squares algorithm for adapting an FIR filter. 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 procedure 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, an algorithm that is mathematically equivalent to the FNTF algorithm in the sense that exactly the same filter output is produced. However, it shows a significantly smaller computational complexity for large filter lengths at the expense of some relatively small delay. The FSU FNTF algorithm (like the FNTF algorithm) has good convergence and tracking properties. This renders the FSU FNTF algorithm very interesting for applications such as acoustic echo cancellation.

Corresponding author: Karim Maouche
Institut eurecom, 2229 route des Crêtes, B.P. 193, 06904 Sophia Antipolis Cedex, France
Tel: +33 4 93 00 26 32, Fax: +33 4 93 00 26 27, E-mail: maouche@eurecom.fr

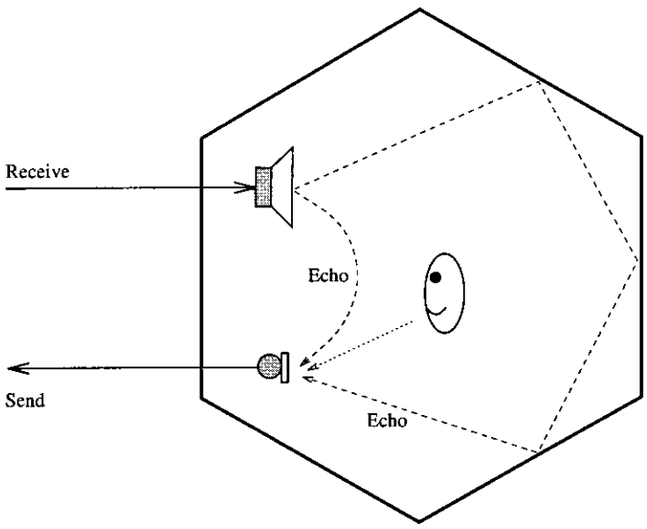

Figure 1: Acoustic echo phenomenon in hands-free communications.

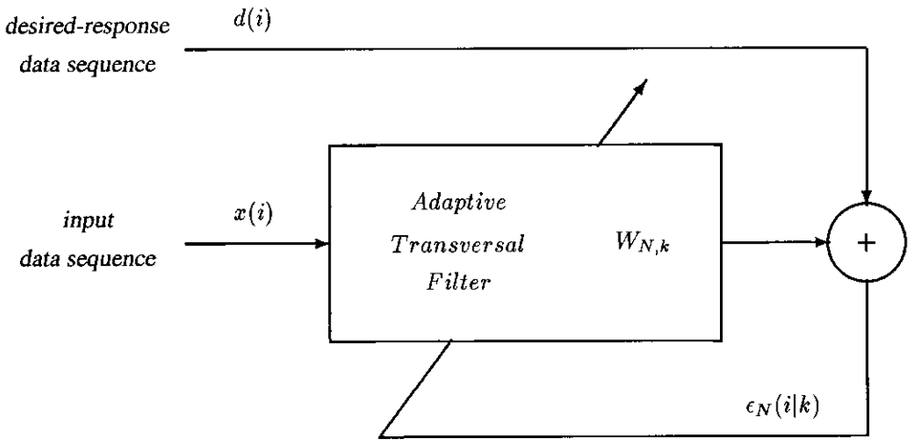

Figure 2: The adaptive FIR filtering scheme.

1 Introduction

Hands-free communications have become popular during the last few years. In these new systems, the 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 telephones in cars or can lead 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 [1] for a bibliography on the subject). 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 and 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 solution does not allow a real full-duplex communication. Moreover, it can degrade significantly the communication. 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 is to synthesize an estimate of the real echo by using an adaptive filter (which models the acoustic path) whose input is the loudspeaker signal. The echo estimate is subtracted from the microphone signal (called the desired signal) which is the sum of the acoustic echo and the local sounds (double talk and background noise). This subtraction gives the filtering error which is sent to the far end speaker and which controls the updating mechanism of the adaptive filter. 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. Often, this translates into thousands of samples for sampling frequencies that lead to satisfactory audio quality. The number of degrees of freedom is close to the number of samples of the acoustic impulse response.

The FSU FNTF Algorithm		
#	**Computation**	**Cost per L samples**
1	2 SFTF Prediction part	$8ML$
2	$\begin{bmatrix} g_L^H(k) \\ g_L^H(k^d) \\ \eta_{N,L,k}^{p\,H} \\ -\widehat{d}_{N,L,k}^{p\,H} \end{bmatrix} = \begin{bmatrix} [\, P_{k-L}\ 0_{N-M} \,] \\ [\, 0_{N-M}\ P_{k^d-L} \,] \\ \left[\, 0\ \widetilde{C}_{N,M,k-L} \,\right] \\ [\, W_{N,k-L}\ 0 \,] \end{bmatrix} X_{N+1,L,k}^H$	$(3+2\frac{N+1}{L})FFT(2L)+4N+6ML$
3	Schur-FNTF Procedure: Input: $g_L(k),\ g_L(k^d),\ \eta_{N,L,k}^p,\ -\widehat{d}_{N,L,k}^p$ Output: $\Phi_{k-L+i}\ ,\ i=1,\cdots,L\ ,\ \underline{\epsilon}_{N,L,k}$	$5.5L^2$
4	Compute $\widetilde{S}_{N+1,L,k}$ and $\widetilde{U}_{N+1,L,k}$	$2M(L-1)$
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	$[\, W_{N,k}\ \ 0 \,] = [\, W_{N,k-L}\ 0 \,] + \underline{\epsilon}_{N,L,k}^H \left[\, \underline{\widetilde{C}}_{N,M,k}\ 0 \,\right]$	$(1+2\frac{N+1}{L})FFT(2L)+2N+.5L^2+2ML$
Total cost per sample:		$4(1+\frac{N+1}{L})\frac{FFT(2L)}{L}+6\frac{N+1}{L}+6L+18M$

Table 2: The FSU FNTF Algorithm.

N	63	127	255	511	1023	2047	4095	8191	16383
L (optimal)	16	16	32	64	64	128	128	256	256
Complexity ($\times N$)	6.92	4.25	2.64	1.76	1.15	0.79	0.55	0.38	0.27

Table 3: Optimal Complexity of the FSU FNTF Algorithm ($M = 16$).

- The echo path varies in the case of movement of people or objects in the room. It varies also (at a slower rate) with ambient temperature, pressure and humidity changes.
- The input and disturbance signals in the system identification are speech signals. They are greatly correlated and nonstationary. Moreover, silent periods of the input signal render the identification problem ill-conditioned.

There are two major classes of adaptive algorithms. One is the Least-Mean-Square (LMS) algorithm which is based on a stochastic-gradient method [2],[3]. The LMS algorithm has a computational complexity of $2N$ (N is the Finite Impulse Response (FIR) filter length). This renders the LMS algorithm very popular. Nevertheless, its performance becomes worse 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 [4]. The RLS algorithm shows a complexity of $\mathcal{O}(N^2)$. However, for FIR filtering, consecutive regression vectors are related through a shift operation. This allows for the derivation of fast RLS algorithms with a complexity of $\mathcal{O}(N)$. The most popular among them is the Fast Transversal Filter (FTF) algorithm because of its low computational complexity equal to $7N$ [5]. Unfortunately, these fast versions suffer from roundoff error accumulation that leads to numerical instability. Recently, a numerically Stabilized version of the FTF algorithm (SFTF algorithm) was derived which shows a computational complexity of $8N$ [6]. The RLS algorithm is much more statistically efficient than the LMS algorithm whith superior performances [7]. However its complexity (even in the fast versions) disqualifies it from being used in acoustic echo cancellation systems.

This situation has motivated quite a bit of research to develop alternative solutions. One can distinguish three main areas. The first one uses Infinite Impulse Response (IIR) adaptive filtering in order to reduce the number of degrees of freedom. Unfortunately, IIR adaptive filters drawbacks such as instability and poor convergence properties. The second area consists of synthesizing adaptive algorithms that are between the two major families of adaptive algorithms in terms of computational complexity and performance. For example, we can cite the Fast Newton Transversal Filter (FNTF) algorithm [8], [9] which is an approximation to the FTF algorithm and the Fast Affine Projection (FAP) algorithm [10],[11]which is a generalization to the Normalized LMS (NLMS) algorithm. The third direction of research is the one where block adaptive algorithms are used. One can find among

The FNTF Algorithm		
#	**Computation**	**Cost per sample**
	$k^d = k-N+M \quad , \quad n = k, k^d$	
1	$e_M^p(n) = A_{M,n-1} X_{M+1}(n)$	M
2	$\widetilde{C}^{0\,H}_{M+1,n} = -\lambda^{-1}\alpha_M^{-1}(n-1) e_M^p(n)$	
3	$\widetilde{S}_{M+1,n} = \widetilde{C}^0_{M+1,n} A_{M,n-1}$	
4	$\widetilde{C}_{M+1,n} = \begin{bmatrix} 0 & \widetilde{C}_{M,n-1} \end{bmatrix} + \widetilde{S}_{M+1,n}$	M
5	$\gamma^{-s}_{M+1}(n) = \gamma_M^{-1}(n-1) - \widetilde{C}^0_{M+1,n} e_M^p(n)$	
6	$e_M(n) = e_M^p(n)\gamma_M(n-1)$	
7	$A_{M,n} = A_{M,n-1} + e_M(n) \begin{bmatrix} 0 & \widetilde{C}_{M,n-1} \end{bmatrix}$	M
8	$\alpha_M^{-1}(n) = \lambda^{-1}\alpha_M^{-1}(n-1) - \widetilde{C}^{0\,H}_{M+1,n}\gamma^s_{M+1}(n)\widetilde{C}^0_{M+1,n}$	
9	$r_M^{p\,f}(n) = B_{M,n-1} X_{M+1}(n)$	M
10	$r_M^{p\,s}(n) = -\lambda\beta_M(n-1)\widetilde{C}^{M\,H}_{M+1,n}$	
11	$r_M^{p\,(1)}(n) = 1.5\, r_M^{p\,f}(n) - 0.5\, r_M^{p\,s}(n)$	
12	$r_M^{p\,(2)}(n) = 2.5\, r_M^{p\,f}(n) - 1.5\, r_M^{p\,s}(n)$	
13	$\widetilde{U}_{M+1,n} = \widetilde{C}^M_{M+1,n} B_{M,n-1}$	
14	$\begin{bmatrix} \widetilde{C}_{M,n} & 0 \end{bmatrix} = \widetilde{C}_{M+1,n} - \widetilde{U}_{M+1,n}$	M
15	$\gamma_M^{-s}(n) = \gamma^{-s}_{M+1}(n) + \widetilde{C}^M_{M+1,n} r_M^{p\,f}(n)$	
16	$r_M^{(j)}(n) = r_M^{p\,(j)}(n)\gamma_M^s(n)\ ,\ j = 1,2$	
17	$B_{M,n} = B_{M,n-1} + r_M^{(1)}(n) \begin{bmatrix} \widetilde{C}_{M,n} & 0 \end{bmatrix}$	M
18	$\beta_M(n) = \lambda\beta_M(n-1) + r_M^{(2)}(n) r_M^{p\,(2)^H}(n)$	
19	$\gamma_M^{-1}(n) = \lambda^{-M}\alpha_M(n)\beta_M^{-1}(n)$	
20	$\gamma^{-1}_{N,M}(k) = \gamma^{-1}_{N,M}(k-1) - \widetilde{S}^0_{M+1,k} e_M^p(k) + \widetilde{U}^M_{M+1,k^d} r_M^{p\,f}(k^d)$	
21	$\widetilde{C}_{N+1,M,k} = \begin{bmatrix} 0 & \widetilde{C}_{N,M,k-1} \end{bmatrix} + \begin{bmatrix} \widetilde{S}_{M+1,k} & 0_{N-M} \end{bmatrix}$	
22	$\begin{bmatrix} \widetilde{C}_{N,M,k} & 0 \end{bmatrix} = \widetilde{C}_{N+1,M,k} - \begin{bmatrix} 0_{N-M} & \widetilde{U}_{M+1,k^d} \end{bmatrix}$	
23	$\epsilon_N^p(k) = d(k) + W_{N,k-1} X_N(k)$	N
24	$\epsilon_N(k) = \gamma_{N,M}(k)\epsilon_N^p(k)$	
25	$W_{N,k} = W_{N,k-1} + \epsilon_N(k)\widetilde{C}_{N,M,k}$	N
Total cost per sample:		$2N+12M$

Table 1: The FNTF Algorithm.

them, block algorithms that are mathematically equivalent to the sample by sample version such as the Fast Exact LMS [12] or frequency domain adaptive algorithms and subband adaptive algorithms [13].

In this paper, we present a new fast algorithm for approximate RLS adaptive filtering that constitutes a good solution to the acoustic echo cancellation problem or to any problem which deals with long FIR adaptive filtering. The Fast Subsampled-Updating Fast Newton Transversal Filter (FSU FNTF) algorithm is based upon the FNTF algorithm. The FNTF algorithm originates 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 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 has been shown to perform close to the RLS algorithm. However, when one deals with longer filters than those used in the mobile-radio context ($N > 256$ at a sampling frequency of 8 KHz), the FNTF algorithm and even the LMS algorithm can not be implemented because of current technological limitations.

In [14],[15] 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; the FSU RLS and FSU SFTF algorithms which present a reduced complexity for relatively long FIR filters. These two algorithms are mathematically equivalent to the RLS algorithm (in the sense that they both provide the same filtering output than the one of the RLS algorithm and give the same filter estimates at the subsampling instants), but have completely different algorithmic structures. Moreover, the FSU SFTF algorithm is numerically stable which is not the case for the FSU RLS algorithm.

Here, we apply the Subsampled-Updating Strategy (SUS) to the FNTF algorithm. This algorithm is divided in two parts, a prediction part and a filtering part. Since the prediction part has a relatively small computational complexity, we will keep it unchanged and the SUS will only be applied to the filtering part. Hence, the FNTF Kalman gain and the filter estimate are updated from quantities that were available L instants before. Now, even though, the filter estimate is provided every L

[12] J. Benesty and P. Duhamel, "A Fast Exact Least Mean Square Adaptive Algorithm," *IEEE Trans. Sig. Proc.*, vol. SP-40, no. 12, pp. 2904–2920, Dec. 1992.

[13] A. Gilloire, Eric Moulines, Dirk Slock, and Pierre Duhamel, "State of the Art in Acoustic Echo Cancellation," in *Digital Signal Processing in Telecommunications*, Anibal R. Figueiras-Vidal, Ed., pp. 45–91. Springer-Verlag, 1996.

[14] 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. 2, pp. 5–20, 1994.

[15] K. Maouche and D. T. M. Slock, "The Fast Subsampled-Updating Stabilized Fast Transversal Filter (FSU SFTF) Algorithm for Adaptive Filtering," *submitted to IEEE Trans. SP*, 1999.

[16] J.M. Cioffi and T. Kailath, "Windowed Fast Transversal Filters Adaptive Algorithms with Normalization," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. ASSP-33, no. 3, pp. 607–625, June 1985.

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

[18] T. Kailath, S.Y. Kung, and M. Morf, "Displacement ranks of matrices and linear equations," *J. Math. Anal. Appl.*, vol. 68, no. 2, pp. 295–407, 1979, (See also *Bull. Amer. Math. Soc.*, vol. 1, pp. 769–773, 1979.).

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

[20] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and state of the art," *Signal Processing*, vol. SP-19, no. 4, pp. 259–299, April 1990.

samples, it turns out that the successive a priori filtering errors can be computed efficiently by using a Schur type procedure. So, the new algorithm computes the same filtering errors as the FNTF algorithm even though the filter estimate is not updated at the sampling frequency. Using the FFT for computing some convolutions and updating the Kalman gain from its value L instants before in an efficient way gives us finally what we have called the Fast SU FNTF (FSU FNTF) algorithm which shows a reduced computational complexity, rendering it especially suited for adapting very long filters such as in the acoustic echo cancellation problem. Moreover, it appeared that the new algorithm has a more robust structure against roundoff error accumulation than the FNTF algorithm.

The rest of the paper 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 procedure that allows the computation of the filtering errors when the SUS is applied. 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. In the new algorithm, the Kalman gain has to be updated every L samples. In section 6, after showing how to do this efficiently, we give the FSU FNTF algorithm. 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 [5],[6],[16], 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 (see Fig. 2)

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

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

References

[1] E. Hänsler, "The hands-free telephone problem - An annotated bibliography," *Signal Processing*, pp. 259–271, 1992.

[2] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[3] O. Macchi, *The Least Mean Squares Approach with Applications in Transmission*, John Wiley & Sons, New York, 1995.

[4] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, 1996, third edition.

[5] J.M. Cioffi and T. Kailath, "Fast, recursive least squares transversal filters for adaptive filtering," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. ASSP-32, no. 2, pp. 304–337, April 1984.

[6] D.T.M. Slock and T. Kailath, "Numerically Stable Fast Transversal Filters for Recursive Least-Squares Adaptive Filtering," *IEEE Trans. Signal Proc.*, vol. SP-39, no. 1, pp. 92–114, Jan. 1991.

[7] E. Eleftheriou and D. Falconer, "Tracking Properties and Steady-State Performance of RLS Adaptive Filter Algorithms," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. ASSP-34, no. 5, pp. 1097–1110, Oct. 1986.

[8] G.V. Moustakides and S. Theodoridis, "Fast Newton Transversal Filters – A New Class of Adaptive Estimation Algorithms," *IEEE Trans. SP*, vol. SP-39, no. 10, pp. 2184–2193, Oct. 1991.

[9] T. Petillon, A. Gilloire, and S. Theodoridis, "The Fast Newton Transversal Filter: an Efficient Scheme for Acoustic Echo Cancellation in Mobile Radio," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. ASSP-42, no. 3, pp. 509–518, March 1994.

[10] K. Ozeki and T. Umeda, "An Adaptive Algorithm Using Orthogonal Projection to an Affine Subspace and its Properties," *Trans. IECE Japan*, vol. J67-A, pp. 126–132, 1984.

[11] S. L. Gay and Sanjeev Tavathia, "The Fast Affine Projection Algorithm," in *Proc. ICASSP Conf.*, Detroit, USA, May 9-12 1995, pp. 3023–3026.

criterion

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

where $\lambda \in (0, 1]$ is the exponential forgetting factor and $\|v\|^2 = v^H \, v$.

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 [17, 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). Equations (8)-(10) constitute the joint-process or filtering part of the RLS algorithm and takes $2N+1$ operations per sample. 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, leading to algorithms such as the FTF with computational complexity $7N$. We now recall the FNTF algorithm, from which our new algorithm will be derived.

block length L (constrained to be a power of 2) that gives the lowest complexity. Note that the FSU FNTF algorithm becomes less complex than the FNTF algorithm when $N > 270$ (in the case where $M = 16$) and that the complexity is decreasing w.r.t. N. These results are illustrated in Fig. 3. Note also that the computational complexity as a function of the subsampling factor is a convex function with a minimum given by the solution of the transcendental equation:

$$3\log 2L^2 + 2L + (2 - 5\log 2)(N+1) = 2(N+1)\log L\ . \tag{56}$$

7 Conclusions

We have derived a new algorithm that is mathematically equivalent to the FNTF algorithm. The FSU FNTF and the FNTF algorithms produce exactly the same filtering error signal and hence exhibit the same performance, which is close to that of the RLS algorithm, especially when the input signal is an AR process. The computational complexity of the FSU FNTF algorithm is smaller than that of the FNTF algorithm for large filter lengths at the cost of a relatively small 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 are promising for applications such as acoustic echo cancellation.

3 The FNTF Algorithm

In the FTF algorithm, the Kalman gain and the likelihood variable are computed in the prediction part of the algorithm. The update of the inverse sample covariance matrix in the RLS algorithm is replaced by the update of its generators [18] which happen to be the optimal forward and backward prediction filters and the Kalman gain of order N (plus the update of some scalar quantities). The FNTF algorithm is an approximation of the FTF algorithm. It uses the fact that for AR signals of order M, the inverse covariance matrix is a banded matrix of bandwidth $2M+1$. This fact allows the use of prediction filters of order M in the FTF scheme. The 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 padding 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 extrapolation of the Kalman gain $\tilde{C}_{N,M,k}$ 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] needed a certain amount of data storage while in [9], two FTF algorithm prediction parts are run in parallel in order to avoid this extra memory use. 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 in its numerically stabilized form is given in Table 1 where $A_{M,k}$ 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 prediction errors, $\tilde{C}_{M+1,k} = \left[\tilde{C}^0_{M+1,k} \cdots \tilde{C}^M_{M+1,k}\right]$ 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 roundoff errors in the prediction part [6]. The prediction part of the FNTF has a computational complexity of $12M$ and the joint-process part takes $2N$ operations. In what follows, we will deal uniquely with the single channel case. The extension to the multi-channel case is straightforward. The FNTF algorithm can also be described in the following way which emphasizes its rotational structure:

$$\tilde{P}_k \;=\; \Phi^p(k)\, P_{k-1}$$

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.

Finally, combining, the two FTF prediction part algorithms that allow the computation of the successive pseudo rotation matrices, the Schur-FNTF procedure which gives the output error filter of the FNTF algorithm, the efficient computation of $\tilde{U}_{N+1,L,k}$ and $\tilde{S}_{N+1,L,k}$ and the use of the FFT for the update of the filter estimate, we get the FSU FNTF algorithm which is summarized in Tab. 2. Its initialization is done as follows:

- For the SFTF that operates on $\{x(k)\}$:

$$\begin{aligned} &A_{M,0} = [\,1\;0\;\cdots\;0\,]\;,\; B_{M,0} = [\,0\;\cdots\;0\;1\,] \\ &\alpha_M(0) = \lambda^N\mu\;,\; \beta_M(0) = \lambda^{N-M}\mu \\ &\tilde{C}_{M,0} = [\,0\;\cdots\;0\,]\;,\; \gamma_M(0) = 1\;. \end{aligned} \tag{52}$$

- For the SFTF that operates on $\{x(k-N+M)\}$:

$$\begin{aligned} &A_{M,-N+M} = [\,1\;0\;\cdots\;0\,]\;,\; B_{M,-N+M} = [\,0\;\cdots\;0\;1\,] \\ &\alpha_M(-N+M) = \lambda^M\mu\;,\; \beta_M(-N+M) = \mu \\ &\tilde{C}_{M,-N+M} = [\,0\;\cdots\;0\,]\;,\; \gamma_M(-N+M) = 1\;. \end{aligned} \tag{53}$$

- For the extrapolation and filtering part:

$$\begin{aligned} &\gamma_{N,M}^{-1}(0) = 1\;\;,\;\; \tilde{C}_{N,M,0} = [\,0\;\cdots\;0\,] \\ &W_{N,0} = W_0\;\;. \end{aligned} \tag{54}$$

We do not provide a learning curve for the FSU FNTF algorithm since it would be identical to that of the FNTF algorithm. For performance comparisons of the FSU FNTF algorithm with other adaptive filtering algorithms, one should consult the references introducing the FNTF algorithm.

The computational complexity of the FSU FNTF algorithm is

$$C_{FSUFNTF} = 4\left(\frac{N+1}{L}+1\right)\frac{FFT(2L)}{L}+6\frac{N+1}{L}+6L+18M \tag{55}$$

operations per sample. One can see the computational complexity reduction that is achieved in the case where FFT is done via the split radix algorithm [20]($FFT(2m) = mlog_2(2m)$ real multiplications for real signals) in Tab. 3 where for each filter length N, we find through inspection the

$$\tilde{P}_{k^d} = \Phi^p(k^d)\, P_{k^d-1} \quad , \quad k^d = k - N + M$$

$$\begin{bmatrix} \begin{bmatrix} \tilde{C}_{N,M,k} & 0 \end{bmatrix} \\ \begin{bmatrix} W_{N,k} & 0 \end{bmatrix} \end{bmatrix} = \Phi^f(k) \begin{bmatrix} \begin{bmatrix} P_{k-1} & 0_{N-M} \end{bmatrix} \\ \begin{bmatrix} 0_{N-M} & P_{k^d-1} \end{bmatrix} \\ \begin{bmatrix} 0 & \tilde{C}_{N,M,k-1} \end{bmatrix} \\ \begin{bmatrix} W_{N,k-1} & 0 \end{bmatrix} \end{bmatrix} , \tag{11}$$

where

$$\tilde{P}_k = \begin{bmatrix} \begin{bmatrix} \tilde{C}_{M,k} & 0 \end{bmatrix} \\ A_{M,k} \\ B_{M,k} \end{bmatrix} , \; P_k = \begin{bmatrix} \begin{bmatrix} 0 & \tilde{C}_{M,k} \end{bmatrix} \\ A_{M,k} \\ B_{M,k} \end{bmatrix} , \tag{12}$$

$$\Phi^p(k) = \Phi_2^p(k)\, \Phi_1^p(k) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ r_M^1(k) & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \tilde{C}^0_{M+1,k} & -\tilde{C}^M_{M+1,k} \\ e_M(k) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{13}$$

$$\Phi^f(k) = \Phi_2^f(k)\Phi_1^f(k) = \begin{bmatrix} 1 & 0 \\ & \\ \epsilon_N(k) & 1 \end{bmatrix} \begin{bmatrix} 0 & \tilde{C}^0_{M+1,k} & 0 & 0 & 0 & -\tilde{C}^M_{M+1,k^d} & 1 & 0 \\ & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{14}$$

In order to reduce the amount of computations in the FNTF algorithm, we propose to apply the SUS. This idea comes from the fact that for relatively long adaptive filters, it is not necessary to update the filter at each new input sample because there is no significant change in the filter coefficients after one update. The SUS does not necessarily involve approximations and the key ingredient is that even though the adaptive filter is not updated all the time, it is possible to compute efficiently the filtering errors that would arise if the filter estimate were updated at the sampling 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 and efficient computations of certain quantities help to reduce the complexity of the SU algorithms and will give FSU algorithms. In the FNTF algorithm, the prediction part costs $12M$ operations which is not computationally demanding. So, this part is kept unchanged and the SUS is only applied to the filtering part of the FNTF. Hence, the objective is to compute at time k the extrapolated Kalman gain $\tilde{C}_{N,M,k}$ of order N and the filter $W_{N,k}$ from their values at time $k - L$. For instance, one can quite simply compute (see (25) of Tab. 1) the value of the filter estimate at

The recursions in (46) 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 (44)

$$\left[\begin{array}{cc} \widetilde{C}_{N,M,k} & 0 \end{array}\right] = \left[\begin{array}{cc} 0_L & \widetilde{C}^{0:N-L}_{N,M,k-L} \end{array}\right] + \left(\widetilde{S}_{N+1,L,k}\right)_L - \left(\widetilde{U}_{N+1,L,k}\right)_L \quad , \tag{47}$$

where $\left(\widetilde{S}_{N+1,L,k}\right)_L$ 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 $M+1$ positions. Hence, the update of the Kalman gain vector (47) needs $2M+1$ 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 decomposition of the Kalman gain matrix in (44), we split the product in (15) as follows

$$\underline{\epsilon}^H_{N,L,k} \left[\begin{array}{cc} \underline{\widetilde{C}}_{N,M,k} & 0 \end{array}\right] = \underline{\epsilon}^H_{N,L,k}\, \mathcal{T}\left(\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]\right) + \underline{\epsilon}^H_{N,L,k}\, \widetilde{S}_{N+1,L,k} - \underline{\epsilon}^H_{N,L,k}\, \widetilde{U}_{N+1,L,k} \ . \tag{48}$$

Since $\widetilde{S}_{N+1,L,k}$ and $\widetilde{U}_{N+1,L,k}$ are sparse matrices, the second and third products in (48) are computed by performing the required multiplications/additions. These products take respectively $(M+1)L + .5L(L-1)$ and $(M+1)L$ multiplications per L samples. The first matrix being Toeplitz, we can further reduce the $NL - .5L(L-1)$ multiplications per L samples which are needed for the computation of $\underline{\epsilon}^H_{N,L,k}\, \mathcal{T}\left(\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]\right)$ by using the FFT technique.

Consider the following decomposition of $\mathcal{T}\left(\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]\right)$ in N_L sub-matrices of order $L \times L$

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

then, the product becomes

$$\underline{\epsilon}^H_{N,L,k}\, \mathcal{T}\left(\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]\right) = \left[\begin{array}{ccc} \underline{\epsilon}^H_{N,L,k}\, \mathcal{T}^1_{L,L} & \cdots & \underline{\epsilon}^H_{N,L,k}\, \mathcal{T}^{N_L}_{L,L} \end{array}\right] \ . \tag{50}$$

Every sub-product in (50) is now computed as follows:

$$\begin{aligned} \underline{\epsilon}^H_{N,L,k}\, \mathcal{T}^j_{L,L} &= \left[\begin{array}{cc} 0_L & \underline{\epsilon}^H_{N,L,k} \end{array}\right] \overline{T}^i_{L,L} \left[\begin{array}{c} I_L \\ 0_{L\times L} \end{array}\right] \\ &= \left[\left(\left[\begin{array}{cc} 0_L & \underline{\epsilon}^H_{N,L,k} \end{array}\right] F_{2L}\right) \mathrm{diag}(\tau^{i\,H} F_{2L})\right] \frac{1}{2L} F^H_{2L} \left[\begin{array}{c} I_L \\ 0_{L\times L} \end{array}\right] \quad , \end{aligned} \tag{51}$$

where $\tau^{i\,H}$ is the first row of the $2L \times 2L$ right shift circulant matrix obtained by embedding $\mathcal{T}^i_{L,L}$ in the same manner as was done for the Toeplitz data matrix $X_{L,L,k-(i-1)L}$ in (35). As it is shown in (51), the product $\underline{\epsilon}^H_{N,L,k}\, \mathcal{T}\left(\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]\right)$ is done by adding L zeros to $\underline{\epsilon}^H_{N,L,k}$, computing the corresponding DFT, computing the DFT of $\tau^{i\,H}$ (there are N_L vectors like this), computing the product

time k from its value L instants before

$$[\, W_{N,k} \;\; 0 \,] = [\, W_{N,k-L} \;\; 0 \,] + \underline{\epsilon}^{H}_{N,L,k} \left[\, \underline{\tilde{C}}_{N,M,k} \;\; 0 \,\right] , \tag{15}$$

where

$$\underline{\epsilon}^{H}_{N,L,k} \;\; = \;\; [\, \epsilon_N(k-L+1) \;\cdots\; \epsilon_N(k) \,] \tag{16}$$

$$\underline{\tilde{C}}_{N,M,k} \;\; = \;\; \begin{bmatrix} \tilde{C}_{N,M,k-L+1} \\ \vdots \\ \tilde{C}_{N,M,k} \end{bmatrix} . \tag{17}$$

$\underline{\epsilon}^{H}_{N,L,k}$ is the vector of L successive a posteriori output errors and $\underline{\tilde{C}}_{N,M,k}$ is the $L \times N$ Kalman gain matrix where the rows are the L successive FNTF Kalman gains. First, a certain 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 different filters at these instants. We will see later on how to compute the Kalman gain matrix $\underline{\tilde{C}}_{N,M,k}$ without updating the Kalman gain vector at each input sample and moreover how to compute efficiently the vector-matrix product $\underline{\epsilon}^{H}_{N,L,k} \left[\, \underline{\tilde{C}}_{N,M,k} \;\; 0 \,\right]$.

4 The Schur-FNTF procedure

Let us introduce the negative of the filter output

$$\hat{d}^{p}_{N}(k) = d(k) - \epsilon^{p}_{N}(k) \;\;\; , \;\;\; \hat{d}_{N}(k) = d(k) - \epsilon_{N}(k) \;\; , \tag{18}$$

the reversed complex-conjugate regression vector

$$x_{N,k} = [\, x_{k-N+1} \; x_{k-N+2} \cdots \; x_k \,]^{H} \;\;\; , \tag{19}$$

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

$$X_{N+1,L,k} = \begin{bmatrix} X^{H}_{N+1}(k-L+1) \\ \vdots \\ X^{H}_{N+1}(k) \end{bmatrix} = [\, x_{L,k} \; x_{L,k-1} \cdots \; x_{L,k-N} \,] \;\; . \tag{20}$$

By replacing (42) into (17) for every row, we obtain the expression of the Kalman gain matrix $\underline{\tilde{C}}_{N,M,k}$ in term of the Kalman gain vector $\tilde{C}_{N,M,k-L}$

$$\begin{aligned}\left[\begin{array}{cc}\underline{\tilde{C}}_{N,M,k} & 0\end{array}\right] &= \left(\left[\begin{array}{cc}0 & \tilde{C}_{N,M,k-L}\end{array}\right] Z^{i-1}\right)_{i=1:L} + \left(\sum_{l=0}^{i-1}\left[\begin{array}{cc}\tilde{S}_{M+1,k_{l,i}} & 0_{N-M}\end{array}\right] Z^{l}\right)_{i=1:L} \\ &\quad - \left(\sum_{l=0}^{i-1}\left[\begin{array}{cc}0_{N-M} & \tilde{U}_{M+1,k^d_{l,i}}\end{array}\right] Z^{l}\right)_{i=1:L} \\ &= \mathcal{T}\left(\left[\begin{array}{cc}0 & \tilde{C}_{N,M,k-L}\end{array}\right]\right) + \tilde{S}_{N+1,L,k} - \tilde{U}_{N+1,L,k} \ , \end{aligned} \tag{44}$$

the notation $(a_i)_{i=1:L}$ stands for the matrix with L rows in which the i^{th} row is a_i. The first term of the right hand side of (44) is a $L \times (N+1)$ Toeplitz matrix whose first row is $\left[\begin{array}{cc}0 & \tilde{C}_{N,M,k-L}\end{array}\right]$ and first column filled with zeros. The two other terms are $L \times (N+1)$ sparse matrices with the following structures:

$$\begin{aligned}\tilde{S}_{N+1,L,k} &= \begin{bmatrix} * & \cdots & * & 0 & \cdots & \cdots & 0 \\ \vdots & & \vdots & \ddots & \ddots & & \vdots \\ * & \cdots & * & \cdots & * & 0 & \cdots & 0 \end{bmatrix} \\ \tilde{U}_{N+1,L,k} &= \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \\ \vdots & & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & * & \cdots & * \end{bmatrix} \ , \end{aligned} \tag{45}$$

the $*$ standing for nonzero elements. First rows of $\tilde{S}_{N+1,L,k}$ and $\tilde{U}_{N+1,L,k}$ have $M+1$ nonzero elements and last row of $\tilde{S}_{N+1,L,k}$ has $M+L$ nonzero elements. Denoting by $(A)_i$ the i^{th} row of the matrix A, it can be easily shown using their definitions in (44) that the matrices $\tilde{S}_{N+1,L,k}$ and $\tilde{U}_{N+1,L,k}$ are efficiently computed by applying the following recursions on their rows

$$\begin{aligned}\left(\tilde{S}_{N+1,L,k}\right)_1 &= \left[\begin{array}{cc}\tilde{S}_{M+1,k-L+1} & 0_{N-M}\end{array}\right] \\ \left(\tilde{U}_{N+1,L,k}\right)_1 &= \left[\begin{array}{cc}0_{N-M} & \tilde{U}_{M+1,k^d-L+1}\end{array}\right] \end{aligned}$$

For $i = 2, \ldots, L$

$$\begin{aligned}\left(\tilde{S}_{N+1,L,k}\right)_i &= \left(\tilde{S}_{N+1,L,k}\right)_{i-1} Z + \left[\begin{array}{cc}\tilde{S}_{M+1,k-L+i} & 0_{N-M}\end{array}\right] \\ \left(\tilde{U}_{N+1,L,k}\right)_i &= \left(\tilde{U}_{N+1,L,k}\right)_{i-1} Z + \left[\begin{array}{cc}0_{N-M} & \tilde{U}_{M+1,k^d-L+i}\end{array}\right] \end{aligned} \tag{46}$$

End For

Now, consider the following set of filtering operations

$$F_L(k) \triangleq \begin{bmatrix} g_L^H(k) \\ g_L^H(k^d) \\ \eta_{N,L,k}^H \\ -\widehat{d}_{N,L,k}^{p\,H} \end{bmatrix} \triangleq \begin{bmatrix} [\, P_{k-L}\ 0_{N-M}\,] \\ [\, 0_{N-M}\ P_{k^d-L}\,] \\ \left[\, 0\ \widetilde{C}_{N,M,k-L}\,\right] \\ [\, W_{N,k-L}\ 0\,] \end{bmatrix} X_{N+1,L,k}^H \quad , \quad g_L^H(k) \triangleq \begin{bmatrix} \eta_{M,L,k}^H \\ e_{M,L,k}^{p\,H} \\ r_{M,L,k}^{pf\,H} \end{bmatrix} . \tag{21}$$

$F_L(k)$ 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

$$\widehat{d}_{N,L,k}^{p} = d_{L,k} - \epsilon_{N,L,k}^{p} = \begin{bmatrix} d^H(k-L+1) \\ \vdots \\ d^H(k) \end{bmatrix} - \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-L) \end{bmatrix} = \begin{bmatrix} \widehat{d}_N^H(k-L+1|k-L) \\ \vdots \\ \widehat{d}_N^H(k|k-L) \end{bmatrix} . \tag{22}$$

The first column of $F_L(k)$ is

$$F_L(k)\, u_{L,1} = \begin{bmatrix} p(k) \\ p(k^d) \\ 1-\gamma_N^{f^{-1}}(k-L) \\ -\widehat{d}_N^{p}(k-L+1) \end{bmatrix} \quad , \quad p(k) = \begin{bmatrix} 1-\gamma_M^{-1}(k-L) \\ e_M^{p}(k-L+1) \\ r_M^{pf}(k-L+1) \end{bmatrix} \quad , \tag{23}$$

where $u_{L,n}$ is the $L \times 1$ vector with 1 in the n^{th} position and 0 elsewhere.

One can see immediately that since $\widehat{d}_N^{p}(k-L+1)$ is available (last component of the first column of $F_L(k)$), one can compute the a priori filtering error $\epsilon_N^{p}(k-L+1)$ with (22) then, the a posteriori filtering error $\epsilon_N(k-L+1)$ using $\gamma_{N,M}(k-L+1)$ (see (20) and (24) in Tab. 1).

The updating scheme of the FNTF algorithm can be written as

$$\begin{bmatrix} \left[\, \widetilde{P}_k\ 0_{N-M}\,\right] \\ \left[\, 0_{N-M}\ \widetilde{P}_{k^d}\,\right] \\ \left[\, \widetilde{C}_{N,M,k}\ 0\,\right] \\ [\, W_{N,k}\ 0\,] \end{bmatrix} = \Theta_k \begin{bmatrix} [\, P_{k-1}\ 0_{N-M}\,] \\ [\, 0_{N-M}\ P_{k^d-1}\,] \\ \left[\, 0\ \widetilde{C}_{N,M,k-1}\,\right] \\ [\, W_{N,k-1}\ 0\,] \end{bmatrix} , \tag{24}$$

where Θ_k is a 8×8 matrix given by

$$\Theta_k = \begin{bmatrix} I_3 & 0 & 0 \\ 0 & I_3 & 0 \\ 0 & 0 & \Phi_2^f(k) \end{bmatrix} \begin{bmatrix} \Phi_2^p(k) & 0 & 0 \\ 0 & \Phi_2^p(k^d) & 0 \\ 0 & 0 & I_2 \end{bmatrix} \begin{bmatrix} \Phi_1^p(k) & 0 & 0 \\ 0 & \Phi_1^p(k^d) & 0 \\ \cdots & \cdots & \cdots \\ & \Phi_1^f(k) & \end{bmatrix} . \tag{25}$$

Hence, the computation of the vector in (36) requires the padding of v with L zeros, the DFT of the resulting vector, the DFT of $x_{2L,k-(i-1)L}$, 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 $x_{2L,k-iL}, i = 1, \ldots, M-1$ have been computed at previous time instants. The above procedure will only be used for the product of $\left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-L} \end{array}\right]$ 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

$$2N\left[\frac{\mathrm{FFT}(2L)}{L^2} + \frac{2}{L}\right] + \frac{3\mathrm{FFT}(2L)}{L} + 6M \tag{40}$$

computations per sample (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

In order to compute the Kalman gain matrix that appears when the filter estimate $W_{N,k}$ is computed from its value at time $k-L$ (see (15)), we come back to the FNTF Kalman gain update

$$\left[\begin{array}{cc} \widetilde{C}_{N,M,k} & 0 \end{array}\right] = \left[\begin{array}{cc} 0 & \widetilde{C}_{N,M,k-1} \end{array}\right] + \left[\begin{array}{cc} \widetilde{S}_{M+1,k} & 0_{N-M} \end{array}\right] - \left[\begin{array}{cc} 0_{N-M} & \widetilde{U}_{M+1,k^d} \end{array}\right] , \tag{41}$$

where definitions of $\widetilde{S}_{M+1,k}$ and $\widetilde{U}_{M+1,k^d}$ given in (3) and (13) of Tab.1. From the above equation, it is easy to see that each row of the Kalman gain matrix can be expressed as follows

$$\begin{array}{rcl} \textbf{For} \quad i & = & 1, \ldots, L \;\; , \;\; k_{l,i} = k-L+i-l \;\; , \;\; k^d_{l,i} = k^d-L+i-l \\ \left[\begin{array}{cc} \widetilde{C}_{N,M,k-L+i} & 0 \end{array}\right] & = & \left[\begin{array}{cc} 0_i & \widetilde{C}^{0:N-i}_{N,M,k-L} \end{array}\right] + \sum_{l=0}^{i-1}\left[\begin{array}{cc} \widetilde{S}_{M+1,k_{l,i}} & 0_{N-M} \end{array}\right] Z^l - \sum_{l=0}^{i-1}\left[\begin{array}{cc} 0_{N-M} & \widetilde{U}_{M+1,k^d_{l,i}} \end{array}\right] Z^l \\ \textbf{End For} & & \end{array} \tag{42}$$

with the $(N+1) \times (N+1)$ right shift matrix

$$Z = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} . \tag{43}$$

Hence, if we rotate both expressions for $F_L(k)$ in (21) with Θ_{k-L+1}, we obtain $\Theta_{k-L+1} F_L(k)$ which equals

$$\begin{bmatrix} \begin{bmatrix} \widetilde{P}_{k-L+1} & 0_{N-M} \end{bmatrix} \\ \begin{bmatrix} 0_{N-M} & \widetilde{P}_{k^d-L+1} \end{bmatrix} \\ \begin{bmatrix} \widetilde{C}_{N,M,k-L+1} & 0 \end{bmatrix} \\ \begin{bmatrix} W_{N,k-L+1} & 0 \end{bmatrix} \end{bmatrix} X^H_{N+1,L,k} = \begin{bmatrix} q_L(k) & \\ q_L(k^d) & \\ \boxed{\eta^H_{N,L-1,k}} & * \\ -\widehat{d}_N(k-L+1) & \boxed{-\widehat{d}^{\,p\,H}_{N,L-1,k}} \end{bmatrix} \tag{26}$$

$$q_L(k) = \begin{bmatrix} \boxed{\eta^H_{M,L-1,k}} & * \\ e_M(k-L+1) & \boxed{e^{p\,H}_{M,L-1,k}} \\ r^f_M(k-L+1) & \boxed{r^{pf\,H}_{M,L-1,k}} \end{bmatrix} . \tag{27}$$

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

$$\mathcal{S}\left(\Theta_{k-L+1}\, F_L(k)\right) \;=\; F_{L-1}(k)\,, \tag{28}$$

where the operator $\mathcal{S}(\mathcal{M})$ stands for: shift the first, the fourth and the seventh rows of the matrix $\mathcal{M}$ 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 $k-L+i$, $i = 1, \ldots, L$, 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 filtering errors can be computed over the data block without updating the filter estimate. Now, since it is possible to compute the parameters of the successive matrices Θ_{k-L+i}, $i = 1, \ldots, L$, it suffices to accumulate them and apply the resulting matrix to the filters in order to update them. This is what we have called the Schur-FNTF procedure. However, we will not update the filters using this procedure because the computational complexity of the prediction part is already relatively reduced. Hence, the procedure (28) is only used for computing the a priori filtering error over the current data block and the updates of the filters and scalars in the prediction

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

$$v_{N+1,k} = \left[\begin{array}{ccc} v_{L,k}^1 & \cdots & v_{L,k}^{N_L} \end{array} \right] \ , \tag{32}$$

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

$$X_{N+1,L,k} = \left[\begin{array}{cccc} X_{L,L,k} & X_{L,L,k-L} & \cdots & X_{L,L,k-N+L-1} \end{array} \right] \ , \tag{33}$$

then

$$v_{N+1,k} \ X_{N+1,L,k}^H = \sum_{i=1}^{N_L} v_{L,k}^i X_{L,L,k-(i-1)L}^H \ . \tag{34}$$

In other words, we have essentially N_L times the product of a vector of length L with a $L \times L$ Toeplitz matrix. Such a product can be efficiently computed in basically two different ways [19]. 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. In this case, the $L \times L$ Toeplitz matrix $X_{L,L,k}$ is embedded into a $2L \times 2L$ circulant matrix, viz.

$$\underline{X}_{L,L,k}^H \ = \ \left[\begin{array}{cc} * & X_{L,L,k}^H \\ & \\ X_{L,L,k}^H & * \end{array} \right] \ = \ \mathcal{C}\left(x_{2L,k}^H \right) \ , \tag{35}$$

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

$$v_{L,k}^i X_{L,L,k-(i-1)L}^H \ = \ \left[\begin{array}{cc} 0_{1\times L} & v_{L,k}^i \end{array} \right] \mathcal{C}\left(x_{2L,k-(i-1)L}^H \right) \left[\begin{array}{c} I_L \\ 0_{L\times L} \end{array} \right] \ . \tag{36}$$

Now consider the Discrete Fourier Transform (DFT) $\mathcal{V}_{L,k}^i$ of $v_{L,k}^i$

$$\mathcal{V}_{L,k}^i = v_{L,k}^i \ F_L \ , \tag{37}$$

F_L is the $L \times L$ DFT matrix whose generic element is $(F_L)_{p,q} = e^{-j2\pi \frac{(p-1)(q-1)}{L}}$, $j^2 = -1$.
The inverse of F_L is $\frac{1}{L} F_L^H$. It defines the inverse DFT (IDFT)

$$v_{L,k}^i = \mathcal{V}_{L,k}^i \ \frac{1}{L} F_L^H \ . \tag{38}$$

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

$$v \ \mathcal{C}(c^H) \ = \ v \ F_m \ \mathrm{diag}\left(c^H F_m \right) \frac{1}{m} F_m^H \ = \ \left[(v F_m) \ \ \mathrm{diag}\left(c^H F_m \right) \right] \ \frac{1}{m} F_m^H \ . \tag{39}$$

part are done by running the prediction part of the FNTF algorithm. Noticing that the Schur Procedure inherently provides the successive a priori forward and backward prediction errors $e_M^p(k)$ and $r_M^{pf}(k)$, we can remove the computation of these quantities from the prediction part of the FNTF algorithm. This reduces the computational complexity of the prediction part of the FNTF to $8M$ instead of $12M$ operations per sample.

A remarkable property of the Schur-FNTF procedure is the removal of the long-term roundoff error instability due to the recursive computation of the likelihood variable $\gamma_{N,M}(k)$. The recursions are interrupted since the likelihood variable $\gamma_N^{f^{-1}}(k)$ is computed at each new block of data via an inner product (23). In Fig. 4, we give the evolution of $\gamma_{N,M}^{-1}(k) - \gamma_N^{-1}(k)$ where $\gamma_N^{-1}(k)$ is computed as:

$$\gamma_N^{-1}(k) = 1 + X_N^H(k) R_{N,k-1}^{-1} X_N(k) \ , \tag{29}$$

and $R_{N,k-1}^{-1}$ is the inverse sample covariance matrix associated with the FNTF algorithm and computed recursively as:

$$\begin{aligned} \begin{bmatrix} R_{N,k}^{-1} & 0 \\ 0 & 0 \end{bmatrix} = & \begin{bmatrix} 0 & 0 \\ 0 & R_{N,k-1}^{-1} \end{bmatrix} + \alpha_M^{-1}(k-1) \begin{bmatrix} A_{M,k}^H \\ 0_{N-M}^T \end{bmatrix} \begin{bmatrix} A_{M,k}^H \\ 0_{N-M}^T \end{bmatrix}^H \\ & -\beta_M^{-1}(k^d-1) \begin{bmatrix} 0_{N-M}^T \\ B_{M,k^d-1}^H \end{bmatrix} \begin{bmatrix} 0_{N-M}^T \\ B_{M,k^d-1}^H \end{bmatrix}^H \end{aligned} \tag{30}$$

with $R_{N,0}^{-1} = \mu^{-1}\text{diag}\left(\lambda^{-N}, \ldots, \lambda^{-1}\right)$, $\text{diag}(w)$ being a diagonal matrix with the elements of the vector w as the diagonal elements. The quantity $\gamma_{N,M}^{-1}(k) - \gamma_N^{-1}(k)$ represents the deviation due to the accumulation of roundoff errors of the likelihood variable $\gamma_{N,M}^{-1}(k)$ computed recursively in the FNTF algorithm from the likelihood variable $\gamma_N^{-1}(k)$ computed in (29) (within an infinite precision computation environment this deviation will be equal to zero). The parameters were set to: $N = 25, M = 10, \lambda = .99, \mu = .1$ and the input was a white noise with variance equal to 1. The simulation has been done under Matlab where the relative accuracy of numbers is 2.22×10^{-16} (approximately 16 significant decimal digits on computers using IEEE floating point arithmetic). As it can be seen from Fig. 4, $\gamma_{N,M}^{-1}(k) - \gamma_N^{-1}(k)$ is linearly increasing with time (slope of $.2 \times 10^{-16}$ per sample in this case). Even though, this difference grows very slowly, it will lead to a long-term numerical divergence of the FNTF algorithm. This numerical explosion would have been more

noticeable within a smaller precision environment (a fixed point implementation with 16 bits for instance). Note that this phenomenon has been analyzed to be a random walk [6]. The corresponding stabilization has been done by simply removing the recursions in the likelihood variable computation (see (19) of Tab.(1)).

In Fig. 5, we give the evolution of $\gamma_N^{f^{-1}}(k) - \gamma_N^{-1}(k)$, where:

$$\gamma_N^{f^{-1}}(k) = 1 - \tilde{C}_{N,M,k} X_N(k) \; . \tag{31}$$

There, the roundoff error signal stays constant at the level of the Matlab accuracy over one million of white noise input samples. With the likelihood variable computed in (31), the random walk instability due to accumulation of roundoff errors has been removed. Note that the likelihood variable computed in the FSU FNTF algorithm via the Schur procedure corresponds to $\gamma_N^{f^{-1}}(k)$ only at the subsampling instants. In fact, there is a relatively very small deviation of the likelihood variable computed in the FSU FNTF algorithm from $\gamma_N^{f^{-1}}(k)$ between the subsampling instants because of the dynamic of the roundoff error accumulation in the Schur procedure. Nevertheless, this deviation in the data block is corrected at every subsampling instant and hence, has no effect on the stability of the computation of the likelihood variable.

This fact has a big importance for the real-time implementation of the algorithm.

Taking into account that Θ_k in its factorized form (25) has 11 non-trivial elements, the Schur-FNTF procedure as given by (28) takes only $5.5L$ multiplications per sample. Inner products that represent filtering operations are needed for the initialization (computation of $F_L(k)$ in (21)). At this point, the Schur-FNTF procedure is computationally demanding because the products in (21) require $\mathcal{O}(NL)$ multiplications per L incoming samples. We now consider the FFT technique to reduce this 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 [19]. In what follows, we shall often assume for reasons of simplification that L is a power of two and that $N_L = (N+1)/L$ is an integer. To get $F_L(k)$ in (21), 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.