FSU RLS and FSU FTF: Fast Recursive Least-Squares Algorithms Based on Displacement Structure and the FFT

Dirk T.M. Slock   Karim Maouche

Institut EURECOM
2229 route des Crêtes, B.P. 193, 06904 Sophia Antipolis Cedex, FRANCE
slock@eurecom.fr

Abstract

We present two new fast algorithms for Recursive Least-Squares (RLS) adaptive filtering. These algorithms are especially suited for adapting very long filters such as in the acoustic echo cancellation problem. For the FSU RLS, 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). The second algorithm which is the FSU FTF apply the same idea to the FTF algorithm. It uses a Schur procedure to compute the rotation matrix that allows to adapt the filter and use also the FFT. Its computational complexity is of the same order as the FSU RLS.

1 Introduction

Fast RLS algorithms such as the Fast Transversal Filter (FTF) algorithm [1],[2],[3] and the Fast Lattice/Fast QR (FLA/FQR) algorithms [4] 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 $L \geq 1$. The approach we pursue here (which should be especially applicable when $L < N$) consists of using the same strategy as the RLS algorithm : 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. We derive a fast version of the SU RLS algorithm, the FSU RLS algorithm. In a second step, the same idea is applied to the FTF algorithm, which will give the FSU FTF 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 [1],[5],[2],[3], 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.

2.1 The RLS Algorithm

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

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

where $X_N(i) = \left[x^H(i)\, x^H(i-1) \cdots x^H(i-N+1)\right]$ is the regression vector and superscript ^{H} denotes Hermitian (complex conjugate) transpose. In the RLS algorithm, the set of N transversal filter coefficients $W_{N,k} = \left[W^1_{N,k} \cdots W^N_{N,k}\right]$ are adapted so as to minimize recursively the following LS criterion

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

where $\lambda \in (0,1]$ is the exponential weighting factor, $\mu > 0$, $\Lambda_N = \text{diag}\left\{\lambda^{N-1}, \ldots, \lambda, 1\right\}$, $\|v\|^2_\Lambda = v\Lambda v^H$, $\|.\| = \|.\|_I$. 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

$$W_{N,k} \ = \ -P^H_{N,k} R^{-1}_{N,k} \tag{3}$$

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 for $R^{-1}_{N,k}$, we obtain the RLS algorithm:

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

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

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

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

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

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

where $\epsilon^p_N(k)$ and $\epsilon_N(k)$ are the a priori and a posteriori error signals (resp. **p**redicted and filtered errors in the Kalman filtering terminology) and one can verify (or see [1]) that 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. 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 to $\mathcal{O}(N)$ (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 an alternative way to reduce the computational complexity of the RLS algorithm.

2.2 The SU RLS Algorithm

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

$$d_{L,k} = \begin{bmatrix} d^H(k-L+1) \\ \vdots \\ d^H(k) \end{bmatrix},\ x_{L,k} = \begin{bmatrix} x^H(k-L+1) \\ \vdots \\ x^H(k) \end{bmatrix},\ X_{N,L,k} = \begin{bmatrix} X_N^H(k-L+1) \\ \vdots \\ X_N^H(k) \end{bmatrix} = [x_{L,k} \cdots x_{L,k-N+1}] \tag{11}$$

where $X_{N,L,k}$ is a Toeplitz data matrix. We can now obtain the following multi-step updates from (4)

$$P_{N,k} \;=\; \lambda^L P_{N,k-L} + X^H_{N,L,k}\Lambda_L d_{L,k}\ ,\quad R_{N,k} \;=\; \lambda^L R_{N,k-L} + X^H_{N,L,k}\Lambda_L X_{N,L,k}\ . \tag{12}$$

If we plug in these recursions into the solution (3), then we get similarly to the derivation of the RLS algorithm the following recursion

$$\widetilde{\underline{C}}_{N,k} \;=\; -X_{N,L,k}\lambda^{-L}R^{-1}_{N,k-L} \tag{13}$$

$$\underline{\gamma}_N^{-1}(k) \;=\; \Lambda_L^{-1} - X_{N,L,k}\widetilde{\underline{C}}^H_{N,k} \tag{14}$$

$$R^{-1}_{N,k} \;=\; \lambda^{-L}R^{-1}_{N,k-L} \;-\; \widetilde{\underline{C}}^H_{N,k}\underline{\gamma}_N(k)\widetilde{\underline{C}}_{N,k} \tag{15}$$

$$\epsilon^p_{N,L,k} \;=\; d_{L,k} + X_{N,L,k}W^H_{N,k-L} \tag{16}$$

$$\underline{\gamma}_N^{-1}(k)\,\epsilon_{N,L,k} \;=\; \epsilon^p_{N,L,k} \tag{17}$$

$$W_{N,k} \;=\; W_{N,k-L} + \epsilon^H_{N,L,k}\widetilde{\underline{C}}_{N,k} \tag{18}$$

where $\epsilon^p_{N,L,k}$ and $\epsilon_{N,L,k}$ are vectors of a priori and a posteriori errors respectively:

$$\epsilon^p_{N,L,k} \;=\; \begin{bmatrix} \epsilon^H_N(k-L+1|k-L) \\ \vdots \\ \epsilon^H_N(k|k-L) \end{bmatrix}\ ,\quad \epsilon_{N,L,k} \;=\; \begin{bmatrix} \epsilon^H_N(k-L+1|k) \\ \vdots \\ \epsilon^H_N(k|k) \end{bmatrix}\ . \tag{19}$$

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 now consider a first instance of exploiting the FFT technique to reduce the computational complexity in equation (16) by a factor $\mathcal{O}\left(\frac{L}{\log_2 L}\right)$.

2.3 Fast Computation of the Filtering Errors using the FFT

Consider a partitioning of the filter coefficients vector in $M = N/L$ subvectors of length L: $W_{N,k} = \left[\underline{W}^1_{N,k} \cdots \underline{W}^M_{N,k}\right]$. Now consider the vector of (block) a priori filtering errors

$$\epsilon^p_{N,L,k} \;=\; d_{L,k} + X_{N,L,k}W^H_{N,k-L} \;=\; d_{L,k} + \sum_{j=1}^{M} X_{L,L,k-(j-1)L}\underline{W}^{j\,H}_{N,k-L}\ . \tag{20}$$

In other words, we have essentially M 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 [6], 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.

$$\underline{X}^H_{L,L,k} \;=\; \begin{bmatrix} * & X^H_{L,L,k} \\ X^H_{L,L,k} & * \end{bmatrix} \;=\; \mathcal{C}\left(x^H_{2L,k}\right) \tag{21}$$

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

$$X^H_{L,L,k-(j-1)L}\,\underline{W}^{j\,H}_{N,k-L} \;=\; [I_L \;\; 0_{L\times L}]\;\mathcal{C}\left(x^H_{2L,k-(j-1)L}\right)\begin{bmatrix} 0_{L\times 1} \\ \underline{W}^{j\,H}_{N,k-L} \end{bmatrix}. \tag{22}$$

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

$$\mathcal{C}(c^H)\,v \;=\; \frac{1}{m}\,\mathcal{C}(c^H)\,F_m^H F_m v \;=\; \frac{1}{m}\,F_m^H\;\mathrm{diag}^H\left(F_m\,c\right)\,F_m v \tag{23}$$

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 (22) requires the padding of $\underline{W}^{j\,H}_{N,k-L}$ with L zeros, the DFT of the resulting vector, the DFT of $x_{2L,k-(j-1)L}$, the product of the two DFTs, and the (scaled) IDFT of this product. When the FFT is used to perform the DFTs, this leads to a computationally more efficient procedure than the straightforward matrix-vector product which would require L^2 multiplications. Note that at time k, only the FFT of $x_{2L,k}$ needs to be computed; the FFTs of $x_{2L,k-jL}$, $j=1,\ldots,M-1$ have been computed at previous time instants. Remark also that we need to apply the inverse DFT (and the scaling by $\frac{1}{2L}$) only once, after having summed up the M products in the frequency domain. The above procedure reduces the N computations per sample for $\epsilon^p_{N,L,k}$ to

$$N\left[\frac{\mathrm{FFT}(2L)}{L^2}+\frac{2}{L}\right]+2\frac{\mathrm{FFT}(2L)}{L} \tag{24}$$

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 Relation Between the Filtering Errors and Kalman Gains 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.

Let us introduce the following notation

$$\begin{aligned}
\underline{\epsilon}^p_{N,k} &= \begin{bmatrix} \epsilon^{p\,H}_N(k-L+1) \\ \vdots \\ \epsilon^{p\,H}_N(k) \end{bmatrix} = \begin{bmatrix} \epsilon^H_N(k-L+1|k-L) \\ \vdots \\ \epsilon^H_N(k|k-1) \end{bmatrix} && (25)\\
,_{N,L,k} &= \mathrm{diag}\left\{\gamma_N(k-L+1),\ldots,\gamma_N(k)\right\} && (26)\\
D_{N,L,k} &= \Lambda_L\;,_{N,L,k} && (27)
\end{aligned}$$

where $\underline{\epsilon}^p_{N,k}$ is a vector of L a priori errors of the RLS algorithm. Then one can show that

$$\epsilon^{p\,H}_{N,L,k}\,\underline{\gamma}_N(k)\,\epsilon^p_{N,L,k} \;=\; \underline{\epsilon}^{p\,H}_{N,k}\,D_{N,L,k}\,\underline{\epsilon}^p_{N,k}\;. \tag{28}$$

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

$$\underline{\gamma}_N(k) \;=\; U_{N,L,k}\,D_{N,L,k}\,U^H_{N,L,k} \tag{29}$$

where $U_{N,L,k}$ is upper triangular with unit diagonal, and the diagonal factor is indeed the $D_{N,L,k}$ introduced in (27). One can show that there exists a unit-diagonal triangular relation between $\epsilon^p_{N,L,k}$ and $\underline{\epsilon}^p_{N,k}$ and hence from (28) and (29), this triangular factor must be $U_{N,L,k}$, viz.

$$U^H_{N,L,k}\,\epsilon^p_{N,L,k} \;=\; \underline{\epsilon}^p_{N,k}\;. \tag{30}$$

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 (29) can easily be made part of the inversion of $\underline{\gamma}_N(k)$ in the SU RLS algorithm. One can now also easily show

$$\underline{\gamma}_N(k)\,\epsilon^p_{N,L,k} \;=\; \underbrace{U_{N,L,k}\Lambda_L\,,\,\underbrace{{}_{N,L}(k)\,\underbrace{U^H_{N,L,k}\qquad\underbrace{\epsilon^p_{N,L,k}}_{\text{a priori SURLS errors}}}_{\underline{\epsilon}^p_{N,k}\ \text{a priori RLS errors}}}_{\underline{\epsilon}_{N,k}\ \text{a posteriori RLS errors}}}_{\epsilon_{N,L,k}\ \text{a posteriori SURLS errors}} \quad . \tag{31}$$

Similarly, there exists a relation between the Kalman gain in the SU RLS algorithm and L consecutive Kalman gains of the RLS algorithm. One can show that

$$U^H_{N,L,k}\underline{\widetilde{C}}_{N,k} \;=\; \Lambda_L^{-1}\begin{bmatrix}\widetilde{C}_{N,k-L+1}\\ \vdots \\ \widetilde{C}_{N,k}\end{bmatrix} \quad . \tag{32}$$

Let $u_{N,L,k}$ be the last column of $U_{N,L,k}$. Then (32) leads in particular to

$$u^H_{L,1}\,\underline{\widetilde{C}}_{N,k} \;=\; \lambda^{-L+1}\widetilde{C}_{N,k-L+1} \tag{33}$$

$$u^H_{N,L,k}\,\underline{\widetilde{C}}_{N,k} \;=\; \widetilde{C}_{N,k} \quad . \tag{34}$$

$u_{L,i}$ being the $L\times 1$ vector with 1 at the ith position and zeros elsewhere.

3 Displacement Structure of the SU RLS Kalman Gain Quantities

Consider the displacement structure of a matrix R:

$$\nabla_\lambda\,R \;=\; R - \lambda\,Z\,R\,Z^H \;=\; \sum_{i=1}^{\delta} u_i\,v_i^H \tag{35}$$

where δ, called displacement rank, is the rank of $R - \lambda\,Z\,R\,Z^H$, Z is the lower shift matrix (ones on the first subdiagonal and zeros elsewhere), and the vectors u_i, v_i are called the generators of R for the following reason. By solving the Lyapunov equation (35), it is straightforward to obtain the following representation for R:

$$R \;=\; \nabla_\lambda^{-1}\left(\sum_{i=1}^{\delta} u_i\,v_i^H\right) \;=\; \sum_{i=1}^{\delta}\sum_{j=0}^{\infty}\lambda^j\,Z^j\,u_i\,v_i^H\,(Z^H)^j \;=\; \sum_{i=1}^{\delta}\mathcal{L}(u_i)\,\widetilde{\Lambda}\,\mathcal{L}^H(v_i) \tag{36}$$

where $\widetilde{\Lambda} = \operatorname{diag}\{1,\ \lambda,\ \lambda^2,\ldots\}$ 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 $R^{-1}_{N,k}$.

3.1 Displacement Structure of the Inverse Sample Covariance Matrix

In [7], the following displacement structure was derived

$$\nabla_\lambda\begin{bmatrix}R^{-1}_{N,k} & 0\\ 0 & 0\end{bmatrix} \;=\; A^H_{N,k}\alpha_N^{-1}(k)A_{N,k} - B^H_{N,k}\beta_N^{-1}(k)B_{N,k} + \lambda\left[0\ \ \widetilde{C}_{N,k}\right]^H\gamma_N(k)\left[0\ \ \widetilde{C}_{N,k}\right] \tag{37}$$

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 [1]).

3.2 Displacement Structure of the Kalman Gain

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

$$e^p_{N,L,k} \;=\; X_{N+1,L,k}\, A^H_{N,k-L} \tag{38}$$

$$r^p_{N,L,k} \;=\; X_{N+1,L,k}\, B^H_{N,k-L}\;. \tag{39}$$

The displacement structure of the SURLS Kalman gain turns out to be

$$\begin{aligned}\nabla_\lambda \left[\underline{\widetilde{C}}_{N,k} \;\; 0\right] \;&=\; -e^p_{N,L,k}\lambda^{-L}\alpha_N^{-1}(k{-}L)A_{N,k-L} + r^p_{N,L,k}\lambda^{-L}\beta_N^{-1}(k{-}L)B_{N,k-L} \\ &\quad -\left(\eta_{N,L,k}-u_{L,1}\right)\lambda^{-L+1}\gamma_N(k{-}L)\left[0 \;\; \widetilde{C}_{N,k-L}\right]\end{aligned} \tag{40}$$

where

$$\eta_{N,L,k} = X_{N+1,L,k}\left[0 \;\; \widetilde{C}_{N,k-L}\right]^H\;. \tag{41}$$

By using (33), 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

$$\eta_{N,L,k} \;=\; \lambda^{L-1}X_{N+1,L,k}\left[0 \;\; \underline{\widetilde{C}}_{N,k-1}\right]^H u_{L,1} \;=\; \lambda^{L-1}X_{N,L,k-1}\underline{\widetilde{C}}^H_{N,k-1}u_{L,1} \tag{42}$$

which leads to

$$\eta_{N,L,k} \;=\; \lambda^{L-1}\left(\Lambda_L^{-1} - \underline{\gamma}_N^{-1}(k{-}1)\right)u_{L,1}\;. \tag{43}$$

3.3 Displacement Structure of the Likelihood Variable

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

$$\nabla_\lambda\, \underline{\gamma}_N^{-1}(k) = \nabla_\lambda\, \Lambda_L^{-1} + \lambda^{-L}\,\nabla_\lambda\left(X_{N,L,k}R^{-1}_{N,k-L}X^H_{N,L,k}\right)\;. \tag{44}$$

By using the displacement structure of the Kalman gain given in (40), the displacement structure of the likelihood variable becomes

$$\begin{aligned}\nabla_\lambda\, \underline{\gamma}_N^{-1}(k) \;&=\; e^p_{N,L,k}\lambda^{-L}\alpha_N^{-1}(k{-}L)e^{p\,H}_{N,L,k} - r^p_{N,L,k}\lambda^{-L}\beta_N^{-1}(k{-}L)r^{p\,H}_{N,L,k} \\ &\quad +\left(\eta_{N,L,k}-u_{L,1}\right)\lambda^{-L+1}\gamma_N(k{-}L)\left(\eta_{N,L,k}-u_{L,1}\right)^H\;.\end{aligned} \tag{45}$$

This last equation exhibits the Hermitian structure inherited from $\underline{\gamma}_N^{-1}(k)$.
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 (36). 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 $\underline{\widetilde{C}}_{N,k}$ and $\underline{\gamma}_N^{-1}(k)$ by using the SU RLS equations (13), (14), it suffices to update the filters $A_{N,k}$, $B_{N,k}$ and $\left[0 \;\; \widetilde{C}_{N,k}\right]$, and to compute the filter outputs $e^p_{N,L,k}$, $r^p_{N,L,k}$ and $\eta_{N,L,k}$ from their definitions (38), (39) and (41), using the data available at the time instant k.

4 The FSU RLS Algorithm

We can rewrite equation (18) in the form

$$\left[W_{N,k} \;\; 0\right] = \left[W_{N,k-L} \;\; 0\right] + \epsilon^H_{N,L,k}\left[\underline{\widetilde{C}}_{N,k} \;\; 0\right]\;. \tag{46}$$

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

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	$\begin{bmatrix} e^H_N(k-L+1) \\ e^p_{N,L,k+1} \end{bmatrix} = \begin{bmatrix} X_{N+1,L,k}\, A^H_{N,k-L+1} \\ X^H_{N+1}(k+1)\, A^H_{N,k-L+1} \end{bmatrix}$	$(1+\frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)} + 3\mathrm{N}$
5	$e^p_{N,L,k} = \begin{bmatrix} e^H_N(k-L+1) \\ \left(e^p_{N,L,k+1}\right)_{1:L-1} \end{bmatrix} - \eta_{N,L,k}\, e^H_N(k-L+1)$	L
6	$\underline{\gamma}^{-1}_N(k) = \nabla^{-1}_\lambda \left\{ e^p_{N,L,k} \lambda^{-L} \alpha^{-1}_N(k-L) e^{p\,H}_{N,L,k} \right.$	
	$- r^p_{N,L,k} \lambda^{-L} \beta^{-1}_N(k-L) r^{p\,H}_{N,L,k}$	
	$\left. + \left(\eta_{N,L,k} - u_{L,1}\right) \lambda^{-L+1} \gamma_N(k-L) \left(\eta_{N,L,k} - u_{L,1}\right)^H \right\}$	
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	$\underline{\gamma}^{-1}_N(k) \left[\epsilon_{N,L,k}\ \ r_{N,L,k}\ \ e_{N,L,k+1}\right] = \left[\epsilon^p_{N,L,k}\ \ r^p_{N,L,k}\ \ e^p_{N,L,k+1}\right]$	$3\mathrm{L}^2$
10	$\left[\underline{\widetilde{C}}_{N,k}\ \ 0\right] = \nabla^{-1}_\lambda \left\{ - e^p_{N,L,k} \lambda^{-L} \alpha^{-1}_N(k-L) A_{N,k-L} \right.$	
	$+ r^p_{N,L,k} \lambda^{-L} \beta^{-1}_N(k-L) B_{N,k-L}$	
	$\left. - \left(\eta_{N,L,k} - u_{L,1}\right) \lambda^{-L+1} \gamma_N(k-L) \left[0\ \ \widetilde{C}_{N,k-L}\right] \right\}$	
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	$\gamma^{-1}_N(k) = \left(G_{N,L,k}\right)_{L,L}$	
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		$(20 + 8\frac{\mathrm{N+1}}{\mathrm{L}})\frac{\mathrm{FFT(2L)}}{\mathrm{L}} + 34\frac{\mathrm{N}}{\mathrm{L}} + 5.5\mathrm{L}$

obtained by resolving the system of equations (17) via the generalized Schur algorithm. Now, if we replace $\left[\underline{\widetilde{C}}_{N,k}\ 0\right]$ in (46) by its expression given in terms of its displacement vectors (via (36) and (40)), the joint-process update equation takes the form

$$[W_{N,k}\ \ 0] = [W_{N,k-L}\ \ 0] + \epsilon^H_{N,L,k} \sum_{i=1}^{3} d_i \mathcal{T}^i_{L,L}\, \widetilde{\Lambda}_L\, \mathcal{G}^i_{L,N+1} \tag{47}$$

where d_i, $\mathcal{T}^i_{L,L}$ and $\mathcal{G}^i_{L,N+1}$ are respectively scalars, $L \times L$ lower and $L \times (N+1)$ upper triangular Toeplitz matrices. The last term in equation (47) 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 section 2.3).

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 $\sum_{i=1}^{3} p^4_{L,i}$.

The RLS Kalman gain can be updated by using (34) and the representation of $\underline{\widetilde{C}}_{N,k}$ in terms of its generators. The updating of the prediction filters is similar to the updating of $W_{N,k}$. A minor complication arises in the update of the forward prediction filter, the details of which can be found in [8]. The resulting FSU RLS algorithm can be found in Table I.

5 The FSU FTF Algorithm

5.1 The Fast Transversal Filter Algorithm

The Fast Transversal Filter exploit efficiently the shift invariance structure of the adaptive FIR filtering problem. Its computational complexity is $7N$. This algorithm can be formalized as follows:

$$\begin{aligned}
\begin{bmatrix} \left[\widetilde{C}_{N,k}\ 0\right] \\ A_{N,k} \\ B_{N,k} \\ [W_{N,k}\ 0] \end{bmatrix} &= \Theta_k \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-1}\right] \\ A_{N,k-1} \\ B_{N,k-1} \\ [W_{N,k-1}\ 0] \end{bmatrix} \\
\alpha_N(k) &= \lambda\alpha_N(k-1) + e_N(k)\, e^p_N(k) \\
\beta_N(k) &= \lambda\beta_N(k-1) + r_N(k)\, r^p_N(k) \\
\gamma_N(k) &= \lambda^N \beta_N(k) / \alpha_N(k)
\end{aligned} \tag{48}$$

where Θ_k is a (4×4) rotation matrix

$$\Theta_k = \Theta^1_k\, \Theta^2_k\, \Theta^3_k\, \Theta^4_k \tag{49}$$

and the four (4×4) matrices Θ^i_k $i = 1, 2, 3, 4$ are given by

$$\begin{aligned}
\Theta^1_k &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \epsilon_N(k) & 0 & 0 & 1 \end{bmatrix} \qquad
\Theta^2_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ r_N(k) & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\Theta^3_k &= \begin{bmatrix} 1 & 0 & \dfrac{r^p_N(k)}{\lambda\beta_N(k-1)} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad
\Theta^4_k = \begin{bmatrix} 1 & -\dfrac{e^p_N(k)}{\lambda\alpha_N(k-1)} & 0 & 0 \\ e_N(k) & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .
\end{aligned} \tag{50}$$

In order to have these matrices, one must compute the a priori errors $e_N^p(k)$, $r_N^p(k)$ and $\epsilon_N^p(k)$ which are the outputs of the filters $A_{N,k-1}$, $B_{N,k-1}$ and $W_{N,k-1}$. We can also constuct these matrices by using the Schur algorithm as we will see in the next section.

5.2 Schur Algorithm for the Fast Transversal Filter

Let us introduce

$$\begin{aligned} \widehat{d}_N^p(k) &= d(k) - \epsilon_N^p(k) & (51)\\ \widehat{d}_N(k) &= d(k) - \epsilon_N(k) \quad . & (52) \end{aligned}$$

Consider the following product

$$F_L(k) = \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-L}\right] \\ A_{N,k-L} \\ B_{N,k-L} \\ \left[W_{N,k-L}\ 0\right] \end{bmatrix} X_{N+1,L,k}^H = \begin{bmatrix} \eta_{N,L,k}^H \\ e_{N,L,k}^{p\,H} \\ r_{N,L,k}^{p\,H} \\ -\widehat{d}_{N,L,k}^{p\,H} \end{bmatrix} . \qquad (53)$$

The first column of $F_L(k)$ is

$$F_L(k)\ u_{L,1} = \left[1-\gamma_N^{-1}(k-L)\ \ e_N^p(k-L+1)\ \ r_N^p(k-L+1)\ \ -\widehat{d}_N^p(k-L+1)\right]^T . \qquad (54)$$

So, with the recursions for $\alpha_N(k-L+1)$ and $\beta_N(k-L+1)$ that are

$$\begin{aligned} \alpha_N(k-L+1) &= \lambda\alpha_N(k-L) + e_N^p(k-L+1)\,\gamma_N(k-L)\,e_N^p(k-L+1) \\ \beta_N(k-L+1) &= \lambda\beta_N(k-L) + r_N^p(k-L+1)\,\gamma_N(k-L+1)\,r_N^p(k-L+1) \qquad (55) \end{aligned}$$

(plus possibly an alternative calculation for $\gamma_N(k-L)$), it is possible to construct Θ_{k-L+1} and then goes on the recursions of the FTF.
Now, if we rotate with Θ_{k-L+1} on both sides of (53), then we have

$$\begin{bmatrix} \left[\widetilde{C}_{N,k-L+1}\ 0\right] \\ A_{N,k-L+1} \\ B_{N,k-L+1} \\ \left[W_{N,k-L+1}\ 0\right] \end{bmatrix} X_{N+1,L,k}^H = \begin{bmatrix} \boxed{\eta_{N,L-1,k}^H} & * \\ e_N(k-L+1) & \boxed{e_{N,L-1,k}^{p\,H}} \\ r_N(k-L+1) & \boxed{r_{N,L-1,k}^{p\,H}} \\ -\widehat{d}_N(k-L+1) & \boxed{-\widehat{d}_{N,L-1,k}^{p\,H}} \end{bmatrix} . \qquad (56)$$

By shifting to the right the first row of the right hand side of (56) and take the three vectors $e_{N,L-1,k}^p$, $r_{N,L-1,k}^p$ and $-\widehat{d}_{N,L-1,k}^{p\,H}$, we obtain $F_{L-1}(k)$ and hence can do the same operations to obtain the next rotation matrices. So by using the well known Schur algorithm working on the four vectors given by the matrix $F_L(k)$, it is possible to obtain the successive rotation matrices from time $k-L+1$ to time k.

5.3 The FSU FTF Algorithm

In a subsampled updating strategy, one need to obtain the adapted filter at time k from the one at time $k-L$. In the case of the FTF algorithm, this will be done by using the Schur procedure described previously but without computing the different filters at each iteration.
To have a polynomial formulation of the problem, let's define

$$\begin{bmatrix} \widetilde{C}_k(z) \\ A_k(z) \\ B_k(z) \\ W_k(z) \end{bmatrix} = \begin{bmatrix} \left[\widetilde{C}_{N,k}\ 0\right] \\ A_{N,k} \\ B_{N,k} \\ \left[W_{N,k}\ 0\right] \end{bmatrix} \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-N} \end{bmatrix} . \qquad (57)$$

Hence (48) can be written as

$$\Theta_k \begin{bmatrix} \tilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ W_k(z) \end{bmatrix} = \begin{bmatrix} z^{-1} & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} \tilde{C}_{k-1}(z) \\ A_{k-1}(z) \\ B_{k-1}(z) \\ W_{N,k-1}(z) \end{bmatrix} . \tag{58}$$

Let

$$\Theta_k(z) = \Theta_k \begin{bmatrix} z^{-1} & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \tag{59}$$

Now, in order to adapt the filter at time k from the one at time $k-L$, it is straightforward to find

$$\begin{bmatrix} \tilde{C}_k(z) \\ A_k(z) \\ B_k(k) \\ W_k(z) \end{bmatrix} = \Theta_{k,L}(z) \begin{bmatrix} \tilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ W_{N,k-L}(z) \end{bmatrix} \tag{60}$$

where

$$\Theta_{k,L}(z) = \Theta_k(z)\,\Theta_{k-1}(z)\cdots\Theta_{k-L+1}(z) \quad . \tag{61}$$

The successive matrices appearing in (61) can be obtained as it was shown, by using the Schur algorithm applied to the rows of $F_L(k)$. The computationnal complexity associated with the Schur procedure is $2.5L^2$ multiplications. It appears from (60) that the filter can be adapted at times multiples of L in a equivalent manner as the FTF algorithm by computing the (4×4) polynomial matrix $\Theta_{k,L}(z)$. The accumulation of the successive matrices $\Theta_i(z)$ takes $7.5L^2$ multiplications since there are five non trivial entries in a matrix $\Theta_i(z)$.
Remark that $\Theta_{k,L}(z)$ has the following structure

$$\Theta_{k,L}(z) = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 0 \\ * & * & * & 1 \end{bmatrix} \tag{62}$$

where the stars stand for polynomials of degree at most L. Hence, the product in (60) represents 12 convolutions of polynomials of order L and N. These convolutions can be done using fast convolution techniques. In the present case where the length of the filter is relatively large, we will choose the FFT technique. In that case the complexity is $3(1+2\frac{N+1}{L})FFT(2L)+2(N+1))$ multiplications and $6(N+1)$ additions for each filter. The product (53) which gives $F_L(k)$ is also computed with the FFT and hence, we have to compute the FFTs of the filters just one time.
The FSU FTF algorithm is given in the table II.

5.4 Concluding Remarks

The complexity of the FSU RLS and the FSU FTF are respectively $\mathcal{O}(8\frac{N+1}{L}\frac{FFT(2L)}{L}+34\frac{N}{L}+5.5L)$ and $\mathcal{O}(8\frac{N+1}{L}\frac{FFT(2L)}{L}+32\frac{N}{L}+10L)$ operations per sample. This can be very interesting for long filters. For example, for the FSU RLS 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 $0.8N$ and $0.6N$ per sample. With the FSU FTF, for the same parameters N and L, the multiplicative complexity is respectively $1.1N$ and $0.74N$, compared to $7N$ for the FTF algorithm, the currently fastest RLS algorithm, and $2N$ for the LMS algorithm. The number of additions is somewhat higher. The cost we pay is a processing delay which is of the order of L samples. We have simulated the algorithms and have verified that they work. Preliminary experience appears to indicate that the numerical behavior of the algorithms may require further attention.

Table II: FSU FTF Algorithm

#	Computation	Cost per L samples
1	$\begin{bmatrix} \eta^{H}_{N,L,k} \\ e^{p\,H}_{N,L,k} \\ r^{p\,H}_{N,L,k} \\ -\widehat{d}^{p\,H}_{N,L,k} \end{bmatrix} = \begin{bmatrix} \left[0\ \widetilde{C}_{N,k-L}\right] \\ A_{N,k-L} \\ B_{N,k-L} \\ \left[W_{N,k-L}\ 0\right] \end{bmatrix} X^{H}_{N+1,L,k}$	$(5+4\frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)}+8\mathrm{N}$
2	Schur Algorithm: Input: $\eta_{N,L,k},\ e^{p}_{N,L,k},\ r^{p}_{N,L,k},\ -\widehat{d}^{p}_{N,L,k}$ Output: $\Theta_{k-i}(z)\ \ i=L{-}1,\cdots,0$	$2.5\mathrm{L}^2$
3	$\Theta_{k,L}(z) = \prod_{i=0}^{L-1}\Theta_{k-i}(z)$	$7.5\mathrm{L}^2$
4	$\begin{bmatrix} \widetilde{C}_{k}(z) \\ A_{k}(z) \\ B_{k}(z) \\ W_{k}(z) \end{bmatrix} = \Theta_{k,L}(z) \begin{bmatrix} \widetilde{C}_{k-L}(z) \\ A_{k-L}(z) \\ B_{k-L}(z) \\ W_{N,k-L}(z) \end{bmatrix}$	$(12+4\frac{\mathrm{N+1}}{\mathrm{L}})\mathrm{FFT(2L)}+24\mathrm{N}$
Total cost per sample		$(17+8\frac{\mathrm{N+1}}{\mathrm{L}})\frac{\mathrm{FFT(2L)}}{\mathrm{L}}+32\frac{\mathrm{N}}{\mathrm{L}}+10\mathrm{L}$

References

[1] J.M. Cioffi and T. Kailath. "Fast, recursive least squares transversal filters for adaptive filtering". *IEEE Trans. on ASSP*, ASSP-32(2):304–337, April 1984.

[2] D.T.M. Slock and T. Kailath. "Numerically Stable Fast Transversal Filters for Recursive Least-Squares Adaptive Filtering". *IEEE Trans. Signal Proc.*, ASSP-39(1):92–114, Jan. 1991.

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

[4] 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.

[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] 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.

[7] 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.

[8] Dirk 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". Technical Report RR N$^{\mathrm{O}}$ 92-001, Institut Eurécom, Sophia Antipolis, France, 14 Dec. 1992.