The Fast Subsampled-Updating Recursive Least-Squares (FSU RLS) Algorithm for Adaptive Filtering Based on Displacement Structure and the FFT[†]

Dirk T.M. Slock and Karim Maouche

Institut Eurecom, B.P. 193, 06904 Sophia Antipolis Cedex, France.

Please send all correspondence to:

Dirk Slock
Institut Eurecom
2229 route des Crêtes
B.P. 193
F-06904 Sophia Antipolis Cedex
France

slock@eurecom.fr

[†]This work was supported in part by the CNET, France, under contract nr 93 6 B 005. This work has also benefited from cooperations within the French group of signal processing researchers GDR 134 TdSI.

Number of pages: 33

Number of Tables: 1

Number of Figures: 1

Proposed running headline: Fast Subsampled-Updating Recursive Least-Squares

Keywords: Recursive Least-Squares, fast algorithms, displacement structure, Fast Fourier Transform, acoustic echo cancellation.

The Fast Subsampled-Updating Recursive Least-Squares (FSU RLS) Algorithm for Adaptive Filtering Based on Displacement Structure and the FFT

Dirk T.M. Slock and Karim Maouche

Abstract

In this paper, we derive a new fast algorithm for Recursive Least-Squares (RLS) adaptive filtering. This algorithm is especially suited for adapting very long filters such as in the acoustic echo cancellation problem. The starting point is to introduce subsampled updating (SU) in the RLS algorithm. In the SU RLS algorithm, the Kalman gain and the likelihood variable are matrices. Due to the shift invariance of the adaptive FIR filtering problem, these matrices exhibit a low displacement rank. This leads to a representation of these quantities in terms of sums of products of triangular Toeplitz matrices. Finally, the product of these Toeplitz matrices with a vector can be computed efficiently by using the Fast Fourier Transform (FFT).

Zusammenfassung

Dieser Artikel beschreibt die Herleitung eines neuen Algorithmus zur schnellen adaptiven Recursive Least Square (RLS) Filterung . Dieser Algorithmus eignet sich besonders füer aufwendige Filter, wie sie zum Beispiel zur akkustischen Echounterdrüeckung benutzt werden. Im Zentrum dieses Algorithmus steht die Einfüehrung von unterabgetastetem Updating (SU). Der Kalman Gewinn und die Likelihood Variable treten im SU RLS Algorithmus als Matrizen auf. Aufgrund der Verschiebungsinvarianz in der adaptiven FIR Filterung zeigen diese Matrizen einen niedrigen Verschiebungsrang. Dies füehrt zu einer Darstellung dieser Gröessen als Summe von Produkten von trianguläeren Toeplitz Matrizen. Das Produkt dieser Matrizen mit einem Vektor kann auf sehr effiziente Weise mit der Fast Fourier Transform (FFT) berechnet werden.

Résumé

Dans ce papier, nous présentons un nouvel algorithme des moindres carrés récursif rapide. Cet algorithme présente un intérêt certain pour l'adaptation de filtres très longs comme ceux utilisés dans les problèmes d'annulation d'écho acoustique. L'idée de départ est d'utiliser l'algorithme RLS avec une mise à jour "sous-échantillonnée" du filtre. Dans cet algorithme (le SU RLS) le gain de Kalman et la variable de vraisemblance sont des matrices qui ont des rangs de déplacement faibles. Ces quantités sont alors représentées et mises à jour par le biais de leurs générateurs, sous forme de sommes de produits de matrices de Toeplitz triangulaires. Le produit de l'une de ces quantités avec un vecteur peut alors être calculé en utilisant la transformée de Fourier rapide (FFT).

1 Introduction

Fast RLS algorithms such as the Fast Transversal Filter (FTF) algorithm [4],[11],[12] and the Fast Lattice/Fast QR (FLA/FQR) algorithms [8] efficiently exploit the shift invariance structure present in the RLS approach to the adaptive FIR filtering problem. They reduce the computational complexity of $\mathcal{O}(N^2)$ for the conventional RLS algorithm to $\mathcal{O}(N)$ operations per sample. In order to further reduce the computational complexity of these algorithms, it appears that the sampling rate at which the LS filter estimate is provided has to be reduced from the signal sampling rate to a subsampled rate with a subsampling factor of $L \geq 1$. Two strategies emerge in order to accomplish this. One consists of a block processing approach in which the normal equations governing the LS problem are solved every L samples. This leads to Block RLS (BRLS) algorithms. An alternative approach (especially applicable when $L < N$) consists of using the same strategy as the RLS algorithm and to compute the new filter estimate and auxiliary quantities from the same quantities that were available L samples before. We shall call this the Subsampled-Updating RLS (SU RLS) algorithm. Below, we shall consider both approaches in detail, but we shall especially focus on a fast version of the SU RLS algorithm, the FSU RLS algorithm.

2 The Subsampled-Updating RLS Algorithm

In order to formulate the RLS adaptive filtering problem and to fix notation, we shall first recall the RLS algorithm. We shall mostly stick to the notation introduced in [4],[5],[11],[12], 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.

Suitable position for Figure 1

2.1 The RLS Algorithm

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

The resulting error signal is given by (see Fig. 1)

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

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

$$\begin{aligned} \xi_N(k) \;&=\; \min_{W_N} \left\{ \sum_{i=1}^{k} \lambda^{k-i} \left\| d(i) + W_N\, X_N(i) \right\|^2 \;+\; \lambda^{k+1}\mu \left\| W_N - W_0 \right\|^2_{\Lambda_N} \right\} \\ &=\; \sum_{i=1}^{k} \lambda^{k-i} \left\| \epsilon_N(i|k) \right\|^2 \;+\; \lambda^{k+1}\mu \left\| W_{N,k} - W_0 \right\|^2_{\Lambda_N} \end{aligned} \tag{2}$$

where $\lambda \in (0,1]$ is the exponential weighting factor, $\mu > 0$, $\Lambda_N = \operatorname{diag}\left\{\lambda^{N-1},\ldots,\lambda,1\right\}$, $\|v\|^2_\Lambda = v\Lambda v^H$, $\|.\| = \|.\|_I$. The second term in the LS criterion represents a priori information. For instance, prior to measuring the signals, we may assume that W_N is distributed as $W_N \sim \mathcal{N}\left(W_0, R_0^{-1}\right)$, $R_0 = \mu\lambda\Lambda_N$ (or any other distribution with the same first and second order moments). The particular choice for R_0 will become clear in the discussion of the initialization of the FSU RLS algorithm. Minimization of the LS criterion leads to the following minimizer

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

where

$$\begin{aligned} R_{N,k} \;&=\; \sum_{i=1}^{k} \lambda^{k-i} X_N(i) X_N^H(i) \;+\; \lambda^{k+1}\mu\Lambda_N \\ &=\; \lambda R_{N,k-1} + X_N(k) X_N^H(k)\;, \qquad\qquad R_{N,0} = R_0 = \mu\lambda\Lambda_N \\ P_{N,k} \;&=\; \sum_{i=1}^{k} \lambda^{k-i} X_N(i) d^H(i) \;-\; \lambda^{k+1}\mu\Lambda_N W_0^H \\ &=\; \lambda P_{N,k-1} + X_N(k) d^H(k)\;, \qquad\qquad P_{N,0} = -R_0 W_0^H \end{aligned} \tag{4}$$

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:

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

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

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

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

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

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

where $\epsilon^p_N(k)$ and $\epsilon_N(k)$ are the a priori and a posteriori error signals (resp. predicted and filtered errors in the Kalman filtering terminology) and one can verify (or see [4]) that they are related by the likelihood variable $\gamma_N(k)$ as in (9). The overnormalized Kalman gain $\widetilde{C}_{N,k}$ is related to the unnormalized Kalman gain $C_{N,k}$:

$$C_{N,k} \;=\; -X^H_N(k)R^{-1}_{N,k} \;=\; \gamma_N(k)\widetilde{C}_{N,k} \tag{11}$$

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

and the term overnormalized stems from the relation $\widetilde{C}_{N,k} = \gamma^{-1}_N(k)C_{N,k}$. Using the recursions for $W_{N,k}$, $P_{N,k}$, one can verify that the minimum value for the LS criterion satisfies the recursion

$$\xi_N(k) \;=\; \xi_0(k) + W_{N,k}P_{N,k} \;=\; \lambda\xi_N(k{-}1) + \epsilon^p_N(k)\gamma_N(k)\epsilon^{pH}_N(k)\;. \tag{13}$$

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 $\widetilde{C}_{N,k}$ 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 (FTF and FLA/FQR) exploit a certain shift invariance structure in $X_N(k)$ which is inherited by $R_{N,k}$ and $P_{N,k}$, to avoid the Riccati equation in the prediction part and reduce its computational complexity vto $\mathcal{O}(N)$ (the FLA/FQR algorithms also provide $\epsilon^p_N(k)$ but replace $W_{N,k}$ by a transformed set of parameters as in the square-root Kalman filtering/RLS algorithms). We now investigate alternative ways to reduce the computational complexity of the RLS algorithm.

2.2 The Block RLS Algorithm

One way to reduce computational complexity is to not compute the LS filter estimate every sample, but only once every L samples. In the block processing algorithms of [3],[16], the LS solution is in fact not computed recursively in time, but directly from the normal equations (see (3))

$$R_{N,k} W^H_{N,k} \;=\; -P_{N,k}\;. \tag{14}$$

In the prewindowed problem ($x(i) = 0, i \leq 0$), the sample covariance matrix $R_{N,k}$ has a displacement structure (is close to a Toeplitz matrix [7]) with displacement rank equal to 3. Therefore a fast (generalized Levinson) algorithm may be applied to solve (14) in $4N^2 + \mathcal{O}(N)$ operations (see [3, Table V]). Due to its displacement structure, $R_{N,k}$ can be described by only two vectors, namely

$$r_{N,k} \;=\; R_{N,k} u_{N,1} \;=\; \lambda r_{N,k-1} + X_N(k) x^H(k) \tag{15}$$

the first column of $R_{N,k}$ ($u_{n,m}$ is a unit vector of length n with a 1 in the mth position and zeros elsewhere), and $X_{N-1}(k)$. So, the only quantities that are computed recursively in time are the correlation vectors $P_{N,k}$ and $r_{N,k}$, and at times equal to integer multiples of the block length L, the normal equations (14) are solved. In fact, when $\lambda < 1$, instead of using the recursion indicated in (15), the correlation vectors are best computed as

$$P_{N,k} \;=\; \lambda^L P_{N,k-L} + \sum_{i=k-L+1}^{k} \lambda^{k-i} X_N(i) d^H(i) \tag{16}$$

$$r_{N,k} \;=\; \lambda^L r_{N,k-L} + \sum_{i=k-L+1}^{k} \lambda^{k-i} X_N(i) x^H(i) \tag{17}$$

which leads to $2(L+1)N$ operations. So the most significant terms of the computational complexity of the fast BRLS algorithm are $2N + 4\frac{N^2}{L}$ per sample (compared to $2N$ for the LMS algorithm or $7N$ for the minimal FTF algorithm). Although for computational complexity reasons, one is inclined to take L very large (larger than N in fact), such a strategy would be less interesting for two reasons: processing delay (inherent in a block processing strategy) and loss of tracking capability of the adaptive filter. Therefore, we shall henceforth assume $L \leq N$. We may also note that so far, we have not considered the computation of the filtering error (also [3],[16] neglect this issue), which would in a straightforward implementation add N operations per sample.

Using so-called *doubling* techniques, it is possible to further reduce the complexity of the generalized Levinson algorithm. In [1] a doubling algorithm is derived for a general displacement rank δ. Such an algorithm allows for the solution of the system (14) in $\mathcal{O}\left((\delta-1)N(\log_2 N)^2\right)$ operations.

2.3 Fast Computation of the Second Order Statistics using the FFT

It is possible to reduce the computational complexity of the fast BRLS algorithm further by introducing FFT techniques as explained in [16]. In what follows, we shall often assume for simplicity that L is a power of two and that $M = N/L$ is an integer, though more general cases can be considered equally well. We shall introduce the following notation. Let

$$d_{L,k} = \begin{bmatrix} d^H(k-L+1) \\ \vdots \\ d^H(k) \end{bmatrix}, \; x_{L,k} = \begin{bmatrix} x^H(k-L+1) \\ \vdots \\ x^H(k) \end{bmatrix}, \; X_{N,L,k} = \begin{bmatrix} X_N^H(k-L+1) \\ \vdots \\ X_N^H(k) \end{bmatrix} = \left[x_{L,k} \cdots x_{L,k-N+1}\right]. \tag{18}$$

We can now rewrite (16), (17) as

$$P_{N,k} \;=\; \lambda^L P_{N,k-L} + X_{N,L,k}^H \Lambda_L d_{L,k} \;, \quad r_{N,k} \;=\; \lambda^L r_{N,k-L} + X_{N,L,k}^H \Lambda_L x_{L,k} \;. \tag{19}$$

Continuing with $P_{N,k}$ (for $r_{N,k}$, just replace $d_{L,k}$ by $x_{L,k}$), consider a partitioning in $M = N/L$ subvectors of length L:

$$P_{N,k} = \left[\underline{P}_{N,k}^{1\,H} \cdots \underline{P}_{N,k}^{M\,H}\right]^H \tag{20}$$

then (19) reduces for subvector j to

$$\underline{P}_{N,k}^{j} \;=\; \lambda^L \underline{P}_{N,k-L}^{j} + X_{L,L,k-(j-1)L}^H \Lambda_L d_{L,k} \;, \quad j = 1, \ldots, M \;. \tag{21}$$

In other words, we have essentially $2M$ times $2L$ multiplications (for e.g. $\lambda^L \underline{P}_{N,k-L}^{j}$ and $\Lambda_L d_{L,k}$) plus $2M$ times the product of a $L \times L$ Toeplitz matrix with a vector of length L. Such a product can be efficiently computed in basically two different ways. One way is to use fast convolution algorithms [15], 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.

$$\underline{X}^H_{L,L,k} = \begin{bmatrix} * & X^H_{L,L,k} \\ X^H_{L,L,k} & * \end{bmatrix} = \mathcal{C}\left(x^H_{2L,k}\right) \tag{22}$$

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

$$X^H_{L,L,k-(j-1)L}\Lambda_L d_{L,k} = [I_L \;\; 0_{L\times L}]\; \mathcal{C}\left(x^H_{2L,k-(j-1)L}\right) \begin{bmatrix} 0_{L\times 1} \\ \Lambda_L d_{L,k} \end{bmatrix}. \tag{23}$$

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

$$\mathcal{C}(c^H)\, v = \frac{1}{m}\mathcal{C}(c^H)\, F_m^H F_m v = \frac{1}{m} F_m^H \;\mathrm{diag}^H\left(F_m\, c\right)\; F_m v \tag{24}$$

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 (23) requires L multiplications to form the product $\frac{1}{2L}\Lambda_L d_{L,k}$, the padding of the resulting vector 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+L 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 reduces the $2N(1+\frac{1}{L})$ computations per sample for $P_{N,k}$ and $r_{N,k}$ via (16),(17) to

$$2N\left[\frac{2\,\mathrm{FFT}(2L)}{L^2} + \frac{4}{L}\right] + \frac{\mathrm{FFT}(2L)}{L} \tag{25}$$

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.

2.4 Fast Computation of the Filtering Errors using the FFT

In a block processing approach, also the filtering errors can be computed more efficiently than by computing an inner product every sample period. Indeed, consider the following vector of (block) a priori filtering errors

$$\epsilon^{p}_{N,L,k} = \begin{bmatrix} \epsilon^{H}_{N}(k-L+1|k-L) \\ \vdots \\ \epsilon^{H}_{N}(k|k-L) \end{bmatrix} = d_{L,k} + X_{N,L,k} W^{H}_{N,k-L} = d_{L,k} + \sum_{j=1}^{M} X_{L,L,k-(j-1)L} \underline{W}^{j\,H}_{N,k-L} \tag{26}$$

with $W_{N,k} = \left[\underline{W}^{1}_{N,k} \cdots \underline{W}^{M}_{N,k}\right]$. We can use the same circular matrix embedding and DFT techniques as in the previous subsection. Remark that the input data have been transformed before (in the previous subsection) and that we only need to apply the inverse DFT once after having summed up the M products in the frequency domain. This leads to a computational complexity of

$$N\left[\frac{\mathrm{FFT}(2L)}{L^2} + \frac{2}{L}\right] + \frac{\mathrm{FFT}(2L)}{L} \tag{27}$$

per sample instead of N.

2.5 The SU RLS Algorithm

Instead of computing the filter $W_{N,k}$ from scratch every L samples by solving the normal equations (14), we may want to exploit information gathered at the previous solution instant, $k-L$. If we plug in the recursions (19) for $P_{N,k}$ and $R_{N,k}$ into the solution (3), then we get similarly to the derivation of the RLS algorithm the following recursion

$$\begin{aligned}
\underline{\widetilde{C}}_{N,k} &= -X_{N,L,k}\lambda^{-L} R^{-1}_{N,k-L} && (28)\\
\underline{\gamma}^{-1}_{N}(k) &= \Lambda^{-1}_{L} - X_{N,L,k}\underline{\widetilde{C}}^{H}_{N,k} && (29)\\
R^{-1}_{N,k} &= \lambda^{-L} R^{-1}_{N,k-L} - \underline{\widetilde{C}}^{H}_{N,k}\underline{\gamma}_{N}(k)\underline{\widetilde{C}}_{N,k} && (30)\\
\epsilon^{p}_{N,L,k} &= d_{L,k} + X_{N,L,k} W^{H}_{N,k-L} && (31)\\
\underline{\gamma}^{-1}_{N}(k)\,\epsilon_{N,L,k} &= \epsilon^{p}_{N,L,k} && (32)\\
W_{N,k} &= W_{N,k-L} + \epsilon^{H}_{N,L,k}\underline{\widetilde{C}}_{N,k} && (33)
\end{aligned}$$

where $\epsilon_{N,L,k}$ is a vector of a posteriori errors:

$$\epsilon_{N,L,k} = \begin{bmatrix} \epsilon_N^H(k-L+1|k) \\ \vdots \\ \epsilon_N^H(k|k) \end{bmatrix} . \tag{34}$$

In this case, the corresponding unnormalized Kalman gain would be

$$\underline{C}_{N,k} = -\Lambda_L X_{N,L,k} R_{N,k}^{-1} = \underline{\gamma}_N(k)\underline{\widetilde{C}}_{N,k} \tag{35}$$

$$\underline{\gamma}_N(k) = \Lambda_L \left(I_L + X_{N,L,k} \underline{C}_{N,k}^H \right) . \tag{36}$$

Using the recursions for $W_{N,k}$, $P_{N,k}$, one can again verify that the minimum value for the LS criterion satisfies the recursion

$$\xi_N(k) = \xi_0(k) + W_{N,k} P_{N,k} = \lambda^L \xi_N(k-L) + \epsilon_{N,L,k}^{p\,H} \underline{\gamma}_N(k) \epsilon_{N,L,k}^p . \tag{37}$$

While the Subsampled-Updating RLS algorithm thus obtained constitutes a valid algorithm to provide the filter solution $W_{N,k}$ every L samples, it does not represent much computational gain w.r.t. the original RLS algorithm ($L = 1$). We could exploit the FFT technique introduced above to reduce the computational complexity in equations (28),(29) and (31) by a factor $\mathcal{O}\left(\frac{L}{\log_2 L}\right)$. On the other hand, we have to invert $\underline{\gamma}_N^{-1}(k)$, a $L \times L$ matrix. Below, we shall introduce a fast version of the SU RLS algorithm.

2.6 Relation Between the Filtering Errors in Block Mode and in Sequential Mode

Remark that in the SU RLS algorithm, we find filtering errors that are not just predicted one step ahead, but several steps. This results from the fact that the filter $W_{N,k}$ gets updated only once every L samples. The learning curve for the SU RLS algorithm would be the variance of the filtering errors obtained from $\epsilon_{N,L,k}$ and hence would be piecewise constant, coinciding with the learning curve for the RLS algorithm at times that are integer multiples of L, and remaining constant for $L-1$ samples after those instants. However, it turns out to be fairly simple to recover the a priori filtering errors of the conventional RLS algorithm from those of the SU RLS algorithm.

By substituting

$$W_{N,i-1} \;=\; W_{N,k-L} + \sum_{j=k-L+1}^{i-1} \epsilon_N(j|j-1)\,\gamma_N(j)\,\widetilde{C}_{N,j} \qquad (38)$$

into

$$\epsilon_N(i|i-1) \;=\; d(i) + W_{N,i-1}X_N(i) \qquad (39)$$

we get

$$\epsilon_N(i|k-L) \;=\; \epsilon_N(i|i-1) - \sum_{j=k-L+1}^{i-1} \epsilon_N(j|j-1)\,\gamma_N(j)\,\widetilde{C}_{N,j}X_N(i) \;\;,\;\; i > k-L\,. \qquad (40)$$

Hence, we can relate the a priori filtering errors in block mode and in sequential mode as follows

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ * & \cdots & 1 \end{bmatrix} \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-L) \end{bmatrix} = \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-1) \end{bmatrix}. \qquad (41)$$

The lower triangular factor can be identified as follows. By comparing the recursion (37) for the minimal cost with L iterations of recursion (13) for the same quantity, we can identify

$$\epsilon^{p\,H}_{N,L,k}\underline{\gamma}_N(k)\epsilon^{p}_{N,L,k} \;=\; \sum_{i=0}^{L-1} \lambda^i \epsilon^p_N(k-i)\gamma_N(k-i)\epsilon^{pH}_N(k-i)\,. \qquad (42)$$

Let us introduce the following notation

$$\underline{\epsilon}^p_{N,k} \;=\; \begin{bmatrix} \epsilon_N^{p\,H}(k-L+1) \\ \vdots \\ \epsilon_N^{p\,H}(k) \end{bmatrix} = \begin{bmatrix} \epsilon_N^H(k-L+1|k-L) \\ \vdots \\ \epsilon_N^H(k|k-1) \end{bmatrix} \qquad (43)$$

$$,_{N,L,k} \;=\; \mathrm{diag}\left\{\gamma_N(k-L+1),\ldots,\gamma_N(k)\right\} \qquad (44)$$

$$D_{N,L,k} \;=\; \Lambda_L\,,_{N,L,k} \qquad (45)$$

then we can rewrite (42) as

$$\epsilon^{p\,H}_{N,L,k}\underline{\gamma}_N(k)\epsilon^{p}_{N,L,k} \;=\; \underline{\epsilon}^{p\,H}_{N,k}D_{N,L,k}\underline{\epsilon}^p_{N,k}\,. \qquad (46)$$

Now consider the Upper Diagonal Lower (UDL) triangular factorization of the $L \times L$ matrix $\underline{\gamma}_N(k)$, then we get

$$\underline{\gamma}_N(k) \;=\; U_{N,L,k}\,D_{N,L,k}\,U^H_{N,L,k} \qquad (47)$$

where $U_{N,L,k}$ is upper triangular with unit diagonal. By uniqueness of the triangular factorization, $U^H_{N,L,k}$ is the lower triangular factor in (41) while $D_{N,L,k}$ is indeed the diagonal matrix introduced in (45). So we can rewrite (41) as

$$U^H_{N,L,k}\epsilon^p_{N,L,k} \;=\; \underline{\epsilon}^p_{N,k}\,. \tag{48}$$

This relation allows us to compute the a priori filtering errors $\underline{\epsilon}^p_{N,k}$ of the RLS algorithm from the a priori filtering errors $\epsilon^p_{N,L,k}$ in the SU RLS algorithm. The necessary triangular factorization (47) can easily be made part of the inversion of $\underline{\gamma}_N^{-1}(k)$ in the SU RLS algorithm. One can now also easily show

$$\underline{\gamma}_N(k)\,\epsilon^p_{N,L,k} \;=\; \underbrace{U_{N,L,k}\Lambda_L\,,_{N,L}(k)\,\underbrace{U^H_{N,L,k}\,\underbrace{\underbrace{\epsilon^p_{N,L,k}}_{}}_{\text{a priori SURLS errors}}}_{\underline{\epsilon}^p_{N,k}\text{ a priori RLS errors}}}_{\underline{\epsilon}_{N,k}\text{ a posteriori RLS errors}}\;_{\epsilon_{N,L,k}\text{ a posteriori SURLS errors}}\,. \tag{49}$$

2.7 Relation Between the Kalman Gain in Block Mode and in Sequential Mode

There exists a relation between the Kalman gain in the SU RLS algorithm and L consecutive Kalman gains of the RLS algorithm, similar to the relation we found in the previous subsection for the filtering errors. By repeated application of the relation (7), we get

$$R^{-1}_{N,i-1} \;=\; \lambda^{-i+k-L+1}R^{-1}_{N,k-L} - \sum_{j=k-L+1}^{i-1}\lambda^{-i+1+j}\widetilde{C}^H_{N,j}\gamma_N(j)\widetilde{C}_{N,j}\;,\quad i>k{-}L \tag{50}$$

which, using (5), leads to

$$\widetilde{C}_{N,i} \;=\; -\lambda^{-i+k-L}X^H_N(i)R^{-1}_{N,k-L} + \sum_{j=k-L+1}^{i-1}\left(\lambda^{j-i}X^H_N(i)\widetilde{C}^H_{N,j}\gamma_N(j)\right)\widetilde{C}_{N,j}\;,\quad i>k{-}L\,. \tag{51}$$

Putting the relations (51) together for $i=k{-}L{+}1,\ldots,k$ gives

$$\Lambda_L\,\underline{\widetilde{C}}_{N,k} \;=\; \begin{bmatrix}1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ * & \cdots & 1\end{bmatrix}\begin{bmatrix}\widetilde{C}_{N,k-L+1}\\ \vdots\\ \widetilde{C}_{N,k}\end{bmatrix} \tag{52}$$

which can also be rewritten as

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ * & \cdots & 1 \end{bmatrix} \underline{\widetilde{C}}_{N,k} \;=\; \Lambda_L^{-1} \begin{bmatrix} \widetilde{C}_{N,k-L+1} \\ \vdots \\ \widetilde{C}_{N,k} \end{bmatrix} . \tag{53}$$

The lower triangular factor can be identified as follows. By using (7) repeatedly and (30), we get

$$\begin{aligned} \underline{\widetilde{C}}^H_{N,k}\underline{\gamma}_N(k)\underline{\widetilde{C}}_{N,k} &= -R^{-1}_{N,k} + \lambda^{-L} R^{-1}_{N,k-L} \\ &= -\sum_{i=0}^{L-1} \lambda^{-i} \left(R^{-1}_{N,k-i} - \lambda^{-1} R^{-1}_{N,k-i-1}\right) \\ &= \sum_{i=0}^{L-1} \lambda^{-i} \widetilde{C}^H_{N,k-i}\gamma_N(k-i)\widetilde{C}_{N,k-i} \end{aligned} \tag{54}$$

or hence

$$\underline{\widetilde{C}}^H_{N,k}\underline{\gamma}_N(k)\underline{\widetilde{C}}_{N,k} \;=\; \begin{bmatrix} \widetilde{C}_{N,k-L+1} \\ \vdots \\ \widetilde{C}_{N,k} \end{bmatrix}^H \Lambda_L^{-1} D_{N,L,k} \Lambda_L^{-1} \begin{bmatrix} \widetilde{C}_{N,k-L+1} \\ \vdots \\ \widetilde{C}_{N,k} \end{bmatrix} . \tag{55}$$

Using the UDL factorization of $\underline{\gamma}_N(k)$ in (47) and by uniqueness of the triangular factorization, putting (53) and(55) together leads to

$$U^H_{N,L,k}\underline{\widetilde{C}}_{N,k} \;=\; \Lambda_L^{-1} \begin{bmatrix} \widetilde{C}_{N,k-L+1} \\ \vdots \\ \widetilde{C}_{N,k} \end{bmatrix} . \tag{56}$$

Let $u_{N,L,k}$ be the last column of $U_{N,L,k}$. Then (56) leads in particular to

$$u^H_{L,1}\,\underline{\widetilde{C}}_{N,k} \;=\; \lambda^{-L+1}\widetilde{C}_{N,k-L+1} \tag{57}$$

$$u^H_{N,L,k}\,\underline{\widetilde{C}}_{N,k} \;=\; \widetilde{C}_{N,k} \;. \tag{58}$$

3 Displacement Structure of the SU RLS Kalman Gain Quantities

Consider the displacement structure of a matrix R:

$$\nabla_\lambda R \;=\; R - \lambda\, Z\, R\, Z^H \;=\; \sum_{i=1}^{\delta} u_i\, v_i^H \tag{59}$$

where δ is the rank of $R - \lambda Z R Z^H$ and Z is the lower shift matrix (ones on the first subdiagonal and zeros elsewhere). By solving this Lyapunov equation, it is straightforward to obtain the following representation for R:

$$R \;=\; \nabla_\lambda^{-1}\left(\sum_{i=1}^{\delta} u_i\, v_i^H\right) \;=\; \sum_{i=1}^{\delta}\sum_{j=0}^{\infty} \lambda^j\, Z^j\, u_i\, v_i^H\, (Z^H)^j \;=\; \sum_{i=1}^{\delta} \mathcal{L}(u_i)\, \widetilde{\Lambda}\, \mathcal{L}^H(v_i) \tag{60}$$

where $\widetilde{\Lambda} =$ diag$\{1,\ \lambda,\ \lambda^2, \ldots\}$ and $\mathcal{L}(u)$ is a lower triangular Toeplitz matrix with u as first column. We shall exploit this representation for $\underline{\widetilde{C}}_{N,k}$ and $\underline{\gamma}_N^{-1}(k)$ to reduce the computational complexity of the SU RLS algorithm. Considering the definition of these quantities, we see that we first have to consider $R_{N,k}^{-1}$.

3.1 Displacement Structure of the Inverse Sample Covariance Matrix

In [9], the following displacement structure was derived

$$\nabla_\lambda \begin{bmatrix} R_{N,k}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \;=\; A_{N,k}^H \alpha_N^{-1}(k) A_{N,k} - B_{N,k}^H \beta_N^{-1}(k) B_{N,k} + \lambda \begin{bmatrix} 0 & \widetilde{C}_{N,k} \end{bmatrix}^H \gamma_N(k) \begin{bmatrix} 0 & \widetilde{C}_{N,k} \end{bmatrix} \tag{61}$$

where $A_{N,k}$ and $B_{N,k}$ are forward and backward prediction filters and $\alpha_N(k)$ and $\beta_N(k)$ are forward and backward prediction error variances (see [4]).

3.2 Displacement Structure of the Kalman Gain

Equation (28), can be rewritten as

$$\begin{bmatrix} \underline{\widetilde{C}}_{N,k} & 0 \end{bmatrix} = -X_{N+1,L,k}\, \lambda^{-L} \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix}. \tag{62}$$

Applying the ∇_λ operator to (62) yields:

$$\nabla_\lambda \begin{bmatrix} \underline{\widetilde{C}}_{N,k} & 0 \end{bmatrix} = -\lambda^{-L} \left(X_{N+1,L,k} \nabla_\lambda \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \lambda \Delta X_{N+1,L,k} \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix} Z_{N+1}^H \right) \tag{63}$$

with $\Delta X_{N+1,L,k}$ the $L \times (N+1)$ matrix given by

$$\Delta X_{N+1,L,k} = X_{N+1,L,k} Z_{N+1} - Z_L X_{N+1,L,k} = \begin{bmatrix} X_N^H(k-L) & 0 \\ 0 & -x_{L-1,k-N-1} \end{bmatrix} . \tag{64}$$

Let $e^p_{N,L,k}$ and $r^p_{N,L,k}$ be respectively the vectors of forward and backward a priori prediction errors defined by

$$e^p_{N,L,k} = X_{N+1,L,k}\, A^H_{N,k-L} \tag{65}$$

$$r^p_{N,L,k} = X_{N+1,L,k}\, B^H_{N,k-L} \,. \tag{66}$$

By substituting (61), (64) into (63), and using the notation defined in (65), (66) we get

$$\begin{aligned} \nabla_\lambda \left[\underline{\widetilde{C}}_{N,k} \quad 0\right] &= -e^p_{N,L,k}\lambda^{-L}\alpha_N^{-1}(k-L)A_{N,k-L} + r^p_{N,L,k}\lambda^{-L}\beta_N^{-1}(k-L)B_{N,k-L} \\ &\quad - \left(\eta_{N,L,k} - u_{L,1}\right)\lambda^{-L+1}\gamma_N(k-L)\left[0 \quad \widetilde{C}_{N,k-L}\right] \end{aligned} \tag{67}$$

where

$$\eta_{N,L,k} = X_{N+1,L,k}\left[0 \quad \widetilde{C}_{N,k-L}\right]^H \,. \tag{68}$$

Thus, using (60) we can rewrite the Kalman gain in the form

$$\begin{aligned} \left[\underline{\widetilde{C}}_{N,k} \quad 0\right] &= -\lambda^{-L}\alpha_N^{-1}(k-L)\mathcal{L}\left(e^p_{N,L,k}\right)\, \widetilde{\Lambda}_L\, \mathcal{L}^H\left(A_{N,k-L}\right) \\ &\quad +\lambda^{-L}\beta_N^{-1}(k-L)\mathcal{L}\left(r^p_{N,L,k}\right)\, \widetilde{\Lambda}_L\, \mathcal{L}^H\left(B_{N,k-L}\right) \\ &\quad -\lambda^{-L+1}\gamma_N(k-L)\mathcal{L}\left(\eta_{N,L,k} - u_{L,1}\right)\, \widetilde{\Lambda}_L\, \mathcal{L}^H\left(\left[0 \quad \widetilde{C}_{N,k-L}\right]\right) \,. \end{aligned} \tag{69}$$

By using (58), the $L \times 1$ vector $\eta_{N,L,k}$ introduced above can also be expressed in terms of the Kalman gain at time $k-1$ as

$$\eta_{N,L,k} = \lambda^{L-1} X_{N+1,L,k}\left[0 \ \ \underline{\widetilde{C}}_{N,k-1}\right]^H u_{L,1} = \lambda^{L-1} X_{N,L,k-1}\underline{\widetilde{C}}^H_{N,k-1} u_{L,1} \tag{70}$$

which leads to

$$\eta_{N,L,k} = \lambda^{L-1}\left(\Lambda_L^{-1} - \underline{\gamma}_N^{-1}(k-1)\right) u_{L,1} \,. \tag{71}$$

3.3 Displacement Structure of the Likelihood Variable

Consider now the displacement structure of the likelihood variable $\underline{\gamma}_N^{-1}(k)$

$$\nabla_\lambda \, \underline{\gamma}_N^{-1}(k) = \nabla_\lambda \, \Lambda_L^{-1} + \lambda^{-L} \, \nabla_\lambda \left(X_{N,L,k} R_{N,k-L}^{-1} X_{N,L,k}^H \right) \; . \tag{72}$$

Straightforwardly, one has

$$\nabla_\lambda \, \Lambda_L^{-1} = \lambda^{-L+1} u_{L,1} u_{L,1}^H \; . \tag{73}$$

Now, the second term of the right hand side of (72) gives

$$\begin{aligned}
\nabla_\lambda \left(X_{N,L,k} R_{N,k-L}^{-1} X_{N,L,k}^H \right) \; &= \; \nabla_\lambda \left(X_{N+1,L,k} \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix} X_{N+1,L,k}^H \right) \qquad (74) \\
= \; & \left(\nabla_\lambda \left(X_{N+1,L,k} \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \right) \right) X_{N+1,L,k}^H + \lambda Z_L X_{N+1,L,k} \begin{bmatrix} R_{N,k-L}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \left(\Delta X_{N+1,L,k} \right)^H \\
= \; & -\lambda^L \left(\nabla_\lambda \; \left[\underline{\tilde{C}}_{N,k-L} \;\; 0 \right] \right) X_{N+1,L,k}^H - \lambda \gamma_N(k-L) Z_L X_{N+1,L,k} \left[\tilde{C}_{N,k-L} \;\; 0 \right]^H u_{L,1}^H \\
= \; & -\lambda^L \left(\nabla_\lambda \; \left[\underline{\tilde{C}}_{N,k-L} \;\; 0 \right] \right) X_{N+1,L,k}^H - \lambda \gamma_N(k-L) \left(I - u_{L,1} u_{L,1}^H \right) \, \eta_{N,L,k} u_{L,1}^H \; .
\end{aligned}$$

By using the displacement structure of the Kalman gain given in (67), the displacement structure of the likelihood variable becomes

$$\begin{aligned}
\nabla_\lambda \, \underline{\gamma}_N^{-1}(k) \;\; &= \;\; e_{N,L,k}^p \lambda^{-L} \alpha_N^{-1}(k-L) e_{N,L,k}^{p\,H} - r_{N,L,k}^p \lambda^{-L} \beta_N^{-1}(k-L) r_{N,L,k}^{p\,H} \\
& \quad + \left(\eta_{N,L,k} - u_{L,1} \right) \lambda^{-L+1} \gamma_N(k-L) \left(\eta_{N,L,k} - u_{L,1} \right)^H \\
& \quad + u_{L,1} \lambda^{-L+1} \left(1 + \gamma_N(k-L) u_{L,1}^H \eta_{N,L,k} - \gamma_N(k-L) \right) u_{L,1}^H \; .
\end{aligned} \tag{75}$$

¿From (71) and the UDL decomposition of $\underline{\gamma}_N(k)$ in (45), (47), one can see that the first element of $\eta_{N,L,k}$ is

$$\begin{aligned}
u_{L,1}^H \eta_{N,L,k} \;\; &= \;\; \lambda^{L-1} \left(\lambda^{-L+1} - \lambda^{-L+1} \gamma_N^{-1}(k-L) \right) \\
&= \;\; 1 - \gamma_N^{-1}(k-L) \; .
\end{aligned} \tag{76}$$

Finally, replacing (76) in (75) gives the expression of the displacement structure of the likelihood variable in terms of three generators as

$$\begin{aligned}
\nabla_\lambda \, \underline{\gamma}_N^{-1}(k) \;\; &= \;\; e_{N,L,k}^p \lambda^{-L} \alpha_N^{-1}(k-L) e_{N,L,k}^{p\,H} - r_{N,L,k}^p \lambda^{-L} \beta_N^{-1}(k-L) r_{N,L,k}^{p\,H} \\
& \quad + \left(\eta_{N,L,k} - u_{L,1} \right) \lambda^{-L+1} \gamma_N(k-L) \left(\eta_{N,L,k} - u_{L,1} \right)^H \; .
\end{aligned} \tag{77}$$

This last equation exhibits the Hermitian structure inherited from $\underline{\gamma}_N^{-1}(k)$.

Using (60) the likelihood variable can be written as

$$\begin{aligned}\underline{\gamma}_N^{-1}(k) &= \lambda^{-L}\alpha_N^{-1}(k-L)\mathcal{L}\left(e^p_{N,L,k}\right)\,\tilde{\Lambda}_L\,\mathcal{L}^H\left(e^p_{N,L,k}\right)\\ &- \lambda^{-L}\beta_N^{-1}(k-L)\mathcal{L}\left(r^p_{N,L,k}\right)\,\tilde{\Lambda}_L\,\mathcal{L}^H\left(r^p_{N,L,k}\right)\\ &+ \lambda^{-L+1}\gamma_N(k-L)\mathcal{L}\left(\eta_{N,L,k}-u_{L,1}\right)\,\tilde{\Lambda}_L\,\mathcal{L}^H\left(\eta_{N,L,k}-u_{L,1}\right)\ .\end{aligned} \tag{78}$$

Because of the shift invariance of the adaptive filtering problem, the Kalman gain and the likelihood variable have a low displacement rank of 3; that is, these matrices have a structure close to the Toeplitz one and can be replaced by their representation as in (60). The appearence of Toeplitz matrices in this representation allows for an efficient computation of the product of the Kalman gain matrix with an a posteriori error vector by using the FFT. Also, the inversion of the likelihood variable matrix can be done efficiently by using the generalized Schur algorithm. Furthermore, instead of updating $\underline{\tilde{C}}_{N,k}$ and $\underline{\gamma}_N^{-1}(k)$ by using the SU RLS equations (28), (29), it suffices to update the filters $A_{N,k}$, $B_{N,k}$ and $\left[0\ \ \tilde{C}_{N,k}\right]$, and to compute the filter outputs $e^p_{N,L,k}$, $r^p_{N,L,k}$ and $\eta_{N,L,k}$ from their definitions (65), (66) and (68), using the data available at the time instant k.

This updating of the prediction filters and Kalman gain is the subject of the next section.

4 The FSU RLS Algorithm

4.1 Update of the Joint-Process Filter

We can rewrite equation (33) in the form

$$[W_{N,k}\ \ 0] = [W_{N,k-L}\ \ 0] + \epsilon^H_{N,L,k}\left[\underline{\tilde{C}}_{N,k}\ \ 0\right]\ . \tag{79}$$

The reason why we add the zeros is that for the FSU RLS algorithm, it will turn out to be more convenient to assume that $M = \frac{N+1}{L}$ is an integer. The a posteriori filtering error vector $\epsilon_{N,L,k}$ is obtained by resolving the system of equations (32). This is discussed further in section 4.4.

Now, if we replace $\left[\underline{\tilde{C}}_{N,k}\ \ 0\right]$ in (79) by its expression given in (69), the joint-process equation takes the form

$$[W_{N,k}\ \ 0] = [W_{N,k-L}\ \ 0] + \epsilon^H_{N,L,k} \sum_{i=1}^{3} d_i \mathcal{T}^i_{L,L}\, \widetilde{\Lambda}_L\, \mathcal{G}^i_{L,N+1} \tag{80}$$

where d_i, $\mathcal{T}^i_{L,L}$ and $\mathcal{G}^i_{L,N+1}$ are respectively the constants, the $L \times L$ lower and $L \times (N+1)$ upper triangular Toeplitz matrices described in section 3.2.

The last term in equation (80) can be computed in the following manner. For $i = 1,2,3$ do:

- mutiply d_i by $\epsilon_{N,L,k}$ to obtain $p^1_{L,i} = \epsilon^H_{N,L,k} d_i$.
- use the circular embedding and FFT technique to compute the product $p^2_{L,i} = p^1_{L,i}\mathcal{T}^i_{L,L}$.
- multiply $p^2_{L,i}$ with the diagonal matrix $\widetilde{\Lambda}_L$. This gives a $1 \times L$ vector, say $p^3_{L,i} = p^2_{L,i}\widetilde{\Lambda}_L$.
- use again the circular embedding and FFT technique to compute the product $p^4_{L,i} = p^3_{L,i}\mathcal{G}^i_{L,N+1}$ in $\frac{N+1}{L}$ portions of length L (see sections 2.3 and 2.4).

Finally, add the three vectors $p^4_{L,i}$ and obtain $\epsilon^H_{N,L,k}\sum_{i=1}^{3} d_i \mathcal{T}^i_{L,L}\, \widetilde{\Lambda}_L\, \mathcal{G}^i_{L,N+1}$ by applying the inverse FFT to the sum $\sum_{i=1}^{3} p^4_{L,i}$.

4.2 Update of the RLS Kalman Gain $\widetilde{C}_{N,k}$

Equation (58) can be rewritten as

$$\left[\widetilde{C}_{N,k}\ 0\right] = u^H_{N,L,k}\ \left[\underline{\widetilde{C}}_{N,k}\ 0\right]. \tag{81}$$

The computation of this product can be carried out in exactly the same manner as the computation of $\epsilon^H_{N,L,k}\ \left[\underline{\widetilde{C}}_{N,k}\ 0\right]$ in the previous subsection. Note however, that the Fourier transform of the generators needs to be computed only once. This leads to the update of the RLS Kalman gain $\left[\widetilde{C}_{N,k}\ 0\right]$ at time k by representing the SU RLS Kalman gain $\left[\underline{\widetilde{C}}_{N,k}\ 0\right]$ in terms of its generators which comprise the RLS Kalman gain $\left[\widetilde{C}_{N,k-L}\ 0\right]$ at the previous iteration.

As we saw, $u_{N,L,k}$ is the last column of $U_{N,L,k}$, which is the $L \times L$ upper triangular matrix appearing in the UDL decomposition of $\underline{\gamma}_N(k)$ (see (47)). It is possible to obtain $u_{N,L,k}$ from $\underline{\gamma}^{-1}_N(k)$. This will be elaborated upon in section 4.4.

4.3 Update of the Prediction Filters

In a way that parallels the update of the joint-process filter in the SU RLS algorithm, it is possible to derive the following updates for the prediction filters and the associated prediction error variances

$$\begin{aligned} B_{N,k} &= B_{N,k-L} + r^H_{N,L,k} \left[\underline{\tilde{C}}_{N,k} \;\; 0\right] && (82) \\ \beta_N(k) &= \lambda^L \beta_N(k-L) + r^H_{N,L,k}\, r^p_{N,L,k} && (83) \\ A_{N,k} &= A_{N,k-L} + e^H_{N,L,k} \left[0 \;\; \underline{\tilde{C}}_{N,k-1}\right] && (84) \\ \alpha_N(k) &= \lambda^L \alpha_N(k-L) + e^H_{N,L,k}\, e^p_{N,L,k} && (85) \end{aligned}$$

where $e_{N,L,k}$ and $r_{N,L,k}$ are respectively the a posteriori forward and backward prediction error vectors.

The update equation (82) of the backward prediction filter $B_{N,k}$ has the same form as equation (79) of the joint-process filter and is computed similarly. This is not the case for the forward part in equation (84) since $\underline{\tilde{C}}_{N,k-1}$ at time $k-1$ is needed.

To avoid the use of $\underline{\tilde{C}}_{N,k-1}$, one must update $A_{N,k+1}$ first and then compute $A_{N,k}$ from $A_{N,k+1}$. This leads to

$$\begin{aligned} A_{N,k+1} &= A_{N,k-L+1} + e^H_{N,L,k+1} \left[0 \;\; \underline{\tilde{C}}_{N,k}\right] && (86) \\ A_{N,k} &= A_{N,k+1} - e_N(k+1) \left[0 \;\; \tilde{C}_{N,k}\right] && (87) \end{aligned}$$

where $e_N(k+1)$ is the a posteriori forward prediction error at time $k+1$, and similarly for the forward prediction error variance

$$\begin{aligned} \alpha_N(k+1) &= \lambda^L \alpha_N(k-L+1) + e^H_{N,L,k+1}\, e^p_{N,L,k+1} && (88) \\ \alpha_N(k) &= \lambda^{-1} \left(\alpha_N(k+1) - e_N(k+1)\, {e^p_N}^H(k+1)\right) \; . && (89) \end{aligned}$$

In order to compute (86) efficiently, one can rewrite it as

$$A_{N,k+1} = A_{N,k-L+1} + e^H_{N,L,k+1} \left[\underline{\tilde{C}}_{N,k} \;\; 0\right] Z^H_{N+1} \; . \qquad (90)$$

The a posteriori forward prediction error $e_N(k+1)$ can easily be obtained from $e^p_N(k+1)$ by using the well-known relation

$$e_N(k+1) = \gamma_N(k) e^p_N(k+1) \; . \qquad (91)$$

$\gamma_N(k)$ is available after the computation of the likelihood variable $\underline{\gamma}_N^{-1}(k)$ as the inverse of the last element of the diagonal matrix that appears in its LDU decomposition (see (46)).

Now the a priori forward prediction error $e_N^p(k{+}1)$ can be obtained from the a priori forward prediction error vector $e^p_{N,L,k+1}$ by using the same relation as the one between the filtering error in block and sequential mode (48), viz.

$$U^H_{N,L,k}\, e^p_{N,L,k+1} = \underline{e}^p_{N,L,k+1} \ , \tag{92}$$

$\underline{e}^p_{N,L,k}$ being the $L \times 1$ vector of sequential a priori forward prediction errors. Equating the last element on both sides of this relation yields

$$e_N^{p\,H}(k{+}1) = u^H_{N,L,k}\, e^p_{N,L,k+1} \ . \tag{93}$$

Equations (86), (88) involve $e^p_{N,L,k+1}$. On the other hand, the generators of the Kalman gain and the likelihood variable require $e^p_{N,L,k}$. In order to avoid computing both by inner product, we shall use a relation between the vectors $e^p_{N,L,k}$ and $e^p_{N,L,k+1}$. Therefore, consider the vector $v_{N,L+1,k}$ defined by

$$v_{N,L+1,k+1} = X_{N+1,L+1,k+1}\, A^H_{N,k-L+1} \ . \tag{94}$$

It is easy to see that

$$v_{N,L+1,k+1} = \begin{bmatrix} X^H_{N+1}(k{-}L{+}1) A^H_{N,k-L+1} \\ X_{N+1,L,k+1} A^H_{N,k-L+1} \end{bmatrix} = \begin{bmatrix} e^H_N(k{-}L{+}1) \\ e^p_{N,L,k+1} \end{bmatrix} \ . \tag{95}$$

Now (87) taken at time $k{-}L$ can be rewritten as

$$A_{N,k-L+1} \;=\; A_{N,k-L} + e_N(k{-}L{+}1)\left[0 \;\; \tilde{C}_{N,k-L}\right] \ . \tag{96}$$

When we replace $A_{N,k-L+1}$ in equation (94) by the expression above, then $v_{N,L+1,k+1}$ becomes

$$v_{N,L+1,k+1} = \begin{bmatrix} e^p_{N,L,k} \\ e^H_N(k{+}1|k{-}L) \end{bmatrix} + \begin{bmatrix} \eta_{N,L,k} \\ X^H_N(k)\, \tilde{C}^H_{N,k-L} \end{bmatrix} e^H_N(k{-}L{+}1) \tag{97}$$

where $e^p_{N,L,k}$ is computed like $\epsilon_{N,L,k}$ in section 2.4, $e_N(k{+}1|k{-}L) = X^H_{N+1}(k{+}1)A^H_{N,k-L}$ and $\tilde{C}_{N,k-L}X_N(k)$ are computed by explicitly calculating the inner product, $\eta_{N,L,k}$ is computed by calculating the inner product in (68), and $e_N(k{-}L{+}1)$ is available from the previous adaptation (see (91) at time $k{-}L{+}1$). By equating (95) and (97), $e^p_{N,L,k+1}$ can be obtained from $e^p_{N,L,k}$ as

$$e^p_{N,L,k+1} = \begin{bmatrix} \left(e^p_{N,L,k}\right)_{2:L} \\ e^H_N(k{+}1|k{-}L) \end{bmatrix} + \begin{bmatrix} (\eta_{N,L,k})_{2:L} \\ X^H_N(k)\,\tilde{C}^H_{N,k-L} \end{bmatrix} e^H_N(k{-}L{+}1)\ . \tag{98}$$

4.4 Triangular Factorization of the Likelihood Variable

The previous sections showed that $u_{N,L,k}$ and $\gamma_N(k)$ are needed in the process of updating the prediction filters. These quantities can be obtained from the UDL factorization of $\underline{\gamma}_N(k)$. They can also be obtained from the LDU factorization of $\underline{\gamma}^{-1}_N(k)$ whose three generators $e^p_{N,L,k}$, $r^p_{N,L,k}$ and $\eta_{N,L,k}{-}u_{L,1}$ (and some scalars) are available. We also need $\underline{\gamma}_N(k)$ or $\underline{\gamma}^{-1}_N(k)$ to obtain the a posteriori error vectors from the a priori ones.

Consider the LDU decomposition of $\underline{\gamma}^{-1}_N(k)$

$$\underline{\gamma}^{-1}_N(k) = L_{N,L,k}\, G_{N,L,k}\, L^H_{N,L,k}\ . \tag{99}$$

Inverting both sides in this equation gives

$$\underline{\gamma}_N(k) = L^{-H}_{N,L,k}\, G^{-1}_{N,L,k}\, L^{-1}_{N,L,k}\ . \tag{100}$$

By comparing (100) with (47), we get by uniqueness of triangular factorizations

$$U_{N,L,k} = L^{-H}_{N,L,k}\ ,\quad D_{N,L,k} = G^{-1}_{N,L,k}\ . \tag{101}$$

Hence, using (45), we obtain

$$\gamma^{-1}_N(k) = (G_{N,L,k})_{L,L} \tag{102}$$

which is the last element of $G_{N,L,k}$, and

$$u_{N,L,k} = U_{N,L,k}u_{L,L} = L^{-H}_{N,L,k}u_{L,L}\ . \tag{103}$$

So we can compute $u_{N,L,k}$ by resolving the triangular system

$$L^H_{N,L,k}u_{N,L,k} = u_{L,L} \tag{104}$$

which requires $0.5\,L^2 + \mathcal{O}(L)$ operations.

In order to obtain the a posteriori prediction and filtering errors, one needs to solve the three systems of equations

$$\left\{\begin{array}{lcl} \underline{\gamma}_N^{-1}(k)\, \epsilon_{N,L,k} & = & \epsilon^p_{N,L,k} \\ \\ \underline{\gamma}_N^{-1}(k)\, r_{N,L,k} & = & r^p_{N,L,k} \\ \\ \underline{\gamma}_N^{-1}(k)\, e_{N,L,k+1} & = & e^p_{N,L,k+1} \end{array}\right. \tag{105}$$

where the relations for the prediction errors can be shown in a similar fashion as we have shown in section 2.5 for the filtering errors. Since the displacement rank of the likelihood variable is 3, the computation of the LDU factors of $\underline{\gamma}_N^{-1}(k)$ takes $2L^2 + \mathcal{O}(L)$ operations using the unnormalized generalized Schur algorithm [1]. This algorithm only requires the generators of $\underline{\gamma}_N^{-1}(k)$, which are available. The triangular decomposition of $\underline{\gamma}_N^{-1}(k)$ allows the systems of equations in (105) to be computed by consecutively solving two sets of triangular equations for each of the three systems, viz.

$$L_{N,L,k} \left[\underline{\epsilon}^p_{N,k}\ \underline{r}^p_{N,k}\ \underline{e}^p_{N,k+1}\right] = \left[\epsilon^p_{N,L,k}\ \ r^p_{N,L,k}\ \ e^p_{N,L,k+1}\right]$$

$$\left(G_{N,L,k} L^H_{N,L,k}\right) \left[\epsilon_{N,L,k}\ \ r_{N,L,k}\ \ e_{N,L,k+1}\right] = \left[\underline{\epsilon}^p_{N,k}\ \underline{r}^p_{N,k}\ \underline{e}^p_{N,k+1}\right]. \tag{106}$$

These backsubstitutions take $3L^2 + \mathcal{O}(L)$ operations. It is possible to solve these systems of equations in parallel with the factorization of $\underline{\gamma}_N^{-1}(k)$, avoiding backsubstitutions and the storage of the triangular factor $L_{N,L,k}$ (see [1, page 37],[2]). The computational cost for doing so remains roughly the same. Alternatively, one may want to use a generalized doubling Schur algorithm [1] to reduce the computational complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L(\log\ L)^2)$.

4.5 The Complete Algorithm

In Table I, we present a summary of the FSU RLS algorithm by collecting together the recursions derived previously. We remind the reader that $M = \frac{N+1}{L}$ is assumed to be an integer. We may remark however that the reason why we use vectors of length $N+1$ instead

of length N is for notational convenience rather than out of real necessity. At the start of the update from time $k-L$ to time k, we have the following quantities available:
$W_{N,k-L}$, $A_{N,k-L}$, $A_{N,k-L+1}$, $B_{N,k-L}$, $\left[0 \; \widetilde{C}_{N,k-L}\right]$, $\alpha_N(k-L)$, $\alpha_N(k-L+1)$, $\beta_N(k-L)$, $\gamma_N(k-L)$ and $e_N(k-L+1)$.

For the initialization of the FSU RLS algorithm, we can consider the soft constraint initialization technique introduced in [4]. The addition of the soft constraint to the LS cost function (2) can be interpreted as resulting from an unconstrained LS problem in which the input signal at negative times contains one non-zero sample: $x(-N) = \sqrt{\mu}$ (see [4]). In order to simplify the initialization occurring at time $k = 0$, but which requires also some quantities from $k = 1$, we shall assume $x(1) = 0$. So the (non-zero) input signal actually starts at time $k = 2$. With this input signal, the following quantities can be straightforwardly computed from their definition:

$$\begin{aligned}
&W_{N,0} = W_0\\
&A_{N,0} = A_{N,1} = [1 \; 0 \cdots 0]\\
&B_{N,0} = [0 \cdots 0 \; 1] \, , \; \widetilde{C}_{N,0} = [0 \cdots 0]\\
&\alpha_N(0) = \lambda^N \mu , \; \alpha_N(1) = \lambda \alpha_N(0)\\
&\beta_N(0) = \mu \, , \; \gamma_N(0) = 1\\
&e_N(1) = 0 \; .
\end{aligned} \tag{107}$$

With this initialization at $k = 0$, the FSU RLS algorithm in Table I provides updates at times k that are integer multiples of L. Note that the sequential filtering errors of the RLS algorithm are obtained in the process (see (106)).

$\boxed{\text{Suitable position for Table I}}$

4.6 Computational Complexity

The algorithm can be partitioned in four major parts:
The first part comprises equations I-(1-5) (equations (1)-(5) of table I), which involve the inner products of filters with the data matrix $X_{N+1,L,k}$ and give the a priori errors and $\eta_{N,L,k}$. The FFT's of (portions of length L extended with L zeros of) $[W_{N,k-L} \; 0]$, $A_{N,k-L}$, $B_{N,k-L}$ and $\left[0 \; \widetilde{C}_{N,k-L}\right]$ are computed and only $x_{2L,k}$ has to be Fourier transformed at time k for the

data matrix (for the first $L \times L$ block $X_{L,L,k}$ of $X_{N+1,L,k}$, the other $L \times L$ blocks having been treated already at times $k-L$, $k-2L, \ldots$).
The second part (equations I-(6-9)) concerns the triangularization of the likelihood variable using the generalized Schur algorithm. This allows the computation of the sequential a priori errors, the a posteriori errors, and the vector $u_{N,L,k}$. The total amount of operations is $5.5L^2$.
The third part (equations I-(11-20)) involves the update of the generators. Finally, the joint-process filter gets updated in the last part I-(21).

The complexity of the FSU RLS is $\mathcal{O}(8\frac{N+1}{L}\frac{FFT(2L)}{L} + 35\frac{N}{L} + 5.5L)$ operations per sample. This can be very interesting for long filters. For example when $(N, L) = (4095, 256); (8191, 256)$ and the FFT is done via the split radix ($FFT(2m) = mlog_2(2m)$ real multiplications for real signals) the multiplicative complexity is respectively $0.81N$ and $0.61N$ per sample, compared to $7N$ for the FTF algorithm, the currently fastest RLS algorithm. The number of additions is somewhat higher. The cost we pay is a processing delay which is of the order of L samples.

4.7 Rescue Procedure

We have simulated the algorithm and have verified that it works. Preliminary experience appears to indicate that the numerical behavior of the algorithm may require further attention. Given the numerically unstable behavior of the related Fast Transversal Filter algorithm [11], this finding is not really surprising. One way to overcome numerical problems is to elminate long-term round-off error accumulation by reinitializing the state of the algorithm (i.e. the set of quantities that get initialized before running the algorithm) at regular time intervals. Preferably, one should reinitialize the algorithm state to its correct value (from which the state is deviating due to round-off error accumulation). This could be done by computing at regular time intervals, in parallel with the FSU RLS algorithm, the algorithmic quantities of the FSU RLS algorithm using the Block RLS algorithm described in the beginning of this paper, and by refreshing those quantities in the FSU RLS algorithm with their values obtained from the Block RLS algorithm. However, though this procedure would provide an excellent solution for the numerical error propagation problem, it leads to an increase in computational complexity which is furthermore spread out unevenly in time.

An alternative is to use a rescuing procedure [4]. A rescuing procedure is also a reinitialization procedure, but it reinitializes the state of the algorithm to a value that is simple to compute and hence that deviates from the true RLS value. The rescue procedure in [4] replaces the past input signal by single pulse and hence the sample covariance matrix gets replaced by a diagonal matrix. The following rescue procedure is much more data dependent and corresponds to approximating the prewindowed data matrix by a closely related pre- and postwindowed data matrix, and hence the sample covariance matrix gets replaced by a Toeplitz matrix (apart from a pre- and postmultiplication with a diagonal matrix that contains powers of λ, see [9]):

$$\begin{aligned}
&A_{N,k} = J\,B_{N,k}\,\widetilde{\Lambda}_{N+1}\ ,\ A_{N,k+1} = A_{N,k}\\
&\alpha_N(k) = \lambda^N\,\beta_N(k)\ ,\ \alpha_N(k{+}1) = \lambda\alpha_N(k) + |x(k{+}1)|^2\\
&\widetilde{C}_{N,k} = [0\cdots 0]\ ,\ X_N(k) = [0\cdots 0]^H\\
&\gamma_N(k) = 1\ ,\ e_N(k{+}1) = x(k{+}1)\ .
\end{aligned} \tag{108}$$

Note that $B_{N,k}$, being a minimum-phase filter, is kept rather than $A_{N,k}$ in order to guarantee the positive definiteness of the underlying sample covariance matrix. To further reduce the amount of suboptimality introduced by the rescuing procedure, only the state of the prediction part of the algorithm should be reinitialized. The filtering (joint-process) part, which is stable, should not be reinitialized [10]. Hence the filter $W_{N,k}$ remains unchanged. Also, a distinction should be made between the data vector $X_N(k)$ used in the prediction part (which gets zeroed at a rescue as shown in (108)) and the data vector $X_N(k)$ used in the filtering part, which does not get modified at a rescue.

Apart from a reinitialization procedure, a rescue procedure also requires a timing strategy to determine at which time instants to reinitialize. One could rescue periodically. However, the maximum allowable time period depends on the machine precision and on the input signal. Hence, with a periodic rescuing, one tends to design conservatively and hence to rescue too frequently (introducig too much suboptimality w.r.t. to the true RLS performance). It is preferable to monitor the round-off errors and to rescue only when these become too important. This requires a measurement of the accumulated round-off error. Such measurement possibilities are available in the FSU RLS algorithm. Indeed, the last column of the Kalman gain $\left[\underline{\widetilde{C}}_{N,k}\;0\right]$ is zero in an infinite precision environment, but will differ from zero due to accumulated round-off errors in the state of the prediction part of the algorithm. In particular we can compute the first element of the last column of $\left[\underline{\widetilde{C}}_{N,k}\;0\right]$, viz.

$$\begin{aligned}\delta_k \;=\; \left(\left[\underline{\widetilde{C}}_{N,k}\;0\right]\right)_{1,N+1} \;&=\; -\lambda^{-L}\left[e_N^{pH}(k{-}L{+}1)\alpha_N^{-1}(k{-}L)\left(A_{N,k-L}\right)_{N+1}\right.\\ &\qquad \left. -r_N^{pH}(k{-}L{+}1)\beta_N^{-1}(k{-}L)-\lambda\left(\widetilde{C}_{N,k-L}\right)_N\right]\;. \qquad (109)\end{aligned}$$

which requires a negligible amount of computations. In order to decide whether δ_k is big, it can be meaningfully compared with $\left(\widetilde{C}_{N+1,k-L+1}\right)_{N+1}=-r_N^{pH}(k{-}L{+}1)\lambda^{-1}\beta_N^{-1}(k{-}L)$. Or, after averaging instantaneous variations, we decide to carry out a rescue whenever

$$\delta_k^2 \;>\; K\frac{1-\lambda}{\beta_N(k-L)} \qquad (110)$$

Where K is a small constant (e.g. 10^{-2}) to prevent that δ_k becomes too inportant. After a rescue, the accumulated round-off errors are erased and δ_k redeparts from zero (but on the other hand, the least-squares problem has been modified a bit). The test on δ_k and the ensuing potential rescue are carried out after the updating operations at time k, before moving to the update operations at time $k{+}L$. The rescue procedure has no significant effect on the computational complexity of the algorithm.

4.8 Concluding Remarks

We have proposed a new algorithm for exactly solving the RLS problem, with a computational complexity that can be lower than that of the usual fast RLS algorithms. The FSU RLS algorithm applies especially when the filter to be adapted is very long, such as in the acoustic echo cancellation problem. The algorithm trades off computational complexity for some processing delay. For applications such as video conferencing, this delay should not pose a problem since the audio should be synchronized with the video, and the video gets delayed due to its computationally intensive coding.

In [14],[13] we have also presented the FSU FTF algorithm, an alternative algorithm with a very similar computational complexity, but a very different internal structure. These developments lead us to conjecture that perhaps a lower bound on computational complexity has been reached, at least in the context of the strategy of trading off complexity for processing delay. Preliminary experience indicates that the numerical behavior of the FSU FTF algorithm also requires further attention. A more elaborate investigation of these numerical issues is the subject of ongoing research. In particular, a stabilization procedure to make the algorithms inherently stable such as has been done in [11] for the FTF algorithm, would provide a solution that is preferable over rescue procedures.

References

[1] J. Chun. *Fast Array Algorithms for Structured Matrices.* PhD thesis, Stanford University, Stanford, CA, June 1989.

[2] J. Chun, V. Roychowdhury, and T. Kailath. "Systolic Array for Solving Toeplitz Systems of Equations". In *Proc. SPIE's 32nd Conference on Advanced Algorithms and Architectures for Signal Processing III*, pages 975–03, San Diego, CA, Aug. 1988.

[3] J.M. Cioffi. "The Block-Processing FTF Adaptive Algorithm". *IEEE Trans. on ASSP*, ASSP-34(1):77–90, Feb. 1986.

[4] 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.

[5] 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.

[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] D.T.M. Slock. "Reconciling Fast RLS Lattice and QR Algorithms". In *Proc. ICASSP 90 Conf.*, pages 1591–1594, Albuquerque, NM, April 3–6 1990.

[9] D.T.M. Slock. "Backward Consistency Concept and Round-Off Error Propagation Dynamics in Recursive Least-Squares Algorithms". *Optical Engineering*, 31(6):1153–1169, June 1992.

[10] D.T.M. Slock and T. Kailath. "Fast Transversal Filters with Data Sequence Weighting". *IEEE Trans. Acoust. Speech Sig. Proc.*, ASSP-37(3):346–359, March 1989.

[11] 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.

[12] D.T.M. Slock and T. Kailath. "A Modular Prewindowing Framework for Covariance FTF RLS Algorithms". *Signal Processing*, 28(1):47–61, July 1992.

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

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

[15] 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.

[16] X.-H. Yu and Z.-Y. He. "Efficient Block Implementation of Exact Sequential Least-Squares Problems". *IEEE Trans. Acoust., Speech and Signal Proc.*, ASSP-36:392–399, March 1988.

List of Figures

Figure 1: The adaptive FIR filtering system.

Table I: FSU RLS Algorithm

#	Computation	Cost per L samples
1	$\epsilon^{p}_{N,L,k} = d_{L,k} + X_{N+1,L,k} \; [W_{N,k-L} \;\; 0]^{H}$	$(2 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 2(\mathrm{N}+1)$
2	$r^{p}_{N,L,k} = X_{N+1,L,k} \; B^{H}_{N,k-L}$	$(1 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 2(\mathrm{N}+1)$
3	$\eta_{N,L,k} = X_{N+1,L,k} \left[0 \;\; \widetilde{C}_{N,k-L}\right]^{H}$	$(1 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 2(\mathrm{N}+1)$
4	$\begin{bmatrix} e^{p}_{N,L,k} \\ e^{H}_{N}(k{+}1 \mid k{-}L) \end{bmatrix} = \begin{bmatrix} X_{N+1,L,k}\, A^{H}_{N,k-L} \\ X^{H}_{N+1}(k{+}1)\, A^{H}_{N,k-L} \end{bmatrix}$	$(1 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 3\mathrm{N}$
5	$e^{p}_{N,L,k+1} = \begin{bmatrix} \left(e^{p}_{N,L,k}\right)_{2:L} \\ e^{H}_{N}(k{+}1 \mid k{-}L) \end{bmatrix} + \begin{bmatrix} (\eta_{N,L,k})_{2:L} \\ X^{H}_{N}(k)\widetilde{C}^{H}_{N,k-L} \end{bmatrix} e^{H}_{N}(k{-}L{+}1)$	$\mathrm{N}+\mathrm{L}$
6	$\underline{\gamma}^{-1}_{N}(k) = \nabla^{-1}_{\lambda} \Big\{ e^{p}_{N,L,k}\lambda^{-L}\alpha^{-1}_{N}(k{-}L)e^{p\,H}_{N,L,k} - r^{p}_{N,L,k}\lambda^{-L}\beta^{-1}_{N}(k{-}L)r^{p\,H}_{N,L,k} + (\eta_{N,L,k} - u_{L,1})\,\lambda^{-L+1}\gamma_{N}(k{-}L)\,(\eta_{N,L,k} - u_{L,1})^{H} \Big\}$	
7	$L_{N,L,k}\, G_{N,L,k}\, L^{H}_{N,L,k} = \underline{\gamma}^{-1}_{N}(k)$	$\mathrm{Generalized\ Schur\ algorithm: 2L^2}$
8	$L^{H} u_{N,L,k} = u_{L,L}$	$\mathrm{Backsubstitution: 0.5L^2}$
9	$\underline{\gamma}^{-1}_{N}(k)\,[\epsilon_{N,L,k} \;\; r_{N,L,k} \;\; e_{N,L,k+1}] = \left[\epsilon^{p}_{N,L,k} \;\; r^{p}_{N,L,k} \;\; e^{p}_{N,L,k+1}\right]$	$\mathrm{Backsubstitutions: 3L^2}$
10	$\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right] = \nabla^{-1}_{\lambda} \Big\{ -e^{p}_{N,L,k}\lambda^{-L}\alpha^{-1}_{N}(k{-}L)A_{N,k-L} + r^{p}_{N,L,k}\lambda^{-L}\beta^{-1}_{N}(k{-}L)B_{N,k-L} - (\eta_{N,L,k} - u_{L,1})\lambda^{-L+1}\gamma_{N}(k{-}L)\left[0 \;\; \widetilde{C}_{N,k-L}\right] \Big\}$	
11	$B_{N,k} = B_{N,k-L} + r^{H}_{N,L,k}\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right]$	$(3 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 6(\mathrm{N}+1) + 10\mathrm{L}$
12	$\beta_{N}(k) = \lambda^{L}\beta_{N}(k{-}L) + r^{H}_{N,L,k}r^{p}_{N,L,k}$	$\mathrm{L}+1$
13	$\left[\widetilde{C}_{N,k} \;\; 0\right] = u^{H}_{N,L,k}\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right]$	$(3 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 6(\mathrm{N}+1) + 10\mathrm{L}$
14	$\gamma^{-1}_{N}(k) = (G_{N,L,k})_{L,L}$	
15	$A_{N,k+1} = A_{N,k-L+1} + e^{H}_{N,L,k+1}\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right] Z^{H}_{N+1}$	$(3 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 6(\mathrm{N}+1) + 10\mathrm{L}$
16	$e^{p}_{N}(k{+}1) = e^{p\,H}_{N,L,k+1}u_{N,L,k}$	L
17	$e_{N}(k{+}1) = e^{p}_{N}(k{+}1)\gamma_{N}(k)$	1
18	$A_{N,k} = A_{N,k+1} - e_{N}(k{+}1)\left[0 \;\; \widetilde{C}_{N,k}\right]$	N
19	$\alpha_{N}(k{+}1) = \lambda^{L}\alpha_{N}(k{-}L{+}1) + e^{H}_{N,L,k+1}e^{p}_{N,L,k+1}$	$\mathrm{L}+1$
20	$\alpha_{N}(k) = \lambda^{-1}\left(\alpha_{N}(k{+}1) - e_{N}(k{+}1)e^{p\,H}_{N}(k{+}1)\right)$	2
21	$[W_{N,k} \;\; 0] = [W_{N,k-L} \;\; 0] + \epsilon^{H}_{N,L,k}\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right]$	$(6 + \frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 6(\mathrm{N}+1) + 10\mathrm{L}$
Total cost per sample		$(20 + 8\frac{\mathrm{N+1}}{\mathrm{L}})\frac{\mathrm{FFT(2L)}}{\mathrm{L}} + 35\frac{\mathrm{N}}{\mathrm{L}} + 5.5\mathrm{L}$

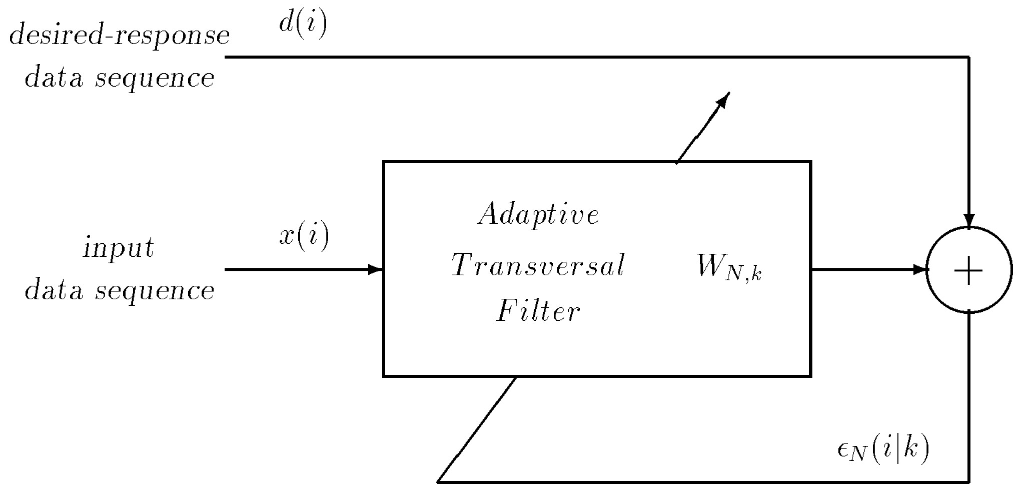
desired-response data sequence
$d(i)$
input data sequence
$x(i)$
Adaptive Transversal Filter
$W_{N,k}$
$+$
$\epsilon_N(i|k)$

Figure 1: