Efficient least squares adaptive algorithms for FIR transversal filtering

Article *in* IEEE Signal Processing Magazine · August 1999

DOI: 10.1109/79.774932 · Source: IEEE Xplore

CITATIONS
234

READS
217

3 authors, including:

G.-O. Glentis
University of Peloponnese
95 PUBLICATIONS **1,084** CITATIONS
SEE PROFILE

S. Theodoridis
Institute of Electrical and Electronics Engineers
350 PUBLICATIONS **8,243** CITATIONS
SEE PROFILE

A Unified View

Efficient Least Squares Adaptive Algorithms For FIR Transversal Filtering

George-Othon Glentis, Kostas Berberidis, and Sergios Theodoridis

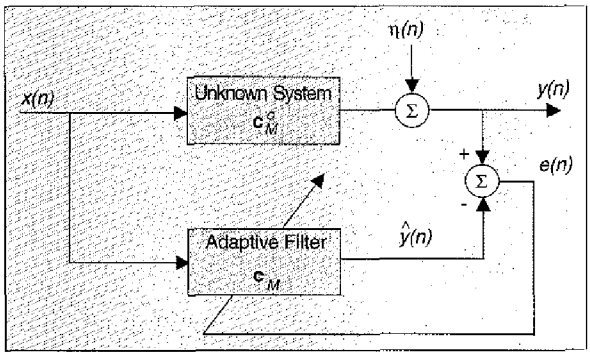
Identifying an unknown system has been a central issue in various application areas such as control, channel equalization, echo cancellation in communication networks and teleconferencing, geophysical signal processing, and many others. Identification is the procedure of specifying the unknown model in terms of the available experimental evidence, that is, a set of measurements of a) the input-output desired response signals, and b) an appropriately chosen error cost function that is optimized with respect to the unknown model parameters. Adaptive identification refers to a particular procedure where we learn more about the model as each new pair of measurements is received, and we update our knowledge to incorporate the newly received information.

In this review article we focus on a particular type of system—the linear Finite Impulse Response (FIR) system. In other words, the class of models in which the search for the optimum filter is conducted assumes that each output value is determined by a weighted combination of a fixed, finite number, M, of past values of the input signal.

The performance of an algorithm can be measured by a number of factors such as

- The accuracy of the obtained solution with respect to the theoretically expected set up
- Its convergence speed
- Its tracking ability with respect to time-varying statistics
- Its computational complexity
- Its robustness to round off error accumulation

1053-5888/99/$10.00©1999IEEE



▲ *1. The FIR systems identification setup.*

Parallelism and pipelining in the computational flow can also be performance-related issues when multiprocessor machines are utilized.

Over the last three decades, a wealth of algorithms have been designed, around the squared error cost function, with a goal to achieve the "best" tradeoff (from a practical point of view) among the above performance factors. The least mean squares (LMS) and recursive least squares (RLS) [10], [11] schemes are the most celebrated examples from this list of "algorithmic happening."

The goal of this review article is to present in a unified and systematic way the most well-known and widely used least-squares (LS) transversal adaptive algorithms. Algorithms such as the LMS, transform-domain LMS, RLS, quasi-newton RLS, fast-Newton, affine-projection, block-approximate implementations, and block-exact implementations are seen as offsprings of a single generic recursion. Such a view makes the trip around all these algorithms easier, and at the same time, reveals the underlying affinity between all these schemes. Furthermore, an extensive list of related references is provided, including the most recent of these algorithms.

As expected, we could not include all the existing schemes. We focused on the basic philosophies and chose to present typical examples.

This overview is organized as follows. The basic LS FIR filtering setup is formulated in "The Wiener Filter." Next, we introduce the concept of stochastic approximation. Different adaptive algorithms are derived in "Adaptive Gradient Algorithms," "Accelerating the Adaptive Gradient Methods," "Adaptive Gauss-Newton Algorithm," and "Adaptive Quasi-Newton Algorithms," by applying a stochastic approximation procedure, either to the steepest descent or to the Gauss-Newton method. We then consider fast-adaptive transversal algorithms in the next section. Fast-block-adaptive algorithms and block-exact-adaptive schemes are presented in "Block Adaptive Filtering." Finally, the performance of some of the most representative adaptive algorithms is evaluated in "Simulation Results," by means of computer simulations, in the context of the system identification.

The following notation is used throughout the article: scalar quantities are denoted as lowercase math, while vector and matrix quantities are denoted as lowercase bold roman and uppercase bold roman respectively.

The Wiener Filter

Adaptive filtering and system-identification algorithms deal with the estimation of a set of parameters of a model of an unknown plant, which, once estimated, gives sufficient information about the system dynamics. Linear system parameterization is an important class of system modeling with a wide area of applications. The most popular among the class of linear models is the one with an FIR structure. This restriction is either intrinsic in the physical system modeling or it is imposed in order to simplify the estimation task and to reduce the computational load in real-time applications. Both static and dynamic FIR models have been considered in the past, and a great variety of algorithms has been proposed for the efficient estimation of the model parameters, [1]-[14].

The system model is described by the difference equation

$$y(n)=\sum_{i=1}^{M} c_i^o\, x(n-i+1)+\eta(n) \tag{1}$$

where $x(n)$ is the input signal, $y(n)$ is the output signal, and c_i^o, and $i=1,\ldots M$, are the filter coefficients. Signal $\eta(n)$ is a disturbance signal. The above equation is compactly written as

$$y(n)=\mathbf{x}_M^T(n)\mathbf{c}_M^o+\eta(n) \tag{2}$$

where

$$\mathbf{x}_M(n)=[x(n)\; x(n-1)\ldots x(n-M+1)]^T \tag{3}$$

is the regressor, or the data vector, of dimensions $M\times 1$, and

$$\mathbf{c}_M^o=[c_1^o c_2^o \ldots c_M^o]^T \tag{4}$$

is the filter coefficients vector, of dimensions $M\times 1$.

The FIR estimator for the system (1) is defined by

$$\hat{y}(n)=\mathbf{x}_M^T(n)\mathbf{c}_M \tag{5}$$

where $\mathbf{c}_M=[c_1 c_2 \ldots c_M]^T$ is an estimate of the system parameters $\mathbf{c}_M^o$. In this case, the estimation error is expressed as the difference between the measured and the predicted system output,

$$e(n)=y(n)-\hat{y}(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M \tag{6}$$

The FIR system identification set up is illustrated in Fig. 1. Given a sequence of an input signal $x(n)$ and a desired measured output signal $y(n)$, the optimum parameters of the FIR model of (5) are estimated by minimizing the mean-squared-error (MSE) criterion

$$V(\mathbf{c}_M)=\mathcal{E}[e^2(n)]=\mathcal{E}[(y(n)-\hat{y}(n))^2] \tag{7}$$

where $\mathcal{E}[\cdot]$ denotes the expectation operator. Thus,

$$\mathbf{c}_M = \arg\min_{\mathbf{c}_M} V(\mathbf{c}_M) = \arg\min_{\mathbf{c}_M} \mathcal{E}[(y(n) - \mathbf{x}_M^T(n)\mathbf{c}_M)^2] \quad (8)$$

The cost function (7) is a quadratic functional of the form

$$V(\mathbf{c}_M) = \mathcal{E}[y^2(n)] + \mathbf{c}_M^T \mathbf{R}_M \mathbf{c}_M - 2\mathbf{d}_M^T \mathbf{c}_M \quad (9)$$

$\mathbf{R}_M$ is the autocorrelation matrix of the input signal $x(n)$, and $\mathbf{d}_M$ is the cross-correlation vector between the input signal $x(n)$ and the system desired output signal $y(n)$, respectively, i.e.,

$$\mathbf{R}_M = \mathcal{E}[\mathbf{x}_M(n)\mathbf{x}_M^T(n)], \quad \mathbf{d}_M = \mathcal{E}[\mathbf{x}_M(n)y(n)] \quad (10)$$

Minimization of the cost function (7), with respect to the filter coefficients vector, is obtained by setting the gradient of $V(\mathbf{c}_M)$ equal to 0, i.e.,

$$\nabla V(\mathbf{c}_M) = \mathbf{0} \rightarrow \mathbf{R}_M \mathbf{c}_M - \mathbf{d}_M = 0 \quad (11)$$

The above condition gives rise to a set of linear equations, the so-called *normal equations*

$$\mathbf{R}_M \mathbf{c}_M = \mathbf{d}_M. \quad (12)$$

The minimum MSE attained is given by

$$E_M^{\min} = \mathcal{E}[y^2(n)] - \mathbf{d}_M^T \mathbf{c}_M. \quad (13)$$

Linear prediction can be cast as a special case of FIR filtering. Let us assume that the signal $x(n)$ is described by the auto-regressive (AR) model equation

$$\sum_{i=0}^{M} a_i^0 x(n-i) = e(n). \quad (14)$$

Here, a_0^0 is set equal to one, i.e., $a_0^0 = 1$, and $e(n)$ is the driving noise signal. The forward predictor for the system (14) is given by

$$\hat{x}(n) = -\mathbf{x}_M^T(n)\mathbf{a}_M \quad (15)$$

The most popular among the class of linear models is the one with an FIR structure.

where, $\mathbf{a}_M = [a_1 a_1 \ldots a_M]^T$ is a vector of dimensions $M \times 1$ that carries the forward predictor coefficients. In this case, the optimum forward predictor, in the Wiener sense, is obtained as the solution of the normal equations

$$\mathbf{R}_M \mathbf{a}_M = -\mathbf{r}_M^f \quad (16)$$

where $\mathbf{r}_M^f = \mathcal{E}[\mathbf{x}_M(n-1)x(n)]$. The backward predictor for the system (14) is treated in a similar way.

Application of either direct or iterative methods for the solution of (12) and (16), can be effectively utilized when the joint statistics of the input and desired output signals, $x(n)$ and $y(n)$, are known in advance. When only measurements are available, stochastic approximation methods come into the scene.

Stochastic Approximation Methods

The typical problem for stochastic optimization can be stated as follows [15]-[21],

$$\text{Minimize } V(\mathbf{c}_M) = \mathcal{E}[Q(\mathbf{c}_M, \eta)] \quad (17)$$

where $\mathbf{c}_M \in \Re^M$ is the optimum parameter sought. $Q(\mathbf{c}_M, \eta)$ is a function whose distribution is unknown. Assuming ergodicity, two basic directions have been suggested for the stochastic approximation of a stochastic optimization problem: the *non-recursive* and *recursive* methods, [15]. In the first case, sampled data of the random variable are used and an approximation of (17) is adopted, i.e.,

$$\text{Minimize } V_N(\mathbf{c}_M) = \frac{1}{N}\sum_{n=0}^{N}(Q(\mathbf{c}_M, \eta(n)). \quad (18)$$

Table 1. The LMS and Its Basic Variants.

Initialization
$\mathbf{c}_M(-1) = \mathbf{0}$

$$e(n) = y(n) - \mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \quad (1)$$

$$\mathbf{c}_M(n) = \mathbf{c}_M(n-1) + \mu(n)\mathbf{x}_M(n)e(n) \quad (2)$$

- $\mu(n)$ constant, the LMS algorithm
- $\mu(n) = \dfrac{\alpha}{\beta + \mathbf{x}_M^T(n)\mathbf{x}(n)}$, $\alpha \in (0,2)$, $0 \le \beta$, the Normalized LMS (NLMS) algorithm
- $\mu(n) = \dfrac{a}{\sigma_x^2(n)}$, $\sigma_x^2(n) = \lambda\sigma_x^2(n-1) + e_M^2(n)$, $\lambda \in (0,1]$, $0 < \alpha < 2/M$, the Power Normalized LMS (PNLMS) algorithm

Recursive stochastic approximation methods result from appropriate modifications of various iterative deterministic optimization algorithms.

In this way, the stochastic optimization problem is transformed into a deterministic one. The solution of (18) is used to approximate the solution of the original problem (17). Application of the non-recursive stochastic approximation method to the system identification problem gives rise to the familiar least-squares estimation

$$\mathbf{c}_M(N)=\arg\min_{\mathbf{c}_M} V_N(\mathbf{c}_M) =\arg\min_{\mathbf{c}_M}\frac{1}{N}\sum_{n=0}^{N}(y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M)^2. \tag{19}$$

The non-recursive stochastic approximation method is a batch-processing algorithm. This is a major drawback for many real-time applications because a large amount of data has to be collected and stored in advance. Alternative ways to alleviate this difficulty are the recursive stochastic approximation schemes, which update the estimator of the optimum parameters whenever new data are available.

Recursive stochastic approximation methods result from appropriate modifications of various iterative deterministic optimization algorithms. Iterative deterministic optimization schemes require the knowledge of either the cost function $V(\mathbf{c}_M)$ and/or its gradient $\nabla V(\mathbf{c}_M)$, and/or its Hessian matrix $\nabla^2 V(\mathbf{c}_M)$, [22]-[23]. The *stochastic approximation counterpart* of a deterministic optimization algorithm is obtained if the cost function, the gradient, and the Hessian, are replaced by unbiased estimates, i.e., $\hat{V}(\hat{\mathbf{c}}_M)$, $\nabla\hat{V}(\hat{\mathbf{c}}_M)$, and $\nabla^2\hat{V}(\hat{\mathbf{c}}_M)$, respectively.

Descent algorithms are perhaps the simplest deterministic optimization methods. The recursive estimator has the form

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}+\mu_i\mathbf{v}_M^i. \tag{20}$$

At each iteration step, i, the update of the estimate is performed along the direction of $\mathbf{v}_M^i$. Variable μ_i regulates the effect of the update on the current value of the estimate. In some cases, it is chosen so that the cost function at step i is minimized, using a line search optimization method, [15],[22]

$$\mu_i=\arg\min_{\mu} V(\mathbf{c}_M^{i-1}+\mu\mathbf{v}_M^i). \tag{21}$$

The most common choice of $\mathbf{v}_M^i$ is the direction determined by the negative gradient of the cost function

$$\mathbf{v}_M^i=-\nabla V(\mathbf{c}_M^{i-1}), \tag{22}$$

which results in the *gradient* or *steepest descent algorithm*. It has the form

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}-\mu_i\nabla V(\mathbf{c}_M^{i-1}). \tag{23}$$

The *Newton-Raphson* method attempts to improve the performance of the steepest descent method by using a properly chosen weighting matrix

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}-\mu_i\mathbf{W}_M^i\nabla V(\mathbf{c}_M^{i-1}). \tag{24}$$

The simplest form of the Newton-Raphson algorithm utilizes the inverse Hessian as a weighting matrix, i.e.,

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}-\mu_i[\nabla^2 V(\mathbf{c}_M^{i-1})]^{-1}\nabla V(\mathbf{c}_M^{i-1}), \tag{25}$$

thus forcing the correction direction to point to the minimal point, in case of quadratic cost functions, such as (9). The step factor μ_i can be fixed, (usually set to $\mu_i=1$), or be estimated by a line search optimization.

Whenever the Hessian is not positive definite or invertible, other suitable weighting directions can be applied. Among them, the *Levenberg-Marquardt* method, suggests the form

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}-\mu_i[\nabla^2 V(\mathbf{c}_M^{i-1})+\delta\mathbf{I}_M]^{-1}\nabla V(\mathbf{c}_M^{i-1}), \tag{26}$$

where $\delta>0$ and is selected so that the weighted matrix is positive definite. In some other cases, where the Hessian is very cumbersome to calculate, an approximation, $\mathbf{A}_M^i$, may be used instead, i.e.,

$$\mathbf{c}_M^i=\mathbf{c}_M^{i-1}-\mu_i[\mathbf{A}_M^i]^{-1}\nabla V(\mathbf{c}_M^{i-1}). \tag{27}$$

In the later cases, the resulting optimization methods are known as *Quasi-Newton* methods. $\mathbf{W}_M^i$ may also be set equal to a constant matrix. These methods which are apparently a special case of the Quasi-Newton approach, are often called *preconditioning methods*.

For all the above deterministic iterative optimization schemes, a number of stochastic approximation algorithms exist. It suffices to replace the cost-function-related terms with appropriate approximate values, which are computed iteratively as each new set of input/output samples $(x(n),y(n))$ is received. Thus, in the stochastic approximation schemes iteration steps coincide with time updates. Summarizing, the recursive stochastic approximation algorithm may be cast as follows

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)-\frac{1}{2}\mu(n)\mathbf{W}_M(n)\mathbf{g}_M(n), \tag{28}$$

where $\mathbf{c}_M(n)$ denotes the estimate of the unknown system parameters vector at time instant n. $\mathbf{g}_M(n)$ is an estimate of the gradient at time instant n

$$\mathbf{g}_M(n)\equiv\nabla\hat{V}(\mathbf{c}_M)=\nabla_{\mathbf{c}_M(n-1)}(\hat{\mathcal{E}}_n[e^2(n)]), \tag{29}$$

where the expectation operator $\mathcal{E}[\cdot]$ has been replaced by a time average estimate $\hat{\mathcal{E}}_n[\cdot]$. $\mathbf{W}_M(n)$ is a (time-dependent) weighting matrix, usually set to be the estimate of the inverse Hessian or its approximation. The factor $\frac{1}{2}$ is used merely for convenience, [8]. Of course, all these substitutions raise important questions concerning convergence issues of the recursive equation. Such issues are not considered here.

Time-adaptive algorithms designed for the estimation of the optimum FIR filter parameters by minimizing (7), can be interpreted as special cases of the basic recursive stochastic approximation scheme of (28).

Adaptive-Gradient Algorithms

Adaptive-gradient algorithms result from the basic scheme dictated by (28), by setting $\mathbf{W}_M(n)=\mathbf{I}_M$. Thus,

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)-\frac{1}{2}\mu(n)\mathbf{g}_M(n). \tag{30}$$

Depending on the choice of the stochastic approximation, several known adaptive methods are derived.

Memoryless Approximation of the Gradient

A popular approximation of the averaging operator, originally introduced by Windrow-Hoff in the early 1960s, is a memoryless estimation based only on the current data point, [7]. Thus,

$$\hat{V}(\mathbf{c}_M)=e^2(n), \tag{31}$$

which results in the gradient estimate

$$\mathbf{g}_M(n)=-2\mathbf{x}_M(n)e(n), \tag{32}$$

where

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \tag{33}$$

is known as the a priori filtering error.

Thus, the celebrated LMS algorithm results, [7], [8]. The algorithm is listed in Table 1. It is perhaps the most popular method for adaptive filtering and system identification, mainly due to the simplicity of its underlying structure and the extensive theoretical analysis and experimental verification of the fundamental properties of the method (convergence, tracking ability, and robustness). The algorithm is trimmed by the so-called "step-size" factor μ. The LMS algorithm converges in the mean to the optimum solution (12) when the step-size factor μ is restricted to be inside the interval,

$$0<\mu<\frac{2}{\lambda_{\max}(\mathbf{R}_M)} \tag{34}$$

where $\lambda(\mathbf{R}_M)$ denotes an eigenvalue of the autocorrelation matrix.

When μ is optimized as dictated by (21), the *normalized* LMS algorithm results, [8]. In this case, μ is time varying and is given by

$$\mu(n)=\frac{1}{\mathbf{x}_M^T(n)\mathbf{x}_M(n)}. \tag{35}$$

The normalized LMS algorithm also allows for a deterministic interpretation, as the filter that minimizes the error norm

$$\mathbf{c}_M(n)=\min_{\mathbf{c}_M}\|\mathbf{c}_M-\mathbf{c}_M(n-1)\|^2 \tag{36}$$

subject to the constraint imposed by the model (5), i.e.,

$$y(n)=\mathbf{x}_M^T(n)\mathbf{c}_M.$$

In this context, the NLMS algorithm is also known as *the projection algorithm*, [2]. A more robust algorithm may result by replacing the optimum step size by a slightly different formula,

$$\mu(n)=\frac{\alpha}{\beta+\mathbf{x}_M^T(n)\mathbf{x}_M(n)},\ \alpha\in(0,2),\ 0\le\beta. \tag{37}$$

The presence of β guarantees that the denominator never becomes zero, while α is a relaxation factor. Another variant, known as the *power-normalized* LMS, [1]-[3], results by setting

$$\mu(n)=\frac{\alpha}{\sigma_x^2(n)}, \tag{38}$$

where $\sigma_x^2(n)$ is the power of the input signal. For stationary signals with power σ_x^2, $\mu(n)$ can be constant, and in

Table 2. The Sliding-Window LMS (SW-LMS) Algorithm.

Initialization
$\mathbf{c}_M(-1)=0$

$$\mathbf{X}_{M,L}(n)=[\mathbf{x}_M(n)\mathbf{x}_M(n-1)\ldots\mathbf{x}_M(n-L+1)] \tag{1}$$

$$\mathbf{y}_L(n)=[y(n)y(n-1)\ldots y(n-L+1)]^T \tag{2}$$

$$\mathbf{e}_L(n)=\mathbf{y}_L(n)-\mathbf{X}_{M,L}^T(n)\mathbf{c}_M(n-1)$$

$$\tilde{\mathbf{g}}_M(n)=\mathbf{X}_{M,L}(n)\mathbf{e}_L(n) \tag{3}$$

$$\tilde{\mu}(n)=\frac{L\tilde{\mathbf{g}}_M^T(n)\tilde{\mathbf{g}}_M(n)}{2\tilde{\mathbf{g}}_M^T(n)\mathbf{X}_{M,L}(n)\mathbf{X}_{M,L}^T(n)\tilde{\mathbf{g}}_M(n)} \tag{4}$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\tilde{\mu}(n)\tilde{\mathbf{g}}_M(n) \tag{5}$$

Table 3a. The Decorrelated LMS Algorithm. ρ Is a Trimming Factor.

Initialization
$\mathbf{c}_M(-1)=\mathbf{0}$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \quad (1)$$

$$\alpha(n)=\frac{\mathbf{x}_M^T(n)\mathbf{x}_M(n-1)}{\mathbf{x}_M^T(n-1)\mathbf{x}_M(n-1)} \quad (2)$$

$$\mathbf{v}_M(n)=\mathbf{x}_M(n)-\alpha(n)\mathbf{x}_M(n-1) \quad (3)$$

$$\mu(n)=\frac{\rho e(n)}{\mathbf{x}_M^T(n)\mathbf{v}_M(n)} \quad (4)$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{v}(n) \quad (5)$$

Table 3b. The Filtered-x LMS Algorithm.

Initialization
$\mathbf{c}_M(-1)=0$

Given an estimate of the forward predictor $\mathbf{a}_M(n)$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \quad (1)$$

$$\mathbf{e}_M=[e(n)e(n-1)\ldots e(n-M+1)]^T \quad (2)$$

$$e^f(n)=x(n)+\mathbf{a}_M^T(n)\mathbf{x}_M(n-1) \quad (3)$$

$$\mathbf{e}_M^f(n)=[e^f(n)e^f(n-1)\ldots e^f(n-M+1)]^T \quad (4)$$

$$\tilde{e}(n)=e(n)+\mathbf{a}_M^T(n)\mathbf{e}_M(n) \quad (5)$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu e_M^f(n)\tilde{e}_M(n) \quad (6)$$

this case, $0<\alpha<\frac{2}{M}$. The normalized LMS (N-LMS) and the power-normalized LMS (PN-LMS) algorithms are listed in Table 1. The basic advantage of these algorithms is the faster convergence rate compared to the original method.

The LMS algorithm as described in Table 1, requires M operations for the estimation of the a priori filtering error, and M additional operations for the update of the filter parameters vector. Thus, a total of $2M+1$ operations is required per processed sample. More efficient schemes have been proposed recently [106], that perform block (exact) adaptive filtering at a lower complexity per processed sample, at the expense of an input-output delay caused by the specific block implementation of these methods, [98], [106]-[108], [135] - [137]. Finally, the NLMS requires $3M+1$ operations while the power-normalized LMS needs $2M+2$ operations only. Further computational simplifications to the LMS algorithm have also been proposed, in order to reduce its complexity. The *quantized-error* LMS applies a quantization to the error signal $e(n)$. The quantized form of the error, $q[e(n)]$ is now a discrete value function. The simplest form of the quantized LMS is the *sign* LMS, which uses three levels, $-1,0,1$ for the error. If $\mu(n)$ is chosen to be a power-of-two number, then the coefficient updating can be implemented without multiplications, using only bit shifts and additions, [27].

The convergence rate of all the above algorithms is heavily dependent on the eigenvalue distribution of the autocorrelation matrix of the input signal. Thus these algorithms converge at unacceptably low rates when the input signal is colored noise or speech.

Sliding-Window Approximation of the Expectation

An alternative to the time-averaging operator is the incorporation of a sliding-window approximation of the form, [24]-[33],

$$\mathcal{E}_n[\cdot]=\frac{1}{L}\sum_{k=n-L+1}^{n}[\cdot] \quad (39)$$

L determines the memory of the approximator and may be *smaller*, *equal*, or *greater* than the filter order M. Thus,

$$\hat{V}(\mathbf{c}_M)=\frac{1}{L}\sum_{k=n-L+1}^{n}[e^2(n)], \quad (40)$$

which results in the gradient estimate

$$\mathbf{g}_M(n)=-\frac{2}{L}\sum_{k=n-L+1}^{n}\mathbf{x}_M(k)(y(k)-\mathbf{x}_M^T(k)\mathbf{c}_M(n-1)). \quad (41)$$

Consider the input data matrix of dimensions $M\times L$

$$\mathbf{X}_{M,L}(n)=[\mathbf{x}_M(n)\mathbf{x}_M(n-1)\ldots\mathbf{x}_M(n-L+1)] \quad (42)$$

and the desired response data vector, of dimensions $L\times 1$

$$\mathbf{y}_L(n)=[y(n)y(n-1)\ldots y(n-L+1)]^T. \quad (43)$$

Using (42) and (43), (41) takes the form

$$\mathbf{g}_M(n)=-\frac{2}{L}\mathbf{X}_{M,L}(n)\mathbf{e}_L(n), \quad (44)$$

where

Table 3c. The Transform-Domain LMS (TD-LMS) Algorithm.

Initialization
$\hat{\mathbf{c}}_M(-1)=0$

Given a unitary transform matrix $\mathbf{S}_M$

$$\mathbf{u}_M(n)=\mathbf{S}_M\mathbf{x}_M(n) \quad (1)$$

$$e(n)=y(n)-\mathbf{u}_M^H(n)\hat{\mathbf{c}}_M(n-1) \quad (2)$$

$$\hat{\mathbf{c}}_M(n)=\hat{\mathbf{c}}_M(n-1)+\mu(n)\mathbf{u}_M(n)e(n) \quad (3)$$

$$\mathbf{e}_L(n)=\mathbf{y}_L(n)-\mathbf{X}_{M,L}^T(n)\mathbf{c}_M(n-1). \quad (45)$$

If we plug the above equations into the general formulation of the stochastic approximation's steepest descent algorithm of (30), a stochastic gradient algorithm, called thereafter, the *sliding-window* LMS algorithm (SW-LMS), of Table 2 appears, where we have set

$$\tilde{\mathbf{g}}_M(n)=\tfrac{L}{2}\mathbf{g}_M(n).$$

The $\tilde{\mu}(n)$ in Table 2, is the result of a line-search procedure, e.g., (21). A special case of the SW-LMS algorithm appears when the step size factor is set constant, i.e, $\mu(n)=\mu\ \forall n$.

The SW-LMS algorithm has better convergence properties compared to the memoryless time-averaging approximation algorithms of Table 1, [30]. However, the computational complexity is now increased by a factor of L, being $4LM+3L$. This hinders the use of the algorithm in real-time applications, unless efficient schemes for the computation of (44) and (45) are adopted by exploiting the special structure of the involved matrices, e.g., [30]. Furthermore, this algorithm is directly related to the recently proposed, fast implementation of the affine projection algorithm, considered in "The Fast-Affine Projection Algorithm."

Table 3d. The Power-Normalized TD-LMS (PN-TD-LMS) Algorithm.

Initialization
$\hat{\mathbf{c}}_M(-1)=0$

Given a unitary transform matrix $\mathbf{S}_M$

$$\mathbf{u}_M(n)=\mathbf{S}_M\mathbf{x}_M(n) \quad (1)$$

$$e(n)=y(n)-\mathbf{u}_M^H(n)\hat{\mathbf{c}}_M(n-1) \quad (2)$$

$$\mathbf{R}_{TD}(n)=\lambda\mathbf{R}_{TD}(n-1)+(1-\lambda)\mathrm{diag}(|u_1(n)|^2\ldots|u_M(n)|^2),\ \lambda\in(0,1) \quad (3)$$

$$\hat{\mathbf{c}}_M(n)=\hat{\mathbf{c}}_M(n-1)+\mu(n)\mathbf{R}_{TD}^{-1}(n)\mathbf{u}_M(n)e(n) \quad (4)$$

Exponentially Forgetting Approximation of the Expectation

Introducing a time-averaging operator having an exponential-forgetting form,

$$\hat{\mathcal{E}}_n[\cdot]=\sum_{k=0}^{n}\lambda^{n-k}[\cdot]\ 0<\lambda\le 1 \quad (46)$$

another variant of the LMS can be produced, the *exponentially forgetting window* EFW-LMS). In this case,

$$\hat{V}(\mathbf{c}_M)=\sum_{k=0}^{n}\lambda^{n-k}[e^2(n)]. \quad (47)$$

This approximation results in the gradient estimate

$$\mathbf{g}_M(n)=-2(\mathbf{d}_M(n)-\mathbf{R}_M(n)\mathbf{c}_M(n-1)), \quad (48)$$

where

$$\mathbf{R}_M(n)=\sum_{k=0}^{n}\lambda^{n-k}\mathbf{x}_M(k)\mathbf{x}_M^T(k)$$

and

$$\mathbf{d}_M(n)=\sum_{k=0}^{n}\lambda^{n-k}\mathbf{x}_M(k)y(k). \quad (49)$$

Table 3e. The Power-Normalized Sliding-Window TDLMS Algorithm.

Initialization
$\hat{\mathbf{c}}_M(-1)=0$

Given a unitary transform matrix $\mathbf{S}_M$

$$\mathbf{u}_M(n)=\mathbf{S}_M\mathbf{x}_M(n) \quad (1)$$

$$\mathbf{U}_{M,L}(n)=[\mathbf{u}_M(n)\mathbf{u}_M(n-1)\ldots\mathbf{u}_M(n-L+1)] \quad (2)$$

$$\mathbf{y}_L(n)=[y(n)y(n-1)\ldots y(n-L+1)]^T \quad (3)$$

$$\mathbf{e}_L(n)=\mathbf{y}_L(n)-\mathbf{U}_{M,L}^H(n)\hat{\mathbf{c}}_M(n-1) \quad (4)$$

$$\tilde{\mathbf{g}}_M(n)=\mathbf{U}_{M,L}(n)\mathbf{e}_L(n) \quad (5)$$

$$\mathbf{R}_{TD}(n)=\lambda\mathbf{R}_{TD}(n-1)+(1-\lambda)\mathrm{diag}(|u_1(n)|^2\ldots|u_M(n)|^2),\ \lambda\in(0,1) \quad (6)$$

$$\hat{\mathbf{c}}_M(n)=\hat{\mathbf{c}}_M(n-1)+\mu(n)\mathbf{R}_{TD}^{-1}(n)\mathbf{u}_M(n)e(n) \quad (7)$$

Accelerating The Adaptive-Gradient Methods

In an attempt to accelerate the convergence speed of the adaptive-gradient algorithms, which are deeply affected by the eigenvalue spread $\lambda_{max}/\lambda_{min}$ of the input autocorrelation matrix, while, at the same time, keeping the computational complexity at a low level, several methods have been proposed that work on input data that have been preconditioned, e.g., pre-whitened or decorrelated, [7], [34]-[35], [36].

Time-Domain Decorrelation

Among others, one way to achieve this task is to use the decorrelation method proposed in [35]. Consider the descent algorithm of (20). A stochastic approximation of this scheme can be obtained by choosing

$$\mathbf{v}_M(n)=\mathbf{x}_M(n)-a(n)\mathbf{x}_M(n-1), \tag{50}$$

where

$$a(n)=\frac{\mathbf{x}_M^T(n)\mathbf{x}_M(n-1)}{\mathbf{x}_M^T(n-1)\mathbf{x}_M(n-1)} \tag{51}$$

is the decorrelation coefficient between $\mathbf{x}_M(n)$ and $\mathbf{x}_M(n-1)$ at time instant n. Thus, the filter updating recursion takes the form

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{v}_M(n). \tag{52}$$

The optimum step size μ is obtained by minimizing (21), which finally results in

$$\mu(n)=\frac{e(n)}{\mathbf{x}_M^T(n)\mathbf{v}_M(n)}. \tag{53}$$

This method can be viewed as an adaptive instrumental variable method, [1]-[3], where the instrument is defined by (50). The algorithm is presented in Table 3a. The improved convergence properties of the method are discussed in [35].

The method of [34] goes a step further. The forward prediction error of order M is utilized to construct the instrument as

$$\mathbf{v}_M(n)\equiv\mathbf{e}_M^f(n)=[e^f(n)e^f(n-1)\ldots e^f(n-M+1)]^T \tag{54}$$

where

$$e^f(n)=x(n)+\mathbf{a}_M^T(n)\mathbf{x}_M(n-1) \tag{55}$$

is the M-th order forward-prediction error of the input data signal, and $\mathbf{a}_M(n)$ is an estimate of the forward predictor at time instant n. Moreover, the instantaneous error $e(n)$ is filtered by the forward predictor at time n, i.e.,

$$\tilde{e}(n)=e(n)+\mathbf{a}_M^T(n)\mathbf{e}_M(n),$$

where

$$\mathbf{e}_M(n)=[e(n)e(n-1)\ldots e(n-M+1)]^T.$$

The forward predictor $\mathbf{a}_M(n)$ is adaptively estimated by means of an LMS algorithm. The algorithm, known as *filtered-x* LMS, is tabulated in Table 3b. Other decorrelation and/or pre-whitening techniques for improving the convergence of the LMS algorithm are discussed in [36].

The computational complexity of the decorrelation LMS type algorithms that appear in Table 3, is $\mathcal{O}(M)$, [37]-[54].

Transform-Domain Decorrelation

An early attempt to improve the performance of the LMS algorithm was the use of unitary transformations on the input data vector, $\mathbf{x}_M(n)$. The resulting algorithms may have increased the convergence rate for some classes of input signals, yet the computational complexity remains similar to that of the original LMS scheme. The algorithmic family established and the variations followed are known as the *transform-domain-adaptive-filtering* algorithms, [37]-[54].

The transform-domain algorithm can be interpreted as a special case of (28). A properly chosen fixed-weighting matrix is utilized to decorrelate, or precondition, the direction vector. Unitary transforms, such as the discrete Fourier transform, the discrete cosine transform and the discrete Hartley transform have been efficiently utilized in the past to speed up the convergence of the LMS algorithm. This preprocessing followed by normalization reduces the eigenvalue spread of the input

Table 4. The Exponential Forgetting-Window RLS.

Initialization

$\mathbf{c}_M(-1)=0$, $\mathbf{R}_M^{-1}(-1)=\mathbf{I}_M\delta,\delta>0$

$$\mathbf{w}_M(n)=\lambda^{-1}\mathbf{R}_M^{-1}(n-1)\mathbf{x}_M(n) \tag{1}$$

$$\alpha_M(n)=1+\mathbf{w}_M^T(n)\mathbf{x}_M(n) \tag{2}$$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \tag{3}$$

$$\epsilon(n)=\frac{e(n)}{\alpha_M(n)} \tag{4}$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mathbf{w}_M(n)\epsilon(n) \tag{5}$$

$$\mathbf{R}^{-1}(n)=\frac{1}{\lambda}\mathbf{R}^{-1}(n-1)-\frac{\mathbf{w}_M^T(n)\mathbf{w}_M(n)}{\alpha_M(n)} \tag{6}$$

signal autocorrelation matrix, and thus, affects the convergence speed of the LMS algorithm. The performance of these methods depends on the degree of the orthogonalization achieved when the unitary transformations are applied to the underlying data. Theoretical results, as well as experimental evidence, is given in [37]-[54].

Let $\mathbf{S}_M$ be a unitary transform (within a constant scalar) of order M, i.e.,

$$\mathbf{S}_M \mathbf{S}_M^H = \beta \mathbf{I}_M \tag{56}$$

H stands for the Hermitian operator (conjugate and transpose). Then, the a priori filtering error of (33) can be rewritten as

$$e(n) = y(n) - \mathbf{u}_M^H(n)\hat{\mathbf{c}}_M(n-1), \tag{57}$$

where

$$\mathbf{u}_M(n) = \mathbf{S}_M \mathbf{x}_M(n) \tag{58}$$

is the transformed input data vector, and

$$\hat{\mathbf{c}}_M(n-1) = \mathbf{S}_M \mathbf{c}_M(n-1)/\beta. \tag{59}$$

Thus, now the elements of the data vector are not shifts of a single signal sequence, as is the case with the elements of $x(n)$. The elements of the transformed data vector, $\mathbf{u}_M(n)$, can be viewed as a batch of multichannel signals, $u_i(n)$, $i=1,2\ldots M$, which are expected to be less correlated than the original set of shifted signals $x(n-i+1)$. Thus, the single-channel FIR filter of order M has been transformed to an equivalent multichannel form by passing the original input $\mathbf{x}_M(n)$ through a set of M filters, as (58) suggests. When the DFT is used as a unitary transformation, $\mathbf{u}_M(n)$ becomes the sliding-window Fourier transform, [49]. The estimated filter $\hat{\mathbf{c}}_M(n)$ is the frequency response of the time domain counterpart, $\mathbf{c}_M(n)$. Thus, we may say that the adaptation takes place in the frequency domain (i.e., frequency-domain adaptive filtering).

Application of a unitary transformation to the LMS recursion gives rise to the transform-domain LMS (TD-LMS)

$$\hat{\mathbf{c}}_M(n) = \hat{\mathbf{c}}_M(n-1) + \mu(n)\mathbf{u}_M(n)e(n) \tag{60}$$

The algorithm is tabulated in Table 3c. The TD-LMS converges to the same MSE error as the original LMS algorithm, and the coefficients converge to the transformed MSE solution

$$\hat{\mathbf{c}}_M = \mathbf{S}_M \mathbf{c}_M/\beta.$$

A more elaborate version of the TD-LMS is obtained by introducing a power normalization of the transform domain data vector. Let us define the diagonal matrix

$$\mathbf{R}_{TD}(n) = \lambda \mathbf{R}_{TD}(n-1) + (1-\lambda)\mathrm{diag}(|u_1(n)|^2 \ldots |u_M(n)|^2), \tag{61}$$

where u_i^2 denotes the square value of the i-th element of vector $\mathbf{u}_M(n)$ Then,

$$\hat{\mathbf{c}}_M(n) = \hat{\mathbf{c}}_M(n-1) - \mu(n)\mathbf{R}_{TD}^{-1}(n)\mathbf{u}_M(n)e(n). \tag{62}$$

The algorithm, known as the power-normalized-transform-domain LMS (PN-TD-LMS), [49], is tabulated in Table 3d.

The computational complexity of the TD-LMS depends on the way the transformation dictated by (58) is computed. Fast $\mathcal{O}(M\log_2 M)$ algorithms exist for the efficient implementation of the DFT, the DCT and the DHT. When the DFT is utilized, the cost per direct and/or inverse transformation is $\frac{M}{4}\log_2(M)$, when the input signal $x(n)$ is real data, [109]. A more efficient method has recently been proposed for the computation of the SW-DFT with k samples hopping, [116]. For real input data, this figure is $M(\log_2(k)+3)-2k$, which reduces to $3M-2$ for the single-step sliding-window case. Efficient methods for estimating the required transformations by means of partitioning have also been derived, [110]-[123]. Such methods are also discussed in the companion paper [36].

The decorrelation and the transformation techniques applied to accelerate the LMS algorithm can be applied to the SW-LMS as well, to obtain the N-TD-SW-LMS, which is tabulated in Table 3e. Fast convolution methods can also be applied to reduce complexity, [26]-[31], [37]-[54], [110]-[123].

Another approach to improve the performance of the LMS algorithm is based on the application of subband decomposition of the pertinent signals and the adaptive filtering within each subband, [124]-[131]. This is a generalization of the transform-domain adaptive filtering, where, the application of the DFT as a preconditioner can be also interpreted as a filter-band decomposition [49]. Wavelet transform domain adaptive algorithms are presented in [135]. The adaptive subband filtering is discussed in some more detail in [36].

The search for the optimum filter can be accelerated when the gradient vector is properly deviated to the minimum point.

Adaptive Gauss-Newton Algorithm

The search for the optimum filter can be accelerated when the gradient vector is properly deviated to the minimum point, [22], [15], [11]. The adaptive Gauss-Newton algorithm results when the weighted matrix $\mathbf{W}_M(n)$ of (28) is selected to be the inverse of the estimated Hessian of the least squares cost function, i.e.,

Table 5a. The Quasi-Newton Algorithm with Toeplitz Autocorrelation Matrix Approximation.

Initialization

$\hat{\mathbf{c}}_M(-1)=0,\ t_0(-1)=\delta>0, t_i(-1)=0, i=1,2\ldots M-1$

$\mathbf{T}_M(n)=[t_{|i-j|}(n)]_{\substack{i=0,1\ldots M-1\\ j=0,1\ldots M-1}}$

$t_i(n)=\lambda t_i(n-1)+x(n)x(n-i),$

$i=0,1\ldots M-1, \lambda\in(0,1)$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \quad (1)$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{T}_M^{-1}(n)\mathbf{x}_M(n)e(n) \quad (2)$$

$$\mu(n)=\frac{1}{\mathbf{x}_M^T(n)\mathbf{T}_M^{-1}(n)\mathbf{x}_M(n)} \quad (3)$$

$$\mathbf{W}_M(n)=[\nabla^2\hat{V}(\hat{\mathbf{c}}_M)]^{-1}=[\nabla^2_{\mathbf{c}_M(n-1)}(\hat{\mathcal{E}}_n[e^2(n)])]^{-1}, \quad (63)$$

where the expectation operator $\mathcal{E}[\cdot]$ has been replaced by a time average estimate $\hat{\mathcal{E}}[\cdot]$.

Let us consider an exponential forgetting window for the approximation of the expectation operator, i.e., (46). Then the gradient estimate is given by (48) and the Hessian estimate is

$$\nabla^2\hat{V}(\mathbf{c}_M)=\mathbf{R}_M(n)$$

The optimum step size is estimated to be $\mu(n)=1$. Thus,

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mathbf{R}_M^{-1}(n)(\mathbf{d}_M(n)-\mathbf{R}_M(n)\mathbf{c}_M(n-1)), \quad (64)$$

which results in the well-known least-squares algorithm

$$\mathbf{c}_M(n)=\mathbf{R}_M^{-1}(n)\mathbf{d}_M(n), \quad (65)$$

which is the same form obtained if the deterministic least squares approach had been followed.

Equation (65) can be solved in a time-recursive way, resulting in the well known RLS adaptive algorithm [8]. The derivation of the RLS algorithm is based on the low-rank updating properties that the sampled auto-correlation matrix and the sampled cross-correlation vector possess, [8], [10], [11], that is,

$$\mathbf{R}_M(n)=\lambda\mathbf{R}_M(n-1)+\mathbf{x}_M(n)\mathbf{x}_M^T(n) \quad (66)$$

and

$$\mathbf{d}_M(n)=\lambda\mathbf{d}_M(n-1)+\mathbf{x}_M(n)y(n). \quad (67)$$

The RLS algorithm is tabulated in Table 4.

Variable $\mathbf{w}_M(n)$, appearing in Table 4, is the *Kalman gain* vector defined as, [59],

$$\lambda\mathbf{R}_M(n-1)\mathbf{w}_M(n)=\mathbf{x}_M(n) \quad (68)$$

The solution at time n is then estimated by

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mathbf{w}_M(n)\epsilon(n) \quad (69)$$

where $\epsilon(n)=y(n)-\mathbf{x}_M^t(n)\mathbf{c}_M(n)$ is the so-called a posteriori filtering error.

The RLS algorithm of Table 4 is exponentially stable, [64]. However, when it is implemented in finite precision arithmetic, instability effects may be noticed. Special care should be taken to prevent the loss of symmetry and positive definiteness of matrix $\mathbf{R}_M^{-1}(n)$, [65]-[67]. In an attempt to cope with this problem, numerically stable square-root RLS algorithms have been developed for the update of the Kalman gain vector and/or the filtering error, [3], [8], [10]. Among them, Bierman's algorithm propagates the upper triangular factor U and the diagonal factor D of the $UD^{-1}U^T$ factorization of matrix $\mathbf{R}_M^{-1}(n)$. It is well known, [1]-[11] that the inverse of the autocorrelation matrix $\mathbf{R}_M(n)$ of the input signal allows for a $UD^{-1}U^T$ factorization of the form

$$\mathbf{R}_M^{-1}(n)=\mathbf{U}_M(n)\mathbf{D}_M^{-1}(n)\mathbf{U}_M^T(n), \quad (70)$$

where

$$\mathbf{U}_M(n)=\begin{bmatrix} 1 & b_1^1(n) & b_2^1(n) & \ldots & b_{M-2}^1(n) & b_{M-1}^1(n)\\ 0 & 1 & b_2^2(n) & \ldots & b_{M-2}^2(n) & b_{M-1}^2(n)\\ 0 & 0 & 1 & \ldots & b_{M-2}^3(n) & b_{M-1}^3(n)\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \ldots & b_{M-2}^{M-2}(n) & b_{M-1}^{M-2}(n)\\ 0 & 0 & 0 & \ldots & 1 & b_{M-1}^{M-1}(n)\\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \quad (71)$$

and

$$\mathbf{D}_M^{-1}(n)=\mathrm{diag}\left\{\frac{1}{\alpha_0^b(n)},\frac{1}{\alpha_1^b(n)},\ldots,\frac{1}{\alpha_{M-1}^b(n)}\right\} \quad (72)$$

Vectors

$$\mathbf{b}_m(n)=[b_m^1(n)b_m^2(n)b_m^3(n)\ldots b_m^m(n)]^T$$

are the LS backward optimal linear predictors of order m, and scalars $\alpha_m^b(n)$ are the corresponding backward prediction error powers, $m=1,2\ldots M-1$. It can be shown, [10], that Bierman's algorithm propagates these predictors in a time-and-order recursive fashion. The overall complexity is the same as that of the RLS. The matrix factorization of $\mathbf{R}_M(n)$ ($\mathbf{R}_M^{-1}(n)$) and the efficient update of these factors gives rise to another large family of algorithms not discussed in this review. These are known as lattice and QR

LS adaptive schemes, [8]-[10], [71], depending on the particular algorithmic scheme used for the factorization. The distinct characteristic of these algorithms is that they propagate a set of error variables and not the unknown filter parameters vector. Numerical properties of these methods are discussed in [74]. Finally, schemes based on Householder factorization have been suggested, e.g., [72], which, besides numerical robustness, exhibit a high degree of parallelism in their computational flow.

Adaptive Quasi-Newton Algorithms

Toeplitz Autocorrelation Matrix Approximation

Let us consider the basic recursive stochastic estimation scheme of (28) and the memoryless approximation of the expectation operator. If the weighting matrix, $\mathbf{W}_M(n)$, is set equal to the true inverse of the autocorrelation matrix given by (10), then, the Newton-LMS algorithm results, i.e.,

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{R}_M^{-1}\mathbf{x}_M(n)e(n) \tag{73}$$

Clearly, (73) is of academic interest only, unless matrix $\mathbf{R}_M$ is somehow estimated using the input signal statistics, or the input data themselves. Depending on the choice of the estimate of $\mathbf{R}_M$, several algorithms appear, [81]-[84].

Table 5b. The Quasi-Newton Algorithm with Banded Inverse Autocorrelation Matrix Approximation.

Initialization

$\hat{\mathbf{c}}_M(-1)=0$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \tag{1}$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{R}_{M|P}^{-1}(n)\mathbf{x}_M(n)e(n) \tag{2}$$

$$\mathbf{R}_{M|P}^{-1}(n)=\sum_{j=0}^{M-P-1}\tilde{\mathbf{b}}_P(n-j)\tilde{\mathbf{b}}_P^T(n-j)/\alpha_P^b(n-j)+\sum_{j=0}^{P-1}\hat{\mathbf{b}}_j(n)\hat{\mathbf{b}}_j^T(n)/\alpha_j^b(n) \tag{3}$$

$$\tilde{\mathbf{b}}_P(n-j)=\begin{bmatrix}0_j\\ \mathbf{b}_P(n-j)\\ 1\\ 0_{M-P-j-1}\end{bmatrix}\quad\hat{\mathbf{b}}_j(n)=\begin{bmatrix}\mathbf{b}_j(n)\\ 1\\ 0_{M-j-1}\end{bmatrix} \tag{4}$$

$$\mu(n)=\frac{1}{\mathbf{x}_M^T(n)\mathbf{R}_{M|P}^{-1}(n)\mathbf{x}_M(n)} \tag{5}$$

Let,

$$\mathbf{W}_M(n)=\mathbf{T}_M^{-1}(n), \tag{74}$$

where, $\mathbf{T}_M(n)$ is a Toeplitz approximation of $\mathbf{R}_M$, estimated from the data as

$$\mathbf{T}_M(n)=[t_{|i-j|}(n)]_{\substack{i=0,1\ldots M-1\\ j=0,1\ldots M-1}},$$
$$t_i(n)=\lambda t_i(n-1)+x(n)x(n-i),$$
$$i=0,1\ldots M-1,\ \lambda\in(0,1) \tag{75}$$

In this case, the optimum step size is found to be

$$\mu(n)=\frac{1}{\mathbf{x}_M^T(n)\mathbf{T}_M^{-1}(n)\mathbf{x}_M(n)}$$

The above "Quasi-Newton" algorithm, based on the Toeplitz approximation, (T-QN), is listed in Table 5a.

Banded Inverse Autocorrelation Matrix Approximation

In an attempt to reduce the computational complexity of the RLS scheme, several methods that take into account the special structure that the input signal $x(n)$ might possess, appear in [85]-[89]. For example, when the input $x(n)$ is a speech signal, a low-order autoregressive model can be adopted to model its generation. Let P be the order of the AR model of the input signal $x(n)$. Then, it can be shown that the backward predictors of higher than P order, $\mathbf{b}_{P+i}(n)$, $i=1\ldots M-P-1$, appearing in (71), are replaced by time-delayed and zero-padded backward predictors of order P, $\mathbf{b}_P(n)$. The corresponding backward error powers $\alpha_{P+i}^b(n)$, are also replaced by time-delayed versions of $\alpha_P^b(n)$. This stems from an autocorrelation matrix approximation, using a min-max optimum extrapolation of the autocorrelation matrix of

Table 5c. The Affine Projection Algorithm (APA).

Initialization

$\hat{\mathbf{c}}_M(-1)=0$

$$\mathbf{e}_L(n)=\mathbf{y}_L(n)-\mathbf{X}_{M,L}^T(n)\mathbf{c}_M(n-1) \tag{1}$$

$$\mathbf{R}_L(n)\cong\mathbf{X}_{M,L}^T(n)\mathbf{X}_{M,L}(n)+\delta\mathbf{I}=\mathbf{R}_L(n-1)-\mathbf{x}_L(n-L)\mathbf{x}_L^T(n-L)+\mathbf{x}_L(n)\mathbf{x}_L^T(n) \tag{2}$$

$$\mathbf{w}_L(n)=\mathbf{R}_L^{-1}(n)\mathbf{e}_L(n) \tag{3}$$

$$\mathbf{v}_M(n)=\mathbf{X}_{M,L}(n)\mathbf{w}_L(n) \tag{3}$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu\mathbf{v}_M(n) \tag{4}$$

Table 6. Summary of Adaptive Stochastic Approximation Algorithms.

	Expectation Approximation	Cost Function
E1	$\hat{E}_n[\cdot]=\sum_{k=n}^{n}[\cdot]$	$\sum_{k=n}^{n}[e^2(n)]$
E2	$\hat{E}_n[\cdot]=\frac{1}{L}\sum_{k=n-L+1}^{n}[\cdot],\quad 0<L\le p$	$\sum_{k=n-L}^{n}[e^2(n)]$
E3	$\hat{E}_n[\cdot]=\sum_{k=0}^{n}\lambda^{n-k}[\cdot],\quad 0<\lambda\le 1$	$\sum_{k=0}^{n}\lambda^{n-k}[e^2(n)]$

	Search Direction
A1	$\mathbf{v}_M(n)=-\mathcal{g}_M(n)$
A2	$\mathbf{v}_M(n)=\mathbf{x}_M(n)-\alpha(n)\mathbf{x}_M(n-1)$
A3	$\mathbf{v}_M(n)=\mathbf{e}_M^f(n)$
A4	$\mathbf{v}_M(n)=-\mathbf{S}_M\mathbf{g}_M(n)$

	Weighting Matrix
B1	$\mathbf{W}_M(n)=\mathbf{I}_M$
B2	$\mathbf{W}_M(n)=\mathbf{R}_{TD}^{-1}(n)$
B3	$\mathbf{W}_M(n)=\mathbf{R}_M^{-1}(n)$
B4	$\mathbf{W}_M(n)=\mathbf{T}_M^{-1}(n)$
B5	$\mathbf{W}_M(n)=U_{M\|P}(n)(n)\mathbf{D}_{M\|P}^{-1}(n)U_{M\|P}^T(n)$
B6	$\mathbf{W}_M(n)=(\mathbf{X}_M^L(n)\mathbf{X}_M^{Lt}(n)+\delta\mathbf{I})^{\#}$

	Algorithm	Expectation Approximation	Direction	Weighting Matrix
1	NLMS	E1	A1	B1
2	SLMS	E1	A1	B1
3	LMS	E1	A1	B1
4	SW-LMS	E2	A1	B1
5	EFW-LMS	E3	A1	B1
6	D-LMS	E1	A2	B1
7	X-LMS	E1	A3	B1
8	TD-LMS	E1	A4	B1
9	NP-TD-LMS	E1	A4	B2
10	NP-TD-SW-LMS	E2	A4	B2
11	RLS	E3	A1	B3
12	T-QN	E1	A1	B4
13	F-QN	E1	A1	B5
14	APA	E3	A1	B6

order M, from the autocorrelation matrix of order P, [85], [86]. Thus, we have the approximation

$$\mathbf{R}_M^{-1}(n)\approx\mathbf{R}_{M|P}^{-1}(n)=u_{M|P}(n)\mathbf{D}_{M|P}^{-1}(n)u_{M|P}^T(n). \quad (76)$$

Factor $\mathbf{D}_{M|P}^{-1}(n)$ is constructed using the method discussed above. $\mathbf{R}_{M|P}^{-1}(n)$ turns out to be a banded matrix corresponding to an approximation of the input signal $x(n)$ by an AR model of order P, [85]

$$\mathbf{R}_{M|P}^{-1}(n)=\sum_{j=0}^{M-P-1}\tilde{\mathbf{b}}_P(n-j)\tilde{\mathbf{b}}_P^T(n-j)/\alpha_P^b(n-j)+\sum_{j=0}^{P-1}\hat{\mathbf{b}}_j(n)\hat{\mathbf{b}}_j^T(n)/\alpha_j^b(n) \quad (77)$$

$$\tilde{\mathbf{b}}_P(n-j)=\begin{bmatrix}\mathbf{0}_j\\ b_P(n-j)\\ 1\\ \mathbf{0}_{M-P-j-1}\end{bmatrix}\quad \hat{\mathbf{b}}_j(n)=\begin{bmatrix}\mathbf{b}_j(n)\\ 1\\ \mathbf{0}_{M-j-1}\end{bmatrix}.$$

Taking (31) again for the approximation of the cost function, and $\mathbf{W}_M(n)$ as

$$\mathbf{W}_M(n)=\mathbf{R}_{M|P}^{-1}(n) \quad (78)$$

we obtain the quasi-Newton algorithm developed in [85]. In this case

$$\mu(n)=\frac{1}{\mathbf{x}_M^T(n)\mathbf{R}_{M|P}^{-1}(n)\mathbf{x}_M(n)}.$$

Alternatively, $\mu(n)$ may be kept constant. The algorithm is tabulated in Table 5b. A fast version of this scheme, known as the "fast-Newton" adaptive filtering algorithm, has been developed in [85]. Other, low complexity, quasi-Newton adaptive algorithms have been developed by proper choice of the weight matrix $\mathbf{W}_M(n)$, [88], [89].

Pseudo-Inverse Autocorrelation Matrix Approximation

Let us now consider the time averaging approximation of (39), and the resulting equations for the residual error, i.e., (41)-(45). A stochastic quasi-Newton algorithm is obtained by selecting $\mathbf{W}_M(n)$ to be the pseudo-inverse

$$\mathbf{W}_M(n)=\left(\frac{1}{L}\mathbf{X}_{M,L}(n)\mathbf{X}_{M,L}^T(n)\right)^{\#}=L\mathbf{X}_{M,L}(n)(\mathbf{X}_{M,L}^T(n)\mathbf{X}_{M,L}(n))^{-1}(\mathbf{X}_{M,L}^T(n)\mathbf{X}_{M,L}(n))^{-1}\mathbf{X}_{M,L}^T(n) \quad (79)$$

In this case, it can be shown that $\mu(n)=1$. An over-relaxation form of the algorithm is obtained by setting $\mu(n)\neq 1$, $\mu(n)\in(0,2)$. This stochastic approximation method results in the so-called *affine projection algorithm*, or (APA), [91]-[105]. The APA, in a regularized and over-relaxation version, is listed in Table 5e. The APA may be also interpreted as a deterministic block projection algorithm that estimates the filter, which minimizes the error norm

$$\mathbf{c}_M(n)=\arg\min_{\mathbf{c},M}\|\mathbf{c}_M-\mathbf{c}_M(n-1)\|^2 \tag{80}$$

subject to the constraint imposed by the equation

$$\mathbf{y}_L(n)=\mathbf{X}^T_{M,L}(n)\mathbf{c}_M$$

Table 6 provides a summary of the discussed algorithms with respect to the adopted expectation approximation, the weighting matrix, and the search directions.

Fast-Adaptive Transversal Algorithms

The computational complexity of all the Gauss-Newton schemes considered so far, is $O(M^2)$. On the other hand, their fast convergence rate and immunity to the eigenvalue spread of the input signal's autocorrelation matrix, make these schemes particularly attractive [8], [10], [62]. A large research effort has been invested in overcoming their major drawback—the high computational complexity—and a number of $\mathcal{O}(M)$ Gauss-Newton algorithmic schemes have been developed [10], [55]-[59].

The Fast-RLS Algorithm

The RLS algorithm provides a method for the time updating of the inverse autocorrelation matrix, bypassing the direct matrix inversion imposed by (65). It was shown [55]-[59] that the Kalman gain vectors required by (69) can be updated by the so-called "fast" schemes. The key point in the development of fast-RLS algorithms is the proper utilization of the shift-invariance property of the data matrix. Let us consider the increased order data vector $\mathbf{x}_{M+1}(n)$. It is partitioned either in a lower or in an upper form, as

$$\mathbf{x}_{M+1}(n)=\begin{bmatrix}\mathbf{x}_M(n)\\ x(n-M)\end{bmatrix}=\begin{bmatrix}x(n)\\ \mathbf{x}_M(n-1)\end{bmatrix} \tag{81}$$

These partitions of the data vector are utilized to obtain appropriate partitions for the increased order autocorrelation matrix $\mathbf{R}_{M+1}(n)$. Based on the above partitions, the Kalman gain vector $\mathbf{w}_M(n)$, at time n, can be obtained from $\mathbf{w}_M(n-1)$, via the increased-order vector $\mathbf{w}_{M+1}(n)$. The overall updating mechanism is illustrated as follows:

$$\begin{array}{ccccc} & & \mathbf{a}_M(n-1) & & \mathbf{b}_M(n-1) \\ & & \downarrow & & \downarrow \\ \mathbf{w}_M(n-1) & \rightarrow & \mathbf{w}_{M+1}(n) & \rightarrow & \mathbf{w}_M(n) \\ & & \downarrow & & \downarrow \\ & & \mathbf{a}_M(n) & & \mathbf{b}_M(n) \end{array}$$

Auxiliary vectors $\mathbf{a}_M(n)$ and $\mathbf{b}_M(n)$ are the LS forward and backward predictors, and result from the general fil-

Table 7. The Stabilized Fast-RLS (SF-RLS) Algorithm with Modifications for Accelerated Tracking Ability.

Initialization

$\mathbf{c}_M(-1)=0, \mathbf{w}_M(-1)=0, \mathbf{a}_M(-1)=[1\ 0\ldots 0]^T, \mathbf{b}_M(-1)=[0\ldots 0\ 1]^T$

$\alpha_M(-1)=1, \alpha^f_M(-1)=\lambda^M\alpha^b_M(-1), \alpha^b_M(-1)=\delta>0$

$$e^f_M(n)=x(n)+\mathbf{a}^T_M(n-1)\mathbf{x}_M(n-1) \tag{1}$$

$$\epsilon^f_M(n)=e^f_M(n)/\alpha_M(n-1) \tag{2}$$

$$\mathbf{a}_M(n)=\mathbf{a}_M(n-1)-\mathbf{w}_M(n-1)\epsilon^f_M(n) \tag{3}$$

$$\alpha^f_M(n)=\lambda\alpha^f_M(n-1)+e^f_M(n)\epsilon^f_M(n) \tag{4}$$

$$k^{wf}_{M+1}(n)=\lambda^{-1}\alpha^f_M(n-1)e^f_M(n) \tag{5}$$

$$\mathbf{w}_{M+1}(n)=\begin{bmatrix}0\\ \mathbf{w}_M(n)\end{bmatrix}+\begin{bmatrix}1\\ \mathbf{a}_M(n-1)\end{bmatrix}k^{wf}_{M+1}(n-1) \tag{6}$$

$$\mathbf{w}_{M+1}(n)=\begin{bmatrix}\delta_M(n)\\ \delta^w_{M+1}(n)\end{bmatrix} \tag{7}$$

$$e^b_M(n)=\lambda\alpha^b_M(n-1)\delta^w_{M+1}(n) \tag{8}$$

$$\tilde{e}^b_M(n)=x(N-m)+\mathbf{x}^T_M(n)\mathbf{b}_M(n-1) \tag{9}$$

$$\Delta^b(n)=\tilde{e}^b_M(n)-e^b_M(n) \tag{10}$$

$$\hat{e}^{b,i}_M(n)=\tilde{e}^{b,i}_M(n)+\sigma^i\Delta^b(n),\ i=1,2,3 \tag{11}$$

$$\mathbf{w}_M(n)=\delta_M(n)-\mathbf{b}_M(n-1)\delta^w_{M+1}(n) \tag{12}$$

$$\alpha_{M+1}(n)=\alpha_M(n-1)-k^{wb}_{M+1}(n)e^f_M(n) \tag{13}$$

$$\alpha_M(n)=\alpha_{M+1}(n)+\delta^w_{M+1}(n)\hat{e}^{b,1}_M(n) \tag{14}$$

$$\tilde{\alpha}_M(n+1)=1+\mathbf{w}^T_M(n)\mathbf{x}_M(n) \tag{15}$$

$$\Delta^*(n)=\tilde{\alpha}_M(n)-\alpha^b_M(n) \tag{16}$$

$$\hat{\alpha}_M(n)=\tilde{\alpha}_M(n)+\sigma\Delta^*(n) \tag{17}$$

$$\epsilon^{b,i}_M(n)=\hat{e}^{b,i}_M(n)/\hat{\alpha}_M(n),\ i=2,3 \tag{18}$$

$$\mathbf{b}_M(n)=\mathbf{b}_M(n-1)-\mathbf{w}_M(n)\epsilon^{b,2}_m(n) \tag{19}$$

$$\alpha^b_M(n)=\lambda\alpha^b_M(n-1)+\hat{e}^{b,3}_M(n)\epsilon^{b,3}_M(n) \tag{20}$$

$$e_M(n)=y(n)-\mathbf{x}^T_M(n)\mathbf{c}_M(n-1) \tag{21}$$

$$\epsilon_M(n)=e_M(n)/\hat{\alpha}_M(n) \tag{22}$$

$$\mu(n)=\alpha_M(n)/(1-\rho\hat{\alpha}_M(n)) \tag{23}$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{w}_M(n)\epsilon_M(n) \tag{24}$$

tering scheme of (6), by setting $y(n)=x(n+1)$ and $y(n)=x(n-M)$, respectively.

The computational complexity of the fast RLS algorithm is $7M$ operations per processed sample. Thus, while the algorithm retains the nice properties of the RLS algorithm, the cost is comparable to that of the LMS. Despite the fact that the fast RLS is theoretically equivalent to the RLS scheme, it exhibits poor numerical error performance, which becomes noticeable even when the algorithm is carefully implemented in double precision arithmetic [63]-[69]. Severe divergence problems are caused by the intrinsic instability of the fast RLS algorithm. The magnification of a single error committed at a time instant, becomes so serious that the algorithm crashes. An early attempt to cope with this problem was concentrated on the periodic reinitialization of the algorithm, when certain variables were taking critical values very close to the theoretical upper or lower bounds. After many years of extensive study and experimentation on the instability problem, a leap forward was achieved by the method of [68], which suggested using a small amount of computational redundancy combined with a suitable feedback mechanism, to control the numerical errors. Two specific variables are used for this purpose, [68]- [70], the a priori backward error $e^b_M(n)$ and the Kalman gain power $\alpha_M(n)$. These are estimated in two different ways, from (8), (13), and (14) of Table 7, and also by their respective definitions:

$$\tilde{e}^b_M(n)=x(N-M)+\mathbf{x}^t_M(n)\mathbf{b}_M(n-1)$$

and

$$\tilde{\alpha}_M(n)=1+\mathbf{x}^t_M(n)\mathbf{w}_M(n). \tag{82}$$

We form the differences

$$\Delta^b_M(n)=\tilde{e}^b_M(n)-e^b_M(n)$$

and

$$\Delta^\alpha_M(n)=\tilde{\alpha}_M(n)-\alpha_M(n). \tag{83}$$

In infinite precision, these differences should be zero; however, in practice, $\Delta^b_M(n)$ grows exponentially while $\Delta^\alpha_M(n)$ grows linearly. These errors are utilized as a feedback mechanism to control the error propagation and provide a new set of variables to be used by the algorithm as

$$\hat{e}^b_M(n)=\tilde{e}^b_M(n)+\sigma^b\Delta^b_M(n)$$

and

$$\hat{\alpha}_M(n)=\tilde{\alpha}_M(n)+\sigma^\alpha\Delta^\alpha_M(n). \tag{84}$$

Tuning variables σ^b and σ^α are carefully selected in order that the overall algorithm remains stable [69]. The complexity of the stabilized fast-RLS algorithm of Table 7 is $O(M)$.

An additional constraint of the stabilized fast-RLS algorithms is the nominal range within which the forgetting factor λ should lie for the algorithm to remain stable, i.e., $\lambda\in(1-1/3p,1)$ [10]. This has an undesired effect in the performance of the fast-RLS schemes, since the convergence rate and tracking ability of the algorithm are affected. As a remedy to this problem, a modified algorithm has been proposed and successfully used in [70], where an extra trimming parameter was introduced as

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{w}_M(n)\epsilon(n) \tag{85}$$

Table 8. The Fast Quasi-Newton (FQN) Algorithm.

Initialization

$\mathbf{t}_M(-1)=\mathbf{I}_M\delta>0, \mathbf{c}_M(-1)=0, \mathbf{w}_M(-1)=0$

Each time instant update the autocorrelation vector

$$\mathbf{t}_M(n)=\lambda\mathbf{t}_M(n-1)+\mathbf{x}_M(n)x(n) \tag{1}$$

If mod(n,N))==0,

a) run the Levinson's algorithm

$$\mathbf{a}_1(n)=-t_1(n)/t_0(n),$$

$$\mathbf{w}_1(n)=-x(n)/t_0(n),$$

$$\alpha_1(n)=t_0(n)+\mathbf{a}_1(n)t_1(n) \tag{2}$$

For $i=1$ to $M-1$

$$\beta^w_{i+1}(n)=x(n-i)+\mathbf{x}^T_i(n)\mathbf{J}_i\mathbf{a}_i(n) \tag{3}$$

$$k^w_{i+1}(n)=\beta^w_{i+1}(n)/\alpha_i(n) \tag{4}$$

$$\mathbf{w}_{i+1}(n)=\begin{bmatrix}\mathbf{w}_i(n)\\0\end{bmatrix}+\begin{bmatrix}\mathbf{J}_i\mathbf{a}_i(n)\\1\end{bmatrix}k^w_{i+1}(n) \tag{5}$$

$$\beta_{i+1}(n)=t_{i+1}+\mathbf{t}^T_i(n)\mathbf{J}_i\mathbf{a}_i(n) \tag{6}$$

$$k_{i+1}(n)=-\beta_{i+1}(n)/\alpha_i(n) \tag{7}$$

$$\mathbf{a}_{i+1}(n)=\begin{bmatrix}\mathbf{a}_i(n)\\0\end{bmatrix}+\begin{bmatrix}\mathbf{J}_i\mathbf{a}_i(n)\\1\end{bmatrix}k_{i+1}(n) \tag{8}$$

$$\alpha_{i+1}(n)=\alpha_i(n)-\beta_{i+1}(n)k_{i+1}(n) \tag{9}$$

END i

b) Update the filter on the time interval $[n-N+1,N]$

For $m=0$ to $N-1$

$$\beta^{wf}_{M+1}(n+m+1)=x(n+m+1)+\mathbf{x}^T_M(n+m)\mathbf{a}_M(n) \tag{10}$$

$$k^{wf}_{M+1}(n+m+1)=\beta^{wf}_{M+1}(n+m+1)/\alpha_M(n) \tag{11}$$

$$\mathbf{w}_{M+1}(n+m+1)=\begin{bmatrix}0\\\mathbf{w}_M(n+m)\end{bmatrix}+\begin{bmatrix}1\\\mathbf{a}_M(n)\end{bmatrix}k^{wf}_{M+1}(n+m+1) \tag{12}$$

$$k^{wb}_{M+1}(n+m+1)=w^{M+1}_{M+1}(n+m+1)$$

$$\begin{bmatrix}\mathbf{w}_M(n+m)\\0\end{bmatrix}=\mathbf{w}_{M+1}(n+m+1)-\begin{bmatrix}\mathbf{J}_M\mathbf{a}_M(n)\\1\end{bmatrix}k^{wb}_{M+1}(n+m+1) \tag{13}$$

$$e_M(n+m+1)=y(n+m+1)-\mathbf{x}^T_M(n+m+1)\mathbf{c}_M(n+m) \tag{14}$$

$$\mathbf{c}_M(n+m+1)=\mathbf{c}_M(n+m)+\mathbf{w}_M(n+m+1)e_M(n+m+1) \tag{15}$$

END m

$\mu(n)$ is optimized in such a way that the a posteriori filtering error is minimized, i.e.,

$$\mu(n)=\arg\min_{\mu}\epsilon^2(n)$$
$$=\arg\min_{\mu}\left(1-\mu\frac{\mathbf{w}_M^T(n)\mathbf{x}_M(n)}{\alpha_M(n)}\right)^2\epsilon^2(n). \quad (86)$$

Minimization of the above cost leads to $\epsilon(n)=0$ and $\mu(n)=\alpha_M(n)/(1-\alpha_M(n))$. In practice,

$$\mu(n)=\frac{\alpha_M(n)}{1-\rho\alpha_M(n)}, \quad (87)$$

where ρ is a trimming variable. The cost is that it results in a higher mean-square-estimation error.

Multichannel extensions of the stabilized fast-adaptive transversal filter are also derived in [76], [77]. Multichannel QR decomposition algorithms are developed in [73].

The Fast-Quasi-Newton Algorithm

The fast-quasi-Newton algorithm offers a fast, although suboptimal, implementation of the RLS method, [82]. The key point lies in the different rates of adaptation used for the filter coefficients and the direction matrix. Thus, although $\mathbf{c}_M(n)$ is updated every single time instant, $\mathbf{T}_M^{-1}(n)$ in (74), is estimated periodically, and afterwards kept constant for a period of N samples. The choice of N depends on the stationarity characteristics of the input signal, and in general, has to be greater than or equal to M, to keep complexity comparable to that of FRLS. Thus, each time $mod(n,N)=0$, an efficient Levinson algorithm is employed that solves for $\mathbf{w}_M(n)$.

Suppose that the solution of the linear system $\mathbf{T}_M(n)\mathbf{w}_M(n)=\mathbf{x}_M(n)$ has been computed. This task can be accomplished by the order-recursive Levinson algorithm or its Schur-type counterpart, [10], at a cost of $O(M^2)$ MADS. Then, using the order partitions of the data vector $\mathbf{x}_M(n+m+1)$, $m=0,1,\ldots N-1$, (81), and the corresponding partitions of the increased-order Toeplitz matrix $\mathbf{T}_{M+1}(n)$, a sequence of step-up/step-down recursions can be established, leading to the successive computation of the variables $\mathbf{w}_M(n+m)$ and $\mathbf{w}_M(n+m+1)$, $m=0,1,\ldots N-1$. This gives an efficient mechanism for updating the Kalman gain vectors over the time range $[n,n+N]$. The algorithm is tabulated in Table 8. The computational complexity of the algorithm is thus, $3M+1.5M/N$ operations per processed sample.

The numerical behavior of the fast-quasi-Newton algorithm is expected to be better than that of the fast-RLS method. The main difference between the adaptation procedure of these algorithms is the way that the computations for the Kalman gain are propagated through time. In the fast-RLS case, the Kalman gain vector is adapted at every time instant by an infinite-memory time-varying adaptive structure, therefore, numerical errors are allowed to propagate. In the fast-quasi-Newton case, the Kalman gain is updated through a fixed-memory adaptive scheme, thus numerical errors are not allowed to propagate more than M time instants. Moreover, Levinson's algorithm, which is periodically used every M samples for the initialization of the recursive updating mechanism, is known to be (forward) stable [78], [79].

Table 9. The Modified Fast-Newton Algorithm with Accelerated Tracking Ability.

Initialization

$\mathbf{c}_M(-1)=0, \alpha_{M|P}^b(-1)=1$

$\mathbf{a}_P(-1)=\lambda^M\delta, \alpha_P^f(-1-M+P)=\lambda^M\delta$

$\mathbf{a}_P(-1)=\lambda^{M-P}\delta, \alpha_P^f(-1-M+P)=\delta>0$

Available at time n from the stabilized fast RLS algorithm

$$\mathbf{S}_{P+1}(n)\equiv\begin{bmatrix}1\\ \mathbf{a}_P(n)\end{bmatrix}k_{P+1}^{wf}(n),e_P^f(n),k_{P+1}^{fv}(n)$$

Available at time $n-M+P$ from the stabilized fast RLS algorithm

$$\mathbf{U}_{P+1}(n-M+P)\equiv$$
$$\begin{bmatrix}\mathbf{b}_P(n-M+P)\\ 1\end{bmatrix}k_{P+1}^{wb}(n-M+P),e_P^b(n-M+P),k_{P+1}^{wb}(n-M+P)$$

Main part

$$\mathbf{w}_{M+1|P}(n)=\begin{bmatrix}0\\ \mathbf{w}_{M|P}(n-1)\end{bmatrix}+\begin{bmatrix}\mathbf{S}_{P+1}(n)\\ 0_{M-P}\end{bmatrix} \quad (2)$$

$$\begin{bmatrix}\mathbf{w}_{M|P}(n)\\ 0\end{bmatrix}=\mathbf{w}_{M+1|P}(n)-\begin{bmatrix}0_{M-P}\\ \mathbf{U}_{P+1}(n-M+P)\end{bmatrix} \quad (4)$$

$$\alpha_{M|P}(n)=\alpha_{M|P}(n-1)-k_{P+1}^{wf}(n)e_P^f(n)+k_{P+1}^{wb}(n-M+P)e_P^b(n-M+P) \quad (5)$$

$$e_M(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \quad (6)$$

$$\epsilon_M(n)=\frac{e_M(n)}{\alpha_{M|P}(n)} \quad (7)$$

$$\mu(n)=\frac{\alpha_{M|P}(n)}{1-\rho\alpha_{M|P}(n)} \quad (8)$$

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-1)+\mu(n)\mathbf{w}_{M|P}(n)\epsilon_M(n) \quad (9)$$

The Fast-Newton Algorithm

The fast-Newton algorithm has been derived optimally on the assumption that the input signal $x(n)$ can be sufficiently represented by an AR(P) process of order $P\le M$. This is an assumption which is valid, for example, when speech signals are involved. If this is not true, optimality is no more valid, and the fast-Newton scheme provides a family of algorithms, depending on the choice of P, varying from the normalized LMS, i.e. $P=0$, to the fast RLS, i.e. $P=M$. Based on this assumption, (76) can be utilized [85],[86], [87]. This has as an implication that the

Kalman gain of order M is related to the delayed backward predictors and forward predictors of order P. These are, in turn, updated via fast-RLS versions. The overall updating mechanism is illustrated as follows:

$$\begin{array}{ccccc} & \mathbf{a}_P(n) & & \mathbf{b}_P(n-M+P) & \\ & \downarrow & & \downarrow & \\ \mathbf{w}_{M|P}(n) & \rightarrow & \mathbf{w}_{M+1|P}(n+1) & \rightarrow & \mathbf{w}_{M|P}(n+1). \end{array}$$

In other words, the prediction part of the FRLS scheme, which is the computationally thirsty part of the algorithm, is performed with predictors of low-order P. The algorithm is tabulated in Table 8.

For the update of the prediction part, any of the adaptive schemes described previously can be utilized, resulting in different complexity figures, varying from $2P^2$ to $7P$ when the RLS and the stabilized FRLS schemes are utilized, respectively. The overall complexity reduces to $2M+9P$ operations when the stabilized FRLS algorithm is engaged.

The Fast-Affine Projection Algorithm

The first step toward the development of a fast-affine projection algorithm is the establishment of a shifting property for the calculation of the a priori filtering error vector $\mathbf{e}_L(n)$, [93]-[98]. To this end, consider the partitioning of the data matrix

$$\begin{aligned} \mathbf{X}_{M,L}(n) &= [\mathbf{X}_{M,L-1}(n) \;\; \mathbf{x}_M(n-L+1)] \\ &= [\mathbf{x}_M(n) \;\; \mathbf{X}_{M,L-1}(n-1)] \end{aligned} \tag{88}$$

If we plug the above partitions into (45), which defines the a priori filtering error over a data block of length L, we get

$$\mathbf{e}_L(n) = \begin{bmatrix} \mathbf{e}_{L-1}(n) \\ \hat{e}(n-L+1) \end{bmatrix}, \tag{89}$$

where

$$\hat{e}(n-L+1) = y(n-L+1) - \mathbf{x}_M^T(n-L+1)\mathbf{c}_M(n-1). \tag{90}$$

It can be shown [93]-[98] that the following shifting property can be established, where equality holds when δ used in (2) of Table 5c, is set equal to zero

$$\mathbf{e}_L(n) = \begin{bmatrix} \mathbf{e}_{L-1}(n) \\ \hat{e}(n-L+1) \end{bmatrix} \approx \begin{bmatrix} e(n) \\ (1-\mu)\mathbf{e}_{L-1}(n) \end{bmatrix}. \tag{91}$$

Following a derivation similar to that of "The Fast-RLS Algorithm," a step-up/step-down recursion is derived for the efficient adaptation of the involved variable $\mathbf{w}_L(n)$, by taking into account the shift-invariance property that vectors $\mathbf{x}_L(n)$ and $\mathbf{e}_L(n)$ possess. This time however, because $\mathbf{R}_L(n)$ is a sliding-window covariance matrix, a sliding-window fast-RLS algorithm is required [60], [61]. Thus, (4) and (5) of Table 10, result. The lower-order forward and backward predictors $\mathbf{a}_L(n)$ and $\mathbf{b}_L(n)$ that appear in these recursions can be obtained by a sliding-window fast-RLS [60]. A set of auxiliary filters, i.e., $\mathbf{h}_M(n)$ and $\mathbf{q}_L(n)$ (see Table 10), are utilized for the development of the remaining recursions of the fast-affine algorithm of Table 10. The complexity of the fast APA is $2M+20L$. Notice, however, that the fast APA estimates the filtering error, and not the unknown filter coefficients.

Table 10. The Fast APA Algorithm.

Initialization
$\mathbf{h}_M(-1)=0$

Available at time n, from a sliding-window fast-RLS algorithm
$\mathbf{a}_L(n), \mathbf{b}_L(n), \alpha_L^f(n), \alpha_L^b(n)$

$$e^b(n) = y(n) - \mathbf{x}_M^T(n)\mathbf{h}_M(n-1) \tag{1}$$

$$\mathbf{r}_{L-1}(n) = \mathbf{r}_{L-1}(n-1) + \mathbf{x}_{L-1}(n-1)x(n) - \mathbf{x}_{L-1}x(n-L-1) \tag{2}$$

$$e(n) = e^b(n) - \mu\mathbf{r}_{L-1}^T(n)\mathbf{q}_L^{L-1}(n-1) \tag{3}$$

$$\mathbf{e}_L(n) = \begin{bmatrix} e(n) \\ (1-\mu)\mathbf{e}_{L-1}(n) \end{bmatrix} \tag{4}$$

$$\beta_{L+1}^f(n) = e(n) + (1-\mu)\mathbf{e}_L^T(n-1)\mathbf{a}_L(n) \tag{5}$$

$$k_{L+1}^{wf}(n) = \frac{\beta_{L+1}^f(n)}{\alpha_L^f(n)+\delta} \tag{6}$$

$$\mathbf{w}_{L+1}(n) = \begin{bmatrix} 0 \\ (1-\mu)\mathbf{w}_L(n-1) \end{bmatrix} + \begin{bmatrix} 1 \\ \mathbf{a}_L(n-1) \end{bmatrix} k_{L+1}^{wf}(n) \tag{7}$$

$$\beta_{L+1}^b(n) = e(n-L+1) + \mathbf{e}_L^T(n)\mathbf{b}_L(n) \tag{8}$$

$$k_{L+1}^{wb}(n) = \frac{\beta_{L+1}^b(n)}{\alpha_L^b(n)+\delta} \tag{9}$$

$$\begin{bmatrix} \mathbf{w}_L(n) \\ 0 \end{bmatrix} = \mathbf{w}_{L+1}(n) - \begin{bmatrix} \mathbf{b}_L(n) \\ 1 \end{bmatrix} k_{L+1}^{wb}(n) \tag{10}$$

$$\mathbf{q}_L(n) = \mathbf{w}_{M,L}(n) + \begin{pmatrix} 0 \\ \mathbf{q}_{L-1}(n) \end{pmatrix} \tag{11}$$

$$\mathbf{q}_L(n) = \begin{bmatrix} \mathbf{q}_{L-1}(n) \\ q_L(n) \end{bmatrix} \tag{12}$$

$$\mathbf{h}_M(n) = \mathbf{h}_M(n-1) + \mu\mathbf{x}_M(n-L+1)q_L(n) \tag{13}$$

Definitions:

$$\mathbf{h}_M(n-1) = \mathbf{c}_M^0 + \mu\sum_{j=1}^{L}\sum_{k=0}^{n-L}\mathbf{x}_M(k)w_L^j(k+j-1)$$

$$\mathbf{q}_L = [q_1(n)q_2(n)\ldots q_L(n)]^T,\; q_i(n) = \sum_{j=1}^{i} w_j(n-i+1),\; i=1,2,\ldots L$$

$$\mathbf{c}_M(n) = \mathbf{h}_M(n-1) + \mu\mathbf{X}_{M,L}(n)\mathbf{q}_L(n)$$

Block-Adaptive Filtering

A large number of block-adaptive-filtering algorithms have been developed during the past two decades. All this work has sprung mainly from a number of seminal papers, which appeared in the late 1970s and early 1980s [28],[29],[37],[38],[39], [132], [133]. Ever since, intensive research has been done in several directions. Depending on the domain of implementation, the block-adaptive techniques can be classified as frequency (or transform) domain and time domain, respectively. The transform domain techniques have been summarized in an excellent survey paper by Shynk [49] in 1992. A list of relevant papers, that have been reported since then, is given in "Accelerating the Adaptive Gradient Methods," and also in the companion article in this issue of *IEEE Signal Processing Magazine* [36].

In this section, we are mainly concerned with efficient block implementations in the time domain. Although FFT may still be used as a tool for fast computation of the involved convolutions, the filtering operations and the filter updating is done in the time domain. In the first part of this section, the so-called block-exact techniques are presented. This is a relatively new trend in block-adaptive filtering, which deserves attention, since it offers significant computational savings without any loss in performance. In the second part, an effort is made to present briefly, but systematically, the so-called approximate block-adaptive techniques, where the term "approximate" has been used to emphasize the inherent difference with the exact techniques.

Block-Exact Techniques

The distinctive trait of the block-exact techniques is that the resulting block-adaptive algorithms have an exact equivalence with their step-by-step counterparts. In an exact-block algorithm, the filtering errors are computed at every time instant, while the filter taps are updated every N time steps, where N is the block length. However, all these quantities are equal to the respective quantities computed by the corresponding step-by-step adaptive algorithm. Thus, the same estimates are obtained at a substantially reduced complexity. Moreover, such schemes incorporate block sizes much smaller than the filter order, therefore, the delay introduced is small, for most practical applications.

Fast-Exact-Least-Mean-Square Algorithm

The first block-exact algorithm that appeared in the literature was the one in [106]. This scheme, called fast-exact LMS (FELMS), is mathematically equivalent, in the sense described above, to the classical LMS algorithm. Given a data block length N, the goal is to compute $\mathbf{c}_M(n)$, based on $\mathbf{c}_M(n-N)$, i.e., the estimate corresponding to the beginning of the current data block, and the input/output samples at all time instants $n,n-1,\ldots n-N$, within the current data block. In order to demonstrate the

The distinctive trait of the block-exact techniques is that the resulting block-adaptive algorithms have an exact equivalence with their step-by-step counterparts.

underlying philosophy of the method, let us consider the filtering operation (33) for the LMS for all time steps. Within the block $n-N+1,\ldots,n$, we get

$$e(n-N+1)=y(n-N+1)-\mathbf{x}_M^T(n-N+1)\mathbf{c}_M(n-N) \tag{92}$$

$$\vdots \quad \vdots \tag{93}$$

$$e(n)=y(n)-\mathbf{x}_M^T(n)\mathbf{c}_M(n-1) \tag{94}$$

In the sequel, all $\mathbf{c}_M(n-k)$ for $k=1,\ldots,n-N+1$ are expressed in terms of $\mathbf{c}_M(n-N)$, using the LMS update equation. For instance, error $e(n-N+3)$ is easily checked to be given by

$$\begin{aligned} e(n-N+3)=&\,y(n-N+3)-\mathbf{x}_M^T(n-N+3)\mathbf{c}_M(n-N)\\ &-\mu e(n-N+1)\mathbf{x}_M^T(n-N+3)\mathbf{x}_M(n-N+1)\\ &-\mu e(n-N+2)\mathbf{x}_M^T(n-N+3)\mathbf{x}_M(n-N+2) \end{aligned} \tag{95}$$

Following a similar procedure to all filtering errors and expressing the resulting relations in matrix form we obtain

$$\tilde{\mathbf{e}}_N(n)=\tilde{\mathbf{y}}_N(n)-\tilde{\mathbf{X}}_{M,N}^T(n)\mathbf{c}_M(n-N)-\mathbf{S}_{N,N}(n)\tilde{\mathbf{e}}_N(n) \tag{96}$$

where

$$\tilde{\mathbf{e}}_N(n)=[e(n-N+1)\ldots e(n)]^T,$$

$$\tilde{\mathbf{y}}_N(n)=[y(n-N+1)\ldots y(n)]^T,$$

and

$$\tilde{\mathbf{X}}_{M,N}(n)=[\mathbf{x}_M(n-N+1)\ldots\mathbf{x}_M(n)].$$

Matrix $\mathbf{S}_{N,N}(n)$ is

$$S_{N,N}(n)=\begin{bmatrix} 0 & 0 & \ldots & 0 & 0\\ s_1(n-N+2) & 0 & \ldots & 0 & 0\\ s_2(n-N+3) & s_1(n-N+3) & \ldots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ s_{N-1}(n) & s_{N-2}(n) & \ldots & s_1(n) & 0 \end{bmatrix} \tag{97}$$

where

$$s_i(n)=\mu\mathbf{x}_M^T(n)\mathbf{x}_M(n-i) \tag{98}$$

Table 11a. The Block-Exact FNTF Algorithm.

Let $[n-N+1,n]$ be the current data block and assume that input-desired output data are available until time instant $n+P$. Then:

Part 1. Sample-by-sample computations.

1.1 Using the prediction part of the stabilized FAEST, compute

$\forall i\in[n-N+P+1,n+P]$

the following quantities associated with the forward and backward predictors:

$$\mathbf{s}_{P+1}(i)=\frac{e_P^f(i)}{\lambda\alpha_P^f(i-1)}\begin{pmatrix}1\\ (\mathbf{a}_P(i-1))\end{pmatrix},$$

$$\mathbf{u}_{P+1}(i)=\frac{e_P^b(i)}{\lambda\alpha_P^b(i-1)}\begin{pmatrix}\mathbf{b}_P(i-1)\\ 1\end{pmatrix}$$

1.2 Compute the gain error power $\alpha_P(i)$ $\forall i\in[n-N+1,n]$ using recursion

$\alpha_P(i)=\alpha_P(i-1)-s_{P+1}^1(i)e_P^f(i)+u_{P+1}^{P+1}(i^d)e_P^b(i^d)$

where

$i^d=i-M+P$

and x^m is the m-th element of vector $\mathbf{x}$.

1.3 Using recursion (27) compute the gain vector $\mathbf{w}_M(i)$ only for $i=n-N+1$.

Part 2. Block computation of the prediction part errors.

2.1 Using fast-FIR filtering techniques compute the following seed errors (initialization phase)

$e_P^f(n-N+1;1),e_P^f(n-N+2;2),\ldots,e_P^f(n+P;N+P)$

$e_P^b(n-N+1;1),e_P^b(n-N+2;2),\ldots,e_P^b(n+P;N+P)$

$e_P^w(n-N+1;1),e_P^w(n-N+2;2),\ldots,e_P^w(n+P;N+P)$

2.2 Recursive scheme

For $i=n-N+2:n+P$

For $k=i-n+N:-1:2$

$e_P^w(i+1;k)=e_P^w(i;k)+s_{P+1}^1(i-k+1)e_P^f(i;k)-u_{P+1}^{P+1}(i-k+1)e_P^b(i;k)$

$e_P^f(i;k-1)=e_P^f(i;k)-\epsilon_P^f(i-k+1)e_P^w(i;k)$

$e_P^b(i;k-1)=e_P^b(i;k)-\epsilon_P^b(i-k+1)e_P^w(i+1;k)$

Part 3. Block computation of the filtering part errors

3.1 Using fast-FIR filtering techniques, compute the following seed errors (initialization phase)

$e_M(n-N+1;1),e_M(n-N+2;2),\ldots,e_M(n;N)$

$e_M^w(n-N+1;1),e_M^w(n-N+2;2),\ldots,e_M^w(n;N)$

3.2 Recursive scheme

For $i=n-N+2:n$

For $k=i-n+N:-1:2$

$e_M^w(i+1;k)=e_M^w(i;k)+s_{P+1}^1(i-k+1)e_P^f(i;k)-u_{P+1}^{P+1}(i^d-k+1)e_P^b(i^d;k)$

$e_M(i;k-1)=e_M(i;k)-\epsilon_M(i-k+1)e_M^w(i+1;k)$

3.3

$$\epsilon_N(N)=\begin{pmatrix}\alpha_M^{-1}(n-N+1)e_M(n-N+1)\\ \vdots\\ \alpha_M^{-1}(n)e_M(n)\end{pmatrix}$$

Part 4. Filter block updating.

4.1 Formation of the involved matrices

For $j=1:N-1$

$$\widetilde{\mathbf{s}}_M(n-N+j+1)=\begin{pmatrix}0\\ \widetilde{\mathbf{s}}_M^{M-1}(n-N+j)\end{pmatrix}+\begin{pmatrix}\mathbf{s}_{P+1}(n-N+j+1)\\ \mathbf{0}_{M-P-1}\end{pmatrix}$$

$$\widetilde{\mathbf{u}}_M(n-N+j+1)=\begin{pmatrix}0\\ \widetilde{\mathbf{u}}_M^{M-1}(n-N+j)\end{pmatrix}+\begin{pmatrix}\mathbf{0}_{M-P}\\ \mathbf{u}_{P+1}^P(n^d-N+j+1)\end{pmatrix}$$

End

Form matrices

$\mathbf{W}_{M,N}(n)$, $\mathbf{S}_{M,N}(n)$,

and $\mathbf{U}_{M,N}(n)$ using their respective definitions.

Then, compute $w_M(n)$ by adding the last columns of these three matrices.

4.2 Computation of the block updating term.

4.2.1. Using fast-FIR filtering techniques to perform the multiplication

$\mathbf{W}_{M,N}(n)\epsilon_N(n)$.

4.2.2 Perform directly the multiplication

$\mathbf{S}_{M,N}(n)\epsilon_N(n)$.

4.2.3 Perform directly the multiplication

$\mathbf{U}_{M\times N}(n)\epsilon_N(n)$.

4.3 Filter updating

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)-\mathbf{W}_{M,N}(n)\epsilon_N(n)+\mathbf{S}_{M,N}(n)\epsilon_N(n)+\mathbf{U}_{M,N}(n)\epsilon_N(n).$$

Solving (96) with respect to the filtering error vector, the following relation is obtained

$$\tilde{\mathbf{e}}_N(n)=\mathbf{G}_{N,N}(n)[\tilde{\mathbf{y}}_N(n)-\tilde{\mathbf{X}}^T_{M,N}(n)\mathbf{c}_M(n-N)], \quad (99)$$

where $\mathbf{G}_{N,N}(n)=[\mathbf{S}_{N,N}(n)+I]^{-1}$.

Now, a block update recursion for the filter $\mathbf{c}_M$ can be easily derived by starting from the step-by-step update of (2) in Table 1a. Writing this equation for all time steps within the current block and combining properly the resulting equations we readily get

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)+\mu\tilde{\mathbf{X}}_{M,N}(n)\tilde{\mathbf{e}}_N(n),$$
$$n=0,N,2N,\ldots \quad (100)$$

(99) and (100) constitute the FELMS pair of equations. The computational savings result from the efficient computation of the above recursions. To this end, observe that the matrix-by-vector product in (99) constitutes a fixed-FIR filtering problem. By properly rearranging the involved entries, this product can be written as

$$\tilde{\mathbf{X}}^T_{M,N}(n)\mathbf{c}_M(n-N)=$$
$$\begin{bmatrix} a_{N-1} & a_N & \cdots & a_{2N-3} & a_{2N-2} \\ a_{N-2} & a_{N-1} & \cdots & \cdots & a_{2N-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_1 & \cdots & \cdots & \cdots & a_N \\ a_0 & a_1 & \cdots & a_{N-2} & a_{N-1} \end{bmatrix} \begin{bmatrix} \mathbf{c}^0 \\ \mathbf{c}^1 \\ \vdots \\ \mathbf{c}^{N-2} \\ \mathbf{c}^{N-1} \end{bmatrix} (n-N) \quad (101)$$

where

$$a_j=[x(n-j)x(n-N-j)\ldots x(n-Ni-j)\ldots x(n-M+N-j)]^T,$$
$$j=0,1,\ldots,2N-2 \quad i=0,1,\ldots,(M/N)-1,$$

and

$$\mathbf{c}^k=[c_k, c_{k+N},\ldots,c_{k+Ni},\ldots,c_{k+M-N}]^T$$
$$k=0,1,\ldots,N-1.$$

Due to the above polyphase representation of the involved quantities, the matrix-by-vector product can now be efficiently computed. As it can be seen, the involved matrix has a block-Toeplitz form, and the whole product can be considered an FIR filtering problem with a block filter of length N. The fast-FIR filtering technique of [134] can now be applied in a straightforward manner.

The matrix-by-vector product in (100) can also be treated in a similar way, since the same matrix $\tilde{\mathbf{X}}_{M,N}(n)$ is involved. Matrix $\mathbf{G}_{N,N}(n)$, due to the triangular structure of $\mathbf{S}_{N,N}(n)$, can be written in the form

$$\mathbf{G}_{N,N}(n)=\mathbf{G}_{N-1}(n)\mathbf{G}_{N-2}(n)\cdots\mathbf{G}_1(n)$$

where $\mathbf{G}_i(n)$ is an $N\times N$ triangular matrix defined as

The involved matrix has a block-Toeplitz form, and the whole product can be considered an FIR filtering problem with a block filter of length *N*.

$$\mathbf{G}_i(n)=\mathbf{Z}_i(n)+I,$$

with $\mathbf{Z}_i(n)$ being an $N\times N$ matrix having vector

$$[-s_i(n-N+i+1)-s_{i-1}(n-N+i+1)\ldots-s_1(n-N+i+1)\,0\ldots0]$$

in its $i+1$ line, and zeros elsewhere. The scalar quantities $s_j(k)$ can be computed via a set of easily derived recursive relations [106].

As it can be readily seen by inspecting (99) and (100), the whole formulation of FELMS would be exactly the same as that of conventional BLMS if matrix $\mathbf{S}_{N,N}(n)$ were equal to the identity matrix. In fact, as the authors in [106] suggest, different approximations of this matrix lead to a whole family of algorithms, with BLMS being one of its members. The computational complexity of the basic formulation of FELMS, as given by (99) and (100), is significantly reduced with respect to that of LMS. The number of operations per output, assuming $N=2^v$ and $M=Q\times N$, is given by the following formulae

$$2\left(\frac{3}{2}\right)^v Q+\left(\frac{3\cdot 2^v-5}{2}\right)+\left(\frac{1}{2}\right)^v \quad MUPI \quad (102)$$

$$2\left(2\left(\frac{3}{2}\right)^v-1\right)Q+4\left(\frac{3}{2}\right)^v+2(2^v-3) \quad ADPI, \quad (103)$$

where $MUPI$ and $AUPI$ stand for multiplications per time instant (step) and additions per time instant, respectively.

Block-Exact-Fast-Newton Algorithm

As was the case with the LMS, a block-exact version of the fast-newton-transversal-filtering algorithm has also been derived in [107] and [108]. The respective algorithm, called block-exact FNTF (BEFNTF) is summarized in Table 11a. The derivation of BEFNTF is quite lengthy, and here we will focus on the main points that reveal the basic philosophy behind the method. Recall from "Fast-Adaptive Transversal Algorithms" the filter update formula written in terms of the dual-Kalman gain and the a posteriori error

$$\mathbf{c}_M(n)=\mathbf{c}_{M|P}(n-1)+\mathbf{w}_{M|P}(n)\epsilon_M(n). \quad (104)$$

Note that the autocorrelation matrix $\mathbf{R}_M(n-1)$, involved in the Kalman gain vector is computed in the FNTF sense. Writing recursion (104) in N successive time steps

The efficient computation of the block-updating term is the most important task in the algorithm because it refers to the most computationally thirsty part of the FNTF algorithm.

and combining the resulting equations we can express $\mathbf{c}_M(n)$ in terms of $\mathbf{c}_M(n-N)$ as follows

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)+\mathbf{G}_{M,N}(n)\boldsymbol{\epsilon}_N(n),$$
$$n=0,N,2N,\ldots \tag{105}$$

where

$$\mathbf{G}_{M,N}(n)=[\mathbf{w}_{M|P}(n-N+1),\ldots,\mathbf{w}_{M|P}(n)] \tag{106}$$

$$\boldsymbol{\epsilon}_N(n)=[\epsilon_M(n-N+1),\ldots,\epsilon_M(n)]^T \tag{107}$$

The above equations set the framework within which the derivation of the BEFNTF algorithm is evolved. The filter taps vector is to be updated N steps ahead. To this end, the block-updating term $\mathbf{G}_{M,N}(n)\boldsymbol{\epsilon}_N(n)$ must first be computed. Note that the efficient computation of the block-updating term is the most important task in the algorithm because it refers to the most computationally thirsty part of the FNTF algorithm (i.e., the filtering part). The efficient computation of the involved matrix-by-vector product is based on a fast scheme for the computation of the exact filtering error vector $\boldsymbol{\epsilon}_N(n)$, and a proper transformation of matrix $\mathbf{G}_{M,N}(n)$ (called henceforth gain matrix).

Let us denote again as $[n-N+1,n]$ the current data block. Having assumed filter $\mathbf{c}_M(n-N)$ to be available, the goal is to compute $\mathbf{c}_M(n)$ via (105). The first step is to compute the exact a posteriori filtering errors $\boldsymbol{\epsilon}_M(n)$ at time steps, $i=n-N+1,\ldots,n$. To this end the following matrix is formed

$$\begin{bmatrix} e_M(n-N+1;1) & & & \\ e_M(n-N+2;1) & e_M(n-N+2;2) & & \\ \vdots & \vdots & \ddots & \\ e_M(n;1) & e_M(n;2) & \ldots & e_M(n;N) \end{bmatrix} \tag{108}$$

where

$$e_M(i;k)=y(i)-\mathbf{x}_M^T(i)\mathbf{c}_M(i-k) \tag{109}$$

is the a priori filtering error at time n based on the filter estimated at time $i-k$, i.e., $\mathbf{c}_M(i-k)$.

All the diagonal elements of the above triangular matrix are defined in terms of the filters $\mathbf{c}_M(n-k)$. Thus, these error variables can be directly computed by their definition. Then, by properly combining the filter update recursion as well as the step-up and step-down relations of FNTF, an efficient-order-recursive (Schur type, [10]) scheme is derived. This scheme computes all the remaining errors of the above matrix. That is, at each row, the recursive scheme starts from the respective diagonal element and runs backwards up to its leftmost element (see 3.1 and 3.2 of the algorithm of Table 11a). The required one-step-ahead, a priori filtering errors are the elements of the first column of the matrix in (108). In fact, in relation (105), the a posteriori filtering errors are required. These can easily be computed from their a priori counterparts (see 3.3 of the algorithm of Table 11a).

Note that several quantities associated with the forward and backward prediction problem of order P are needed in the above scheme. Some of them are computed in a similar way as above, i.e., via a Schur-type scheme (see Part 2), while some others are computed in Part 1 of the algorithm, using the prediction part of the stabilized FAEST algorithm.

Notice that the initialization phase of the Schur-type scheme of Part 2 requires the computation of the a priori prediction errors based on the estimates of the predictors at time $n-N$. These errors may be computed using either fast-FIR filtering techniques, as the one suggested in [134], or FFT-based techniques. The initialization phase of the Schur scheme of Part 3 must be treated in a different manner because the involved convolution is of a particular type. This is due to the fact that the filter length M may be several times longer than the block length N, hence, only a few outputs relative to the filter length are required. An FFT-based efficient scheme is proposed in [107] and [108] for computing efficiently this particular type of convolutions.

Having computed the filtering error vector in Parts 2 and 3 of the algorithm, the next step is to compute the block-updating term. The columns of the involved gain matrix $\mathbf{G}_{M,N}(n)$ are, by definition, the Kalman gain vectors of order M for all time steps within the current block. Properly combining the gain update relations of FNTF for all the involved time steps, it can be shown that the gain matrix $\mathbf{G}_{M,N}(n)$ can be written as a sum of three matrices

$$\mathbf{G}_{M,N}(n)=\mathbf{W}_{M,N}(n)+\mathbf{S}_{M,N}(n)-\mathbf{U}_{M,N}(n) \tag{110}$$

All three matrices are, once more, of a special form that can be exploited from a computational point of view. The columns of matrix $\mathbf{W}_{M,N}(n)$ are shifted versions of the gain vector at time $n-N+1$, thus, it lends itself to efficient matrix vector computations of the FIR filtering type. Matrices $\mathbf{S}_{M,N}(n)$ and $\mathbf{U}_{M,N}(n)$ are associated with the forward and backward predictors respectively, and their columns are computed recursively (see Part 4.1 of the algorithm). Since it is assumed that $P<<M$, each of these two matrices contains only a small fraction of nonzero terms.

Overall, a significant saving in complexity is offered as compared to the step-by-step FNTF. The main savings stem from

▲ The efficient computation of the filtering errors using the Schur-type schemes

▲ The efficient computation of the involved matrix-by-vector products using fast-FIR filtering techniques.

The total complexity of the algorithm of Table 11a, if all the involved convolutions are computed using the FFT-based techniques suggested in [107] and [108] for fast-FIR filtering, is given by

$$M_{tot}=12P+3N+\frac{P}{N}(7p-5)+\frac{M}{N}(5\nu-1)+\frac{2P^2}{N}+4p+\nu \quad MUPI \tag{111}$$

and

$$A_{tot}=14P+4N+\frac{P}{N}(21p-3)+\frac{M}{N}(15\nu+6)+\frac{2P^2}{N}+12p+3\nu \quad ADPI \tag{112}$$

where $N=2^{\nu}$ and $P=2^{p}$.

Block-Exact Stabilized FTF

Block-exact implementations of the fast-RLS scheme have been developed in [136] and [137]. Their algorithm has been called fast-subsampled-updating stabilized FTF (FSU-SFTF). The FSU-SFTF algorithm is based on the stabilized version of the FTF algorithm [69].

The derivation of the FSU-SFTF algorithm is based on the interpretation of the FTF (and SFTF) algorithm as a successive application of rotation matrices to a specific set of filters. Indeed, all the one-step-ahead, filter-update relations of SFTF can be compactly written in the form

$$\begin{bmatrix} \mathbf{w}_M^T(n) & 0 \\ 1 & \mathbf{a}_M^T(n) \\ \mathbf{b}_M^T(n) & 1 \\ \mathbf{c}_M^T(n) & 0 \end{bmatrix}=\Theta(n)\begin{bmatrix} 0 & \mathbf{w}_M^T(n-1) \\ 1 & \mathbf{a}_M^T(n-1) \\ \mathbf{b}_M^T(n-1) & 1 \\ \mathbf{c}_M^T(n-1) & 0 \end{bmatrix} \tag{113}$$

where $\Theta(n)$ is a 4×4 step-by-step rotation matrix defined as

$$\Theta(n)=\Theta_4(n)\Theta_3(n)\Theta_2(n)\Theta_1(n). \tag{114}$$

The 4×4 rotation matrices $\Theta_i(n)$, $i=1,2,3,4$ are in turn given by

$$\Theta_4(n)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \bullet & 0 \\ 0 & 0 & 1 & \bullet \\ \alpha & 0 & 0 & 1 \end{bmatrix} \quad \Theta_3(n)=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{115}$$

$$\Theta_2(n)=\begin{bmatrix} 1 & 0 & c & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \Theta_1(n)=\begin{bmatrix} 1 & e & 0 & 0 \\ d & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{116}$$

where a is the filtering error $\epsilon_M(n)$; b is the backward-prediction a posteriori error $\epsilon_M^b(n)$ (computed in terms of the a priori one); c is the element $-w_{M+1}^{M+1}(n)$ of the Kalman gain vector; d is the a posteriori forward-prediction error $\epsilon_M^f(n)$; and finally, $e=\lambda^{-1}e_M^f(n)\alpha_M^{-1}(n-1)$, with $\alpha_M(n-1)$ being the power of the Kalman gain output. All these quantities are computed like the SFTF in terms of the filters at time $n-1$.

Writing (113) for N consecutive time steps, within the current block, and combining the resulting relations, the following can be easily obtained

$$\begin{bmatrix} \mathbf{w}_M^T(n) & 0 \\ 1 & \mathbf{a}_M^T(n) \\ \mathbf{b}_M^T(n) & 1 \\ \mathbf{c}_M^T(n) & 0 \end{bmatrix}=\Theta(n;N)\begin{bmatrix} 0 & \mathbf{w}_M^T(n-N) \\ 1 & \mathbf{a}_M^T(n-N) \\ \mathbf{b}_M^T(n-N) & 1 \\ \mathbf{c}_M^T(n-N) & 0 \end{bmatrix} \tag{117}$$

where $\Theta(n;N)$ is 4×4 block-rotation matrix, which is defined as the product of all intermediate step-by-step rotation matrices, i.e.,

$$\Theta(n;N)=\Theta(n)\Theta(n-1)\ldots\Theta(n-N+1). \tag{118}$$

The computation of the above rotation matrix requires the one-step-ahead filtering and prediction error variables for all time steps within the block. Using the filter estimates at time $n-N$, the multi-step-ahead filter outputs over the next N time instants can be computed. Then,

Table 11b. The FSU-SFTF Algorithm.

Let $[n-N+1,n]$ be the current data block and assume that all filter estimates are available until time instant $n-N$.

Part 1. Computation of the multi-step-ahead filter outputs.

Using fast-FIR filtering techniques compute the quantities:

$$\begin{bmatrix} \mathbf{e}_N^{wT}(n|n-N) \\ \mathbf{e}_N^{fT}(n|n-N) \\ \mathbf{e}_N^{bT}(n|n-N) \\ \mathbf{d}_N^T(n|n-N) \end{bmatrix}=\begin{bmatrix} 0 & \mathbf{w}_M^T(n-N) \\ 1 & \mathbf{a}_M^T(n-N) \\ \mathbf{b}_M^T(n-N) & 1 \\ -\mathbf{c}_M^T(n-N) & \bullet \end{bmatrix}\tilde{\mathbf{X}}_{M+1,N}(n)$$

Part 2. Computation of the step-by-step rotation matrices using the SFTF-Schur algorithm.

Input: $\mathbf{e}_N^{wT}(n|n-N)$, $\mathbf{e}_N^{fT}(n|n-N)$, $\mathbf{e}_N^{bT}(n|n-N)$, $\mathbf{d}_N^T(n|n-N)$

Output: $\Theta(i)$, $i=n-N+1,\ldots n$

Part 3. Computation of the block rotation matrix.

$\Theta(n;N)=\Theta(n)\Theta(n-1)\ldots\Theta(n-N+1)$

Part 4. Block updating of the involved filters.

$$\begin{bmatrix} \mathbf{w}_M^T(n) & 0 \\ 1 & \mathbf{a}_M^T(n) \\ \mathbf{b}_M^T(n) & 1 \\ \mathbf{c}_M^T(n) & 0 \end{bmatrix}=\Theta(n;N)\begin{bmatrix} \bullet & \mathbf{w}_M^T(n-N) \\ 1 & \mathbf{a}_M^T(n-N) \\ \mathbf{b}_M^T(n-N) & 1 \\ \mathbf{c}_M^T(n-N) & 0 \end{bmatrix}$$

based on these filter outputs, and using the so-called SFTF-Schur algorithm [136], the required one-step-ahead error quantities can be computed for each time instant within $[n-N,n]$.

The FSU-SFTF algorithm is summarized in Table 11b. From the form of the computations involved in Part 1 of the algorithm, it is easily seen that fast-FIR filtering techniques can be directly applied. Efficient convolution methods are also applied in Part 4 of the algorithm. Due to the shifting properties of the Kalman gain in (117), all the quantities involved in this block-updating recursion are transformed in the Z-domain, and the filters are associated with polynomials. The resulting structure involves 12 convolutions of an N-order polynomial with an M-order polynomial. If the orders of the polynomials are relatively large, as it is often the case in many practical situations, the convolutions may be implemented efficiently using FFT-based techniques.

The total computational complexity of the FSU-SFTF algorithm equals

$$\left(17+8\frac{M+1}{N}\right)\frac{FFT(2N)}{N}+32\frac{M}{N}+10N \quad MUPI, \tag{119}$$

where the FFT is performed using the split-radix technique. Hence, $FFT(2N)$ is equal to $N\log_2(2N)$ real multiplications for real signals [140]. Only the number of multiplications per sample is given above. The number of additions is somehow higher [137].

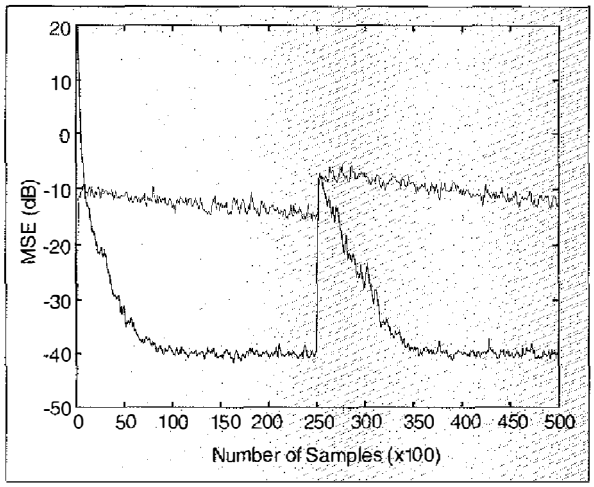
Block-Exact RLS: A block-exact version of the classical RLS algorithm, the so-called fast-subsampled-updating RLS (FSU-RLS) algorithm, has been developed in [138]. Its derivation evolves in a different path than FSU-SFTF, and it is worth pointing out that the resulting computational load is similar to that of FSU-SFTF.

To illustrate the complexity of the block-exact algorithms considered so far, let us consider an FIR filter with length M=2048, which is not untypical for acoustic echo-cancellation applications. For a block length N=128, FELMS requires 738 multiplications per sample; the BEFNTF requires 1140 multiplications per sample, for a prediction order P=14; and the FSU-SFTF requires 3208 multiplications per sample.

Block-Exact Affine-Projection Algorithm

A block-exact version of the fast-affine-projection algorithm has been developed recently [98]. The derivation of the algorithm follows the same line as the one for FELMS. Let the filter length M be an integer multiple of the projection order L. The algorithm is based on the auxiliary filters $\mathbf{h}_M(n)$, $\mathbf{q}_L(n)$, defined in Table 11, and the associated error variable $\mathbf{e}^b(n)$ for all time instants $n-N+1,..n$, within the current block, i.e.,

$$e^b(n-i)=y(n-i)-\mathbf{x}^T(n-i)\mathbf{h}_M(n-i-1),$$
$$i=N-1,N-2,..,0. \tag{120}$$



▲ *2. Learning curve for the NLMS (red) and the PN-TD-LMS (blue) for the stationary input signal.*

Combining the update equation for $\mathbf{h}_M(n)$ given in Table 11, with the above set of equations, the following block recursion is obtained

$$\tilde{\mathbf{e}}_N^b(n)=\tilde{\mathbf{y}}_N(n)-\tilde{\mathbf{X}}_{M,N}^T(n)\mathbf{h}_M(n-N)$$
$$-\mathbf{S}_{N,N}(n-N+1)\mathbf{q}_L(n) \tag{121}$$

where

$$\tilde{\mathbf{e}}_N^b(n)=[e^b(n-N+1)e^b(n-N+2)\cdots e^b(n)]^T \tag{122}$$

and matrix $\mathbf{S}_{N,N}(n-N+1)$ is a time-delayed version of $\mathbf{S}_{N,N}(n)$, as defined in (97) for the FELMS. The filtering errors $e(n-i)$ within the current block are then estimated via

$$e(n-i)=e^b(n-i)+\mathbf{r}_L^T(n-i)\mathbf{q}_L(n-i-1),$$
$$i=0,1,..,N-1 \tag{123}$$

and the block update recursion for $\mathbf{h}_M(n)$ is given by

$$\mathbf{h}_M(n)=\mathbf{h}_M(n-N)+\mu\tilde{\mathbf{X}}_{M,N}(n)\mathbf{q}_L(n) \tag{124}$$

The computational savings are obtained by performing the involved convolutions in (123) and (124), in the same efficient way as FELMS.

Approximate Block Implementations in the Time Domain

The generic block version of an adaptive algorithm can be cast as a generalization of (28), i.e.,

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)-\mu(n)\mathbf{W}(n)\mathbf{g}_M(n),$$
$$n=0,N,2N,.. \tag{125}$$

where N is again the block size and $\mathbf{g}_M(n)$ is an estimate of the gradient at time instant n, i.e.,

$$\mathbf{g}_M(n)=\hat{\mathbf{V}}_{\mathbf{c}_M(n-N)}(\hat{\mathcal{E}}_n[e^2(n)]) \tag{126}$$

By different choices of matrix $\mathbf{W}(n)$ and gradient vector $\mathbf{g}_M(n)$, the block versions of the various algorithms considered in "Stochastic Approximation Methods" can be obtained.

Block LMS

Assuming $\mathbf{W}(n)=\mathbf{I}_M$ and the expectation operator in (126) to be the sliding-window approximation as in (39), the block-LMS (BLMS) algorithm results, i.e.,

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)+\mu(n)\widetilde{\mathbf{X}}_{M,L}\widetilde{\mathbf{e}}_L(n|n-N),$$
$$n=0,N,2N,\ldots \tag{127}$$

$$\widetilde{\mathbf{e}}_L(n|n-N)=\widetilde{\mathbf{y}}_L(n)-\widetilde{\mathbf{X}}^T_{M,L}(n)\mathbf{c}_M(n-N). \tag{128}$$

In other words, the update direction for this case is given by

$$\mathbf{v}(n)=-\mathbf{g}_M(n)=\tfrac{L}{2}\widetilde{\mathbf{X}}_{M,L}\widetilde{\mathbf{e}}_L(n|n-N). \tag{129}$$

The product $\widetilde{\mathbf{X}}_{M,N}\widetilde{\mathbf{e}}_N(n|n-N)$ can be computed via fast convolutional schemes [49].

Observe that two design variables are associated with these algorithms, namely the update block length N and the length of the data size memory, L. Most of the early algorithms used $L=N$, and in particular, $L=N=M$. The emancipation from this constraint requires more sophisticated ways to compute convolutions than a straightforward FFT.

As it was the case with the sample-by-sample versions, $\mu(n)$ can be chosen as either constant or time-varying. In the latter case, line-search schemes have been adopted to optimize convergence speed, and the resulting step size is given by

$$\mu(n)=\frac{L\widetilde{\mathbf{e}}_L^T(n|n-N)\mathbf{H}\widetilde{\mathbf{e}}_L(n|n-N)}{2\widetilde{\mathbf{e}}_L^T(n|n-N)\mathbf{H}^2\widetilde{\mathbf{e}}_L(n|n-N)}, \tag{130}$$

where

$$\mathbf{H}=\widetilde{\mathbf{X}}^T_{M,L}(n)\widetilde{\mathbf{X}}_{M,L}(n).$$

The resulting scheme is known as optimum-block-adaptive (OBA) algorithm [30]. Note that the step size in the OBA algorithm is the same for all the adaptive filter coefficients. Efforts to use different step sizes for the various coefficients have also been made [31], [32].

Another path to accelerate the convergence speed of the BLMS family is to use suitably chosen unitary transformations of the input signal, as it was the case with the sample by sample versions (see "Accelerating the Adaptive Gradient Methods"). In such cases, the update direction should be

$$\mathbf{v}(n)=\mathbf{S}\widetilde{\mathbf{X}}_{M,L}\widetilde{\mathbf{e}}_L(n|n-N)\equiv\mathbf{U}_{M,L}(n)\widetilde{\mathbf{e}}_L(n|n-N) \tag{131}$$

which finally results in the counterparts of (57) and (62), i.e.,

$$\hat{\mathbf{c}}_M(n)=\hat{\mathbf{c}}_M(n-N)+\mu(n)\mathbf{R}^{-1}_{TD}(n)\mathbf{U}_{M,L}\widetilde{\mathbf{e}}_L(n|n-N) \tag{132}$$

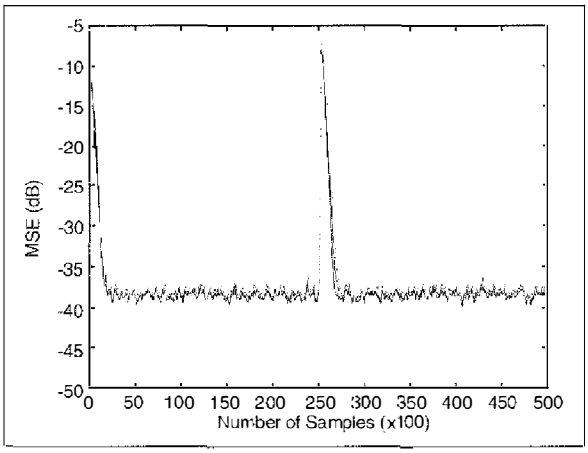
$$\widetilde{\mathbf{e}}_L(n|n-N)=\widetilde{\mathbf{y}}_L(n)-\mathbf{U}^T_{M,L}(n)\hat{\mathbf{c}}_M(n-N). \tag{133}$$

The distinct difference between the approximate-block algorithms, and the block-exact versions, is that in the former schemes, the involved errors are computed in terms of the filter estimate, corresponding to the beginning of each block. In contrast, the block-exact versions use the true errors, at the various time instants within each block.

Block-Quasi-Newton Techniques

These schemes result from the generic form (125) for gradient $\mathbf{g}_M(n)$ as in (129), and for $\mathbf{W}(n)$ having any of the forms discussed in "Adaptive Quasi-Newton Algorithms." A well-known scheme is obtained by assuming Toeplitz approximation of matrix $\mathbf{W}(n)$, as in (75), and is known as a self-orthogonalizing, block-adaptive filter (SOBAF) [81]. The computational complexity of the SOBAF algorithm is $O(M)$ per time step. However, if instead of the Levinson algorithm, the FFT-based technique of [141] is applied to the solution of the involved Toeplitz system, then the complexity reduces to $O(\log M)$.

The preconditioning techniques that recently appeared in [90], [142] can also be set in the same context as the SOBAF technique and, in fact, they can be regarded as self-orthogonalization methods. The Toeplitz preconditioned OBA (TOBA) algorithm in [90] has a similar form as SOBAF, with the additional feature of a time-varying step size, which is chosen to minimize the block MSE. Furthermore, it is suggested that circulant preconditioners can be employed. The latter technique is



3. Learning curve for the SFRLS (blue), the FNTF (red), and the APA (green) for the stationary input signal.

slightly inferior to TOBA, but it lends itself to very efficient implementations.

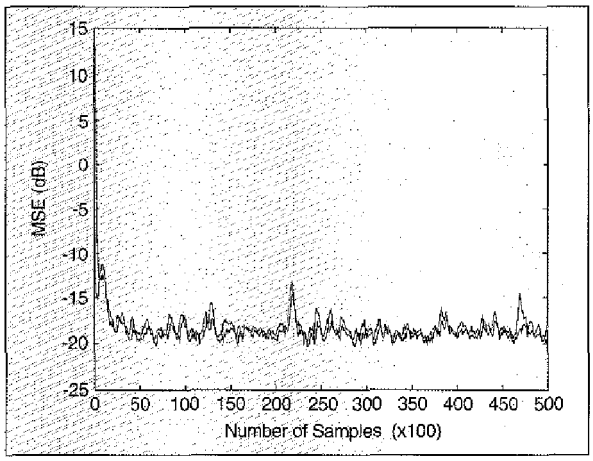
Another choice is to use in place of $\mathbf{W}(n)$ a pseudo-inverse matrix, similar in form to that given in (79), resulting in

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)+\mu\widetilde{\mathbf{X}}_{M,L}(n)$$
$$[\widetilde{\mathbf{X}}^T_{M,L}(n)\widetilde{\mathbf{X}}_{M,L}(n)]^{-1}\widetilde{\mathbf{e}}_L(n|n-N). \quad (134)$$

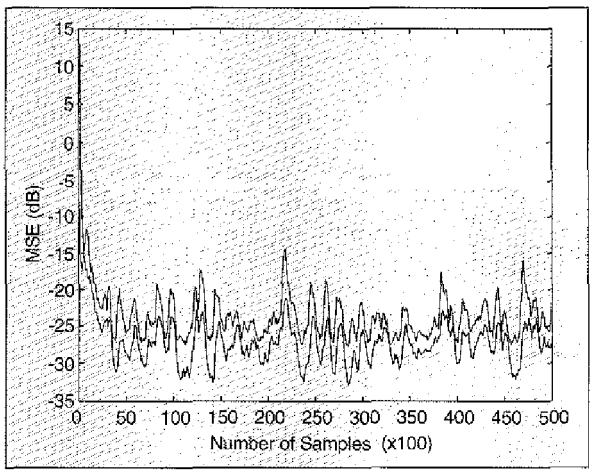
This algorithm is, in fact, a block version of the affine-projection algorithm. The above block update (with $\mu=1$) first appeared in [143]. The same block-adaptive algorithm was rederived in [32], but using different arguments. In fact, the authors in [32] derived a block APA in an effort to develop a BLMS with time-varying step size and no constraints in the block length with respect to the filter order. The resulting algorithm is equivalent to the block APA of [143] for $N\leq M$. In their algorithm, called the general-optimum-block-adaptive (GOBA) algorithm, the correction term in the block update recursion of BLMS is chosen to minimize the squared norm of the a posteriori error vector. This constitutes an interesting alternative interpretation of the block APA. In [99], the same block structure is reinvestigated, and a step size is added to the correction term, resulting in the form given in (134). Note that the block update in (134) contains as extremal cases the normalized-block LMS and the block RLS. The step size μ can be controlled according to the squared norm of the estimation error within the current block. As suggested in [99], the computational complexity can be further reduced by applying the generalized Levinson algorithm [144] for the solution of the system

$$[\widetilde{\mathbf{X}}^T_{M,L}(n)\widetilde{\mathbf{X}}_{M,L}(n)]^{-1}\widetilde{\mathbf{e}}_L(n|n-N),$$

which is involved in the correction term of the above recursion.



▲ *4. Comparison of the total MSE (with external noise included) curves, for two different algorithms.*



▲ *5. Comparison of the residual echo curves, for two different algorithms.*

An alternative path was followed in [145]. In the scheme derived there, matrix $\mathbf{W}(n)$ is replaced by an approximation of the inverse autocorrelation matrix involved in the FNTF. By adopting the assumption that the input autocorrelation matrix does not practically change within a time interval of at least M steps, the banded structure of the inverse autocorrelation matrix can be further simplified. Note that the same reasonable assumption was made in [82] for the derivation of the step-by-step FQN algorithm. The block recursion of the resulting algorithm is given by

$$\mathbf{c}_M(n)=\mathbf{c}_M(n-N)+\mu[\mathbf{p}^1_M(n)+\mathbf{p}^2_M(n)], \quad (135)$$

where

$$\mathbf{p}^1_M(n)=\begin{bmatrix}\widetilde{\mathbf{X}}_{P,L}(n)\widetilde{\mathbf{e}}_L(n|n-N)\\ \mathbf{0}_{M-P}\end{bmatrix} \quad (136)$$

and

$$\mathbf{p}^2_M(n)=\mathbf{C}_{M,M-P}(n)\mathbf{C}^T_{M,M-P}(n)\widetilde{\mathbf{X}}_{M,L}(n)\widetilde{\mathbf{e}}_L(n|n-N) \quad (137)$$

The matrix $\mathbf{C}_{M,M-P}(n)$ is circulant and its columns are shifted versions of the P-th order backward prediction filter, which, according to the above assumption, remains constant for M steps. The two correction vectors $\mathbf{p}^1_M(n)$ and $\mathbf{p}^2_M(n)$ can be efficiently computed using fast convolution techniques. Note that the second correction vector is formed as a succession of three convolutions. The FFT-based implementation of the above block-adaptive algorithm has a complexity comparable to that of BLMS, offering, however, a considerably faster convergence.

Simulation Results

Simulation results from typical system identification experiments will be presented in this section. We have compared the performance of the main algorithmic techniques

discussed in the previous sections. From the several variations of these algorithms, we have chosen to experiment with well-established sample-by-sample versions. Obviously the results from this performance comparison are valid for their block-exact counterparts as well.

The input time series used in the first experiment was a stationary AR process of order 14. The AR parameters used to generate this process were those estimated (using standard techniques) from a segment of real speech corresponding to the Greek phoneme "ε". This AR process was used as input to an FIR system of order 256. The impulse response was a typical impulse response of the acoustic echo path of a car enclosure. At the output of the FIR system white Gaussian noise was added, resulting in an SNR equal to about 30 dB. Thus, the noise contaminated output of the system is given by

$$y(n)=\sum_{i=1}^{256} h_i x(n-i+1)+\eta(n), \tag{138}$$

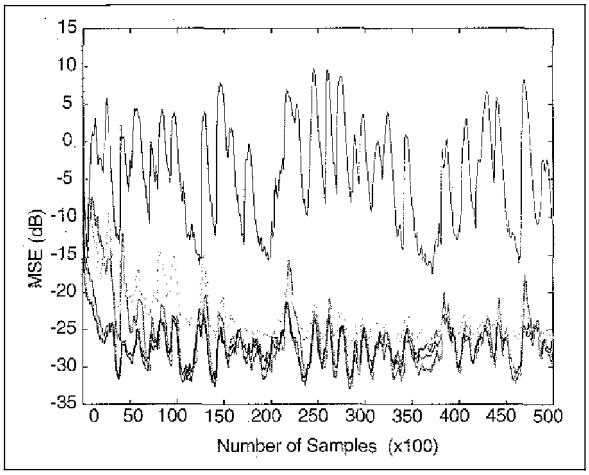
where $x(n)$ are the input samples of the AR(14) process, h_i are the unknown system impulse response coefficients and $\eta(n)$ are the additive noise samples. The systems changed at time instant n=25000 from **h** to −**h**. The unknown FIR system was estimated using:

- The modified stabilized-FRLS algorithm, Table 7, with M=256, λ=.999, and ρ=.9999
- The modified FNTF algorithm, Table 9, with M=256, P=14, λ=.999, and ρ=.9999
- The APA algorithm, Table 10, with M=256, L=14, μ=2, and $\delta=10^{-4}$
- The NLMS algorithm, Table 1, with M=256, α=0.5, and $\beta=10^{-4}$
- The power-normalized transform-domain LMS, Table 3d, with M=256

The filtering error power for each case was computed by averaging the squared instantaneous filtering errors over an exponentially decaying window, with effective memory equal to 128 time instants. The simulation results are shown in Figs. 2 and 3.

In the second series of experiments, the input time series is a real nonstationary speech signal taken with a sampling frequency of 8 kHz. The acoustic path is the same as the one used before. At the output of the acoustic path a white noise with SNR=20 dB was added.

Note that identifying the unknown system is equivalent to reducing the acoustic echo, which disturbs normal speaking. From the total squared filtering error curves we have subtracted the output noise in order to focus on the misadjustment itself, i.e., the residual echo. Before showing the main results of this experiment and in order to emphasize the previous point, we have plotted, as an example, the MSE and the residual echo curves for two of the above algorithms. The MSE plots in Fig. 4 may lead to the erroneous conclusion that both algorithms converge to the same steady state error, and therefore, what it remains is to compare their initial convergence performance. However as shown in Fig. 5, the residual echo curves are well separated by almost 5 dB. Thus, the presence of the external noise, which is usually the dominating component, may be a misleading factor for performance comparisons.



6. Learning curve for the NLMS (yellow), the SFRLS (blue), the FNTF (red), and the APA (green) for the speech input signal and the time-invariant channel.

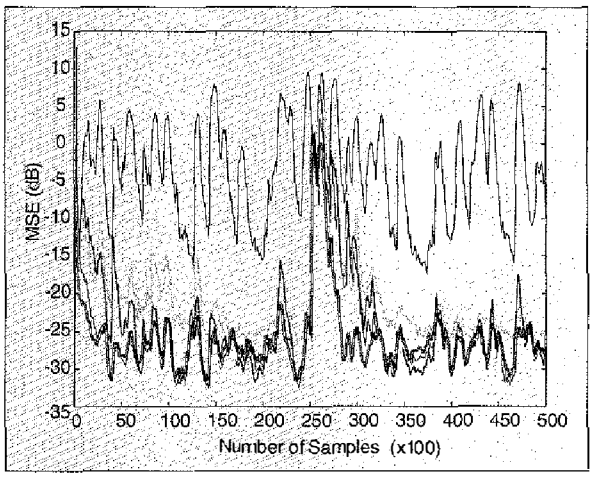
The final results, concerning the convergence and tracking performance of NLMS, stabilized fast RLS, the FNTF, and APA, are plotted in Figs. 6 and 7. In the case of Fig. 6, the unknown system remains invariant, while, in the case of Fig. 7, the systems changes at time instant n=25000 from **h** to −**h**. In each figure, the magenta curve corresponds to the measured echo while the blue, yellow, red, and green curves correspond to the residual echo signals of stabilized fast RLS, NLMS, FNTF, and APA, respectively. The window size here was taken equal to 128 and the steady-state residual errors were always the same. The filtering order in all cases was 256, while the prediction order for FNTF as well as the projection order for APA were both equal to 16. The forgetting factor for both stabilized fast RLS and FNTF was equal to 0.9984. The step sizes for NLMS and APA were chosen equal to 0.4096 and 0.012, respectively. Soft-constrained initialization was used in the case of stabilized fast RLS and FNTF with initial values equal to σ_x^2 and $10\sigma_x^2$, respectively, where σ_x^2 is an estimate of the input power.

From the above simulations one can conclude that:

- The use of preconditioning on the LMS greatly improves its performance.
- The fast RLS, the FTNF, and the APA exhibit comparable performance, provided that the trimming parameters are properly tuned in each case, so that all methods result in approximately the same steady-state error that guarantees a fair comparison of the schemes.

Conclusion

In this article, a unified view of algorithms for adaptive transversal FIR filtering and system identification has



▲ 7. Learning curve for the NLMS (yellow), the SFRLS (blue), the FNTF (red), and the APA (green) for the speech input signal and the time-varying channel.

been presented. Wiener filtering and the stochastic approximation are the origins from which all the algorithms have been derived, via a suitable choice of iterative optimization schemes and appropriate design parameters. Following this philosophy, the LMS algorithm and its offsprings have been presented and interpreted as stochastic approximations of iterative deterministic steepest descent optimization schemes. On the other hand, the RLS and the quasi-RLS algorithms, like the quasi-Newton, the FNTN, and the APA algorithm, have been derived as stochastic approximations of iterative deterministic Newton and quasi-Newton methods. Fast implementations of these methods have been discussed. Block-adaptive, and block-exact adaptive filtering have also been considered. The performance of the adaptive algorithms has been demonstrated by computer simulations, in the context of the acoustic echo cancellation and system identification.

George-Othon A. Glentis received his BS in Physics and Ph.D. in Informatics, both from the University of Athens, Greece. From 1988-1991, he held a four-year research fellowship from the Institute of Informatics of the National Center for Physical Science, Democritos. From 1991-1993, he served with the Greek Air Force. From 1993-1995, he was with the Electrical Engineering Department of the University of Twente, The Netherlands, and with the Faculte des Sciences Appliquees, Universite Catholique de Louvain, Belgium, as an EU HCM research fellow. From 1996-1997, he was with the Department of Informatics, University of Athens, as an EU TMR research fellow. He is currently an assistant professor at the Department of Electronics, Technological Education Institute of Hereklion, Branch at Chania.

Kostas Berberidis received his BS in electrical engineering from the Democritos University, Thrace, Greece, and his Ph.D. in computer engineering and informatics from the University of Patras, Greece. From 1986-1990, he was a research assistant in the Computer Technology Institute (CTI), Patras. During 1991, he served in the Greek Army, in the Speech Processing Lab of the National Defense Research Center. From 1992-1997, he was a researcher at CTI, except for the year 1994, which he spent as a post doctorate fellow at Centre Commund'Etudes de Telediffusion et Telecommunications, Rennes, France. Since December 1997, Berberidis has been an assistant professor at the Department of Computer Engineering and Informatics at the University of Patras.

Sergios Theodoridis received his BS in physics from the University of Athens, Greece, and his MS and Ph.D. degrees from the University of Birmingham, UK. From 1981-1995, he was with the Department of Computer Engineering at the University of Patras, and since 1995 has been with the Department of Informatics at the University of Athens, where he is currently a professor in Signal Processing and Communications.

References

[1] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, MIT Press, 1982.

[2] G. Goodwin and K. Sin, *Adaptive Filtering, Prediction and Control*, Prentice Hall, 1984.

[3] T. Söderström and P. Stoica, *System Identification*, Prentice Hall, 1989.

[4] M.L. Honig and D.G. Messerschmitt, *Adaptive Filters Structures Algorithms and Application*, Kluwer Academic Publishers, 1984.

[5] S.T. Alexander, *Adaptive Signal Processing, Theory and Applications*, Springer-Verlag 1986.

[6] C.F. Cowan and P.M. Grant, *Adaptive Filters*, Prentice-Hall, 1985.

[7] B. Widrow and S. Stearns, *Adaptive Signal Processing*, Prentice Hall, New Jersey, 1985.

[8] S. Haykin, *Adaptive Filter Theory*, Third Edition, Prentice Hall, 1996.

[9] M. G. Bellanger, *Adaptive Digital Filters and Signal Analysis*, Marcel Dekker, 1997.

[10] N. Kalouptsidis and S. Theodoridis, Eds., *Adaptive System Identification and Signal Processing Algorithms*, Prentice Hall 1993.

[11] N. Kalouptsidis, *Signal Processing Systems, Theory and Design*, Wiley 1997.

[12] W. Jenkins et al, *Advanced Concepts in Adaptive Signal Processing*, Kluwer 1996.

[13] G. Zeilniker and F. Taylor, *Advanced Digital Signal Processing*, Dekker 1994.

[14] P.S.R. Diniz, *Adaptive Filtering Algorithms and Practical Implementation*, Kluwer Academic Publishers, 1997.

[15] G. Pflug, *Optimization of Stochastic Models*, Kluwer Academic, 1996.

[16] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximation*, Springer-Verlag, 1987.

[17] L. Ljung, G. Pflug, and H. Walk, *Stochastic Approximation and Optimization of Random Systems*, Birkhauser Verlag, 1992.

[18] H. Kusner and G. Yin, *Stochastic Approximation Algorithms and Applications*, Springer 1997.

[19] A. Albert and L. Gardner, *Stochastic Approximation and Nonlinear Regression*, MIT Press, 1967.

[20] M.T. Wasan, *Stochastic Approximation*, Cambridge 1969.

[21] M. Nevelson, and R. Hasminskii, *Stochastic Approximation and Recursive Estimation*, American Mathematical Society, 1973.

[22] D. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Sons, 1968.

[23] D. Pierre, *Optimization Theory with Applications*, Dover, 1969.

[24] D. Farden, "Stochastic approximation with correlated data," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 1, pp. 105-113, Jan. 1981.

[25] M. Kouritzin, "On the convergence of linear stochastic approximation procedures," *IEEE Trans. Inf. Theory*, vol. IT-42, pp.1305-1309, July 1996.

[26] W. Gardner, "Learning characteristics of stochastic gradient descent algorithms: A general study, analysis, and critique," *Signal Processing*, vol. 6, pp. 113-133, 1984.

[27] W. Sethares, "The Least Mean Square Family," in *Adaptive System Identification and Signal Processing Algorithms*, N. Kalouptsidis and S. Theodoridis, Eds., Prentice-Hall, 1993.

[28] G. Clark, S. Mitra, and S. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits and Systems*, vol. CAS-26, no. 6, pp. 584-592, June 1981.

[29] G. Clark, S. Parker, and S. Mitra, "A unified approach to time and frequency domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 5, pp. 1073-1083, Oct. 1983.

[30] W. Mikhael and F. Wu, "Fast algorithms for block FIR adaptive digital filtering," *IEEE Trans. on Circuits and Systems*, vol. CAS-34, no. 10, pp. 1152-1160, Oct. 1987.

[31] W. Mikhael and F. Wu, "A fast block FIR adaptive digital filtering algorithm with individual adaptation of parameters," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1-10, Jan. 1989.

[32] T. Wang and C. Wang, "On the optimum design of the block adaptive FIR filter," *IEEE Trans. on Signal Proc.*, vol. 41, no. 6, pp. 2131-2140, June 1994.

[33] T. Wang and C. Wang, Comments on "A fast block FIR adaptive digital filtering algorithm with individual adaptation of parameters," *IEEE Trans. Circuits and Systems II*, vol. 39, no. 4, pp. 254-256, April 1992.

[34] M. Mboup, M. Bonnet, and N. Bershad, "LMS coupled adaptive prediction and system identification: a statistical model and transient mean analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-42, no. 10, pp. 2607-2615, Oct. 1994.

[35] J. Doherty and R. Porayath, "A robust echo canceler for acoustic environments," *IEEE Trans. Circuits and Syst. II*, vol. 44, pp. 389-398, May 1997.

[36] C. Breining, P. Dreiseitel, E. Hansler, A. Mader, B. Nitsch, H. Puder, T. Schertler, G. Schmidt, and J. Tilp, "Acoustic echo control—An application of very high order adaptive filters," *IEEE Signal Processing Magazine*, this issue.

[37] M. Dentino, J. McCool, and B. Widrow, "Adaptive filtering in the frequency domain," *Proc. IEEE*, vol. 66, no 12, pp. 1658-1659, Dec. 1978.

[38] E. Ferrara, "Fast implementation of the LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 4, pp. 474-475, Aug. 1980.

[39] S. Narayan, A.M. Peterson, and M.J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust. Speech, Signal Proc.*, vol. ASSP-31, pp. 609-615, June 1983.

[40] J.C. Lee and C.K. Un, "Performance of transform domain adaptive digital filters," *IEEE Trans. Acoust. Speech, Signal Proc.*, vol. ASSP-34, pp. 499-510, June 1986.

[41] P. Bondyopadhyay, "Application of running Hartley transform in adaptive digital filtering," *IEEE Proc.*, pp. 1370-1372, Oct. 1988.

[42] D.F. Marshall, W.K. Jenkins, and J.J. Murphy, "The use of orthogonal transforms for improving performance of adaptive filters, *IEEE Trans. Circuits and Systems*, CAS-36, pp. 474-484, April 1989.

[43] J. Lee and C. Un, "A reduced structure of the frequency domain adaptive digital filter," *Proc. IEEE*, vol. 72, pp. 1816-1818, Dec. 1984.

[44] P. Sommen, P. van Gerwen, H. Kotmans, and A. Janssen, "Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function," *IEEE Trans. Circuits and Systems*, vol. CAS-34, no. 7, pp. 788-798, July 1987.

[45] W. Mikhael and A. Spanias, "Comparison of several frequency domain LMS algorithms," *IEEE Trans. Circ. and Systems*, vol. CAS-34, no. 5, pp. 586-588, May 1987.

[46] W. Mikhael and A. Spanias, "A fast frequency domain adaptive algorithm," *Proc. IEEE*, vol. 76, no. 1, pp. 80-82, Jan. 1988.

[47] W. Mikhael and A. Spanias, "Performance enhancement of the frequency domain adaptive filter," *IEEE Proc. ISCAS-86*, pp. 349-352, May 1986.

[48] J. Lee and C. Un, "Performance analysis of frequency domain Block adaptive digital filters," *IEEE Trans. Circuits and Systems*, vol. CAS-36, no. 2, pp. 173-189, Feb. 1989.

[49] J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Magazine*, vol. 9, pp. 14-39, Jan. 1992.

[50] V. Sanchez, P. Garcia, et al, "Diagonalizing properties of the DCT," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-43, no. 11, Jan. 1995.

[51] J. Chao, H. Perez, and S. Tsujii, "A fast adaptive filter algorithm using eigenvalue reciprocals as stepsizes," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, no. 8, pp. 1343-1353, Aug. 1990.

[52] B. Farhang-Boroujeny, "Application of orthonormal transforms to implementation of quasi-LMS/Newton algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-41, no. 3, pp. 1400-1404, March 1993.

[53] B. Farhang-Boroujeny and S. Gazor, "Selection of orthonormal transforms for improving the performance of the transform domain normalized LMS algorithm," *IEEE Proceedings-F*, vol. 139, No. 5, pp. 327-335, Oct. 1992.

[54] F. Beufays, "Transform-domain adaptive filters: An analytical approach," *IEEE Trans. Acoust. Speech, Signal Proc.*, vol. ASSP-42, pp. 422-431, Feb. 1995.

[55] L. Ljung, M. Morf, and D. Falconer, "Fast calculations of gain matrices for recursive estimation schemes," *Int. Journal of Control*, vol. 27, pp.1-19, Jan. 1978.

[56] G. Carayannis, N. Kaloupsidis, and D. Manolakis, "Fast recursive algorithms for a class of linear equations," *IEEE Trans. ASSP-30*, no 2, pp. 227-239, April 1982.

[57] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans., Acoust., Speech, Signal Process.*, vol ASSP-31, pp. 1394-1402, Dec. 1983.

[58] J. Cioffi and T. Kailath, "Fast recursive LS transversal filters for adaptive processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, pp. 304-337, April 1984.

[59] G. Carayannis, D. Manolakis, and N. Kalouptsidis, "A unified view of parametric processing algorithms for prewindowed signals," *Signal Processing*, vol. 10 (1986), pp. 335-368.

[60] N. Kalouptsidis, G. Carayannis, and D. Manolakis, "A fast covariance type algorithm for sequential LS filtering and prediction," *IEEE Trans. Autom. Control*, vol. AC-29, pp. 752-5, 1984.

[61] D. Slock and T. Kailath, "Fast transversal RLS algorithms," in *Adaptive System Identification and Signal Processing Algorithms*, N. Kalouptsidis and S. Theodoridis, Eds., Prentice-Hall, 1993.

[62] E. Eleftheriou and D.D. Falconer, "Tracking performance and steady-state performance of RLS adaptive filter algorithms," *IEEE Trans. ASSP-34*, pp. 1097- 1109, Sept. 1986.

[63] D.W. Lin, "On digital implementation of fast Kalman algorithms," *IEEE Trans. ASSP- 32*, pp. 998-1005, Oct. 1984

[64] L. Ljung and L. Ljung, "Error propagation properties of recursive least squares adaptation algorithms," *Automatica*, vol. 21, no. 2, 1985.

[65] M. Verhaegen, "Improved understanding of the loss-of-symmetry phenomenon in the conventional Kalman filter," *IEEE Trans. Autom. Control*, vol. AC-34, pp.331-333, 1989.

[66] M. Verhaegen, and P. Van Dooren, "Numerical aspects of different Kalman filter implementation," *IEEE Trans. Autom. Control*, vol. AC-31, pp. 907-17, 1986.

[67] S. Ardalan and S. Alexander, "Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time-varying systems," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-35, pp. 770-83, 1987.

[68] J. Botto, and G. Moustakides, "Stabilizing the fast Kalman algorithms," *IEEE Trans. Acoust. Speech, Signal Processing, ASSP-37*, pp. 1342-1348, Sept. 1989.

[69] D.T.M. Slock and T. Kailath, "Numerical stable fast RLS transversal adaptive filtering," *IEEE Trans. Acoust. Speech, Signal Processing*, ASSP-39, pp. 92- 114, Jan. 1991.

[70] A. Benallal and A. Gilloire, "Improvement of the tracking ability of the numerically stable fast RLS algorithms for adaptive filtering," *Proc. ICASSP-89*, pp. 1031-1034, 1989.

[71] A. Rontogiannis and S. Theodoridis, "New fast QR decomposition least squares algorithms," *IEEE Trans. Signal Processing*, vol. SP-46, pp. 2113-2121, Aug. 1998.

[72] A. Rontogiannis and S. Theodoridis, "On inverse factorization adaptive least squares algorithms," *Signal Processing*, vol. 52, pp. 35-47, 1996.

[73] A. Rontogiannis and S. Theodoridis, "Multichannel fast QRD-LS adaptive filtering: New technique and algorithms," *IEEE Trans. Signal Processing*, vol. SP-46, pp. 2862-2877, Nov. 1998.

[74] P.S.R. Diniz and M.G. Siqueira, "Fixed point error analysis of the QR-recursive least squares algorithm," *IEEE Trans. Circuits and Systems II, Analog and Digital Signal Processing*, vol. 42, pp. 334-348, May 1995.

[75] G.O. Glentis, "On the duality between the fast transversal and the fast QRD adaptive algorithms," *IEEE Trans. Signal Processing*, to appear 1999.

[76] G. Glentis and N. Kalouptsidis, "Fast adaptive algorithms for multichannel filtering and system identification," *IEEE Trans. Signal Processing*, vol. SP-40, pp. 2433- 2458, Oct. 1992.

[77] D. Slock, L. Chisci, H. Lev-Ari, and T. Kailath, "Modular and numerically stable fast transversal filters for multichannel and multiexperiment RLS," *IEEE Trans. Signal Processing*, vol. SP-40, pp. 784-802, April 1992.

[78] G. Cybenko, "The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations," *SIAM J. Sci. Statist. Comput.*, vol. 1, pp. 303-319, 1980.

[79] G. Glentis and N. Kalouptsidis, "Finite precision analysis of a covariance algorithm for LS FIR filtering and AR modeling," *IEEE Trans. Signal Processing*, vol. SP-41, pp. 2990-3002, Oct. 1993.

[80] R. Gitlin and F. Magee, "Self-orthogonalized adaptive equalization algorithms," *IEEE Trans. Comm.*, vol. COM-25, no. 7, pp. 666-672, July 1977.

[81] G. Panda, B. Mulgrew, C.F. Cowan, and P.M. Grant, "A self-orthogonalizing efficient block adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1573-1852, Dec. 1986.

[82] D. Marshal and W. Jenkins, "A fast quasi-Newton adaptive filtering algorithm," *IEEE Trans. Signal Proc.* vol. 40, no. 7, pp. 1652-1662, July 1992.

[83] P.S.R. Diniz, M. de Campos, and A. Antoniou, "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Trans. Signal Proc.*, vol. 43, no. 3, pp. 617-628, March 1995.

[84] M. de Campos and A. Antoniou, "A new Quasi-Newton adaptive filtering algorithm," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 924-934, Nov. 1997.

[85] G. Moustakides and S. Theodoridis, "Fast Newton transversal algorithms-A new class of adaptive estimation algorithms," *IEEE Trans. Signal Processing*, vol. 39, no. 10, pp. 2184-2193, Oct. 1991.

[86] T. Petillion, A. Gilloire, and S. Theodoridis, "The fast Newton transversal filter: An efficient scheme for acoustic echo cancellation in mobile radio," *IEEE Trans. Signal Processing*, vol. 42, no. 3, pp. 509-517, March 1994.

[87] S. Theodoridis, G. Moustakides, and K. Berberidis, "A fast Newton multichannel algorithm for decision feedback equalization," *IEEE Trans. Signal Processing*, vol. SP-43, pp. 327-331, Jan. 1995.

[88] P. Mavridis and G. Moustakides, "Simplified Newton-type adaptive estimation algorithms," *IEEE Trans. Signal Proc.*, vol. 44, no. 8, pp. 1932-1940, Aug, 1996.

[89] B. Farhang-Boroujeny, "Fast LMS/Newton algorithms based on autoregressive modeling and their application to acoustic echo cancellation," *IEEE Trans. Signal Proc.*, vol. 45, pp. 1987-2000, Aug. 1997.

[90] J.S. Lim and C.K. Un, "Optimum block adaptive filtering algorithms using preconditioning techniques," *IEEE Trans. Signal Processing*, vol. 45, pp. 773-778, March 1997.

[91] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electronics and Communications in Japan*, vol. 67-A, no. 5, pp. 19-27, 1984.

[92] M. Fukumoto and S. Tsujii, "New block adaptive algorithm using conjugate gradient method in noisy environment and its performance," *Electronics and Communications in Japan*, part 3, vol. 77, no. 9, pp. 1-11, 1994.

[93] K. Oishi, "A stable algorithm for block adaptive filtering," *Electronics and Communications in Japan*, part 3, vol. 76, no. 5, pp. 59-70, 1993.

[94] S. Gay, "A fast converging, low complexity adaptive filtering algorithm," 3rd Int. Workshop on Acoustic Echo Control, 1993, France.

[95] S. Gay and S. Tavathia, "The fast affine projection algorithm," *IEEE ICASSP*, pp. 3023-3027, 1995.

[96] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "Fast projection algorithm and its step size control," *IEEE ICASSP*, pp. 945-949, 1995.

[97] M. Tanaka, Y. Kaneda, S. Makino, and J. Kojima, "A fast projection algorithm for adaptive filtering," *IEICE Trans. Fundamentals*, vol. E78-A, no. 10m, Oct. 1995.

[98] M. Tanaka, S. Makino, and J. Kojima, "A block exact fast affine projection algorithm," *IEEE Trans. Speech and Audio Processing*, vol. 7, pp. 79-87, Jan. 1999.

[99] M. Montazeri and P. Duhamel, "A set of algorithms linking NLMS and block RLS algorithms," *IEEE Trans. Signal Processing*, vol. 43, no. 2, pp. 444-453, Feb. 1995.

[100] J. Benesty, P. Duhamel, and Y. Grenier, "A multichannel affine projection algorithm with application to multichannel acoustic echo cancellation," *IEEE Signal Processing Letters*, vol. 3, no. 2, pp. 35-38, Feb. 1996.

[101] D. Slock, "Underdetermined growing and sliding window covariance fast transversal filter RLS algorithms, *Signal Processing VI*, pp. 1169-1172, Brussels, 1992.

[102] D. Slock, "The block underdetermined covariance fast transversal filter algorithm for adaptive filtering," *Asilomar Conf. on Sig., Syst., and Comp.*, Pacific Grove, CA, 1992.

[103] D. Morgan and S. Kratzer, "A class of computationally efficient, rapidly converging, generalized NLMS algorithms," *IEEE Signal Processing Letters*, vol. 3, no. 8, pp. 245-247, Aug. 1996.

[104] M. Rupp, "A family of adaptive filter algorithms with properties," *IEEE Trans. Signal Processing*, vol. 46, no. 3, pp. 2771-774, March 1998.

[105] B. Bayakal and A.G. Constantinides, "Underdetermined order recursive least squares adaptive filtering: The concept and algorithms," *IEEE Trans. Signal Processing*, vol. 45, pp. 346-362, Feb. 1997.

[106] J. Benesty and P. Duhamel, "A fast exact LMS adaptive algorithm," *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 2904-2920, Dec. 1992.

[107] K. Berberidis and S. Theodoridis, "An efficient block Newton-type algorithm," *IEEE ICASSP*, pp. 1133-1137, 1995

[108] K. Berberidis and S. Theodoridis, "A new fast block adaptive algorithm," *IEEE Trans. Signal Processing*, vol. SP-47, pp. 75-87, Jan. 1999.

[109] P. Duhamel and M. Vetterli, "Improved Fourier and Hartley transform algorithms: Application to cyclic convolution of real data," *IEEE Trans. Acoust. Speech and Signal Processing*, vol. 35, pp. 818-824, June 1987.

[110] J. Soo and K. Pang, "A multistep size frequency domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-39, no. 1, pp. 115-121, Jan. 1991.

[111] M. Asharif and F. Amano, "Acoustic echo canceler using the FBAF algorithm, *IEEE Trans. Comm.*, vol. COM-42, no. 12, pp. 3090-3094, Dec. 1994.

[112] J. Prado and E. Moulines, "Frequency domain adaptive filtering with applications to acoustic echo cancellation," *Ann. Telecommun.*, vol. 49, no. 7-8, pp. 414-428, 1994.

[113] J. Borallo and M. Otero, "On the implementation of a partitioned block frequency domain adaptive filter for long acoustic echo cancellation," *Signal Processing*, 27, pp. 301-315, 1992.

[114] J. Soo and K. Pang, "Multidelay block frequency domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, no. 2, pp. 373-376, Feb. 1990.

[115] J. Lee and B. Lee, "Transform domain filtering based on structure," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-40, pp. 2061-2064, Aug. 1992.

[116] B. Farhang-Boroujeny and S. Gazor, "Generalized sliding FFT and its application to implementation of Block LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-42, no. 3, pp. 532-538, March 1994.

[117] E. Mulines, O. Amrane, and Y. Grenier, "The generalized multidelay adaptive filter: structure and convergence analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-43, no. 1, pp. 14-28, Jan. 1995.

[118] B. Farhang-Boroujeny, "Analysis and efficient implementation of partitioned block LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-44, no. 11, pp. 2865-2868, Nov. 1996.

[119] W. Chiang, and J. Liu, "Fast algorithm for FIR filtering in the transform domain," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-44, no. 11, pp. 126-129, Jan. 1996.

[120] C. Yon and K. Un, "Fast multidelay block transform domain adaptive filters based on a two-dimensional optimum block algorithm, *IEEE Trans. Circuits and Systems-II*, vol. CAS-II-41, no. 5, pp. 337-345, May 1994.

[121] X. Li and W.K. Jenkins, "The comparison of the constrained and unconstrained frequency domain block LMS adaptive algorithms, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-44, no. 7, pp. 1813-1816, July 1996.

[122] G. Egelmeers and P. Sommen, "A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering," *IEEE Trans. Signal Processing*, vol. 44, pp. 3123-3129, Dec. 1996.

[123] B. Petraglia, Y. Lee, and C.C. Ko, "Sliding transforms for efficient implementation of transform domain adaptive filters," *Signal Processing*, vol. 52, pp. 83-96, 1996.

[124] A. Gilloire and M. Vetterli, "Adaptive filtering in subbands with critical sampling: analysis, experiments, and application to acoustic echo cancelling," *IEEE Trans. Signal Processing*, pp. 1862-1875, Aug. 1992.

[125] M. Petraglia and S. Mitra, "Adaptive FIR filter structure based on the generalized subband decomposition of FIR filters," *IEEE Trans. Circuits and Systems II*, CAS-40, II, pp. 354-362, June 1993.

[126] A. Mahalanobis, S. Song, S. Mitra, and M. Petraglia, "Adaptive FIR filters based on structural subband decomposition for system identification problems," *IEEE Trans. Circuits and Systems II*, CAS-40, II, pp. 375-381, June 1993.

[127] F. Amano, H. Meana, A. de Luca, and G. Duchen, "A multirate acoustic echo canceler structure," *IEEE Trans. Comm.*, vol. 43, pp. 2172-2176, July 1995.

[128] D. Morgan and J. Thi, "A delayless subband adaptive filter architecture," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-43, no. 8, pp. 1819-1830, Aug. 1995.

[129] M. de Courville and P. Duhamel, "Adaptive filtering in subbands using a weighted criterion," *IEEE Trans. Signal Processing*, SP-46, no. 9, pp. 2359-2371, Sept. 1998.

[130] B. Farhang-Boroujeny and Z. Wang, "Adaptive filtering in subbands: Design issues and experimental results for acoustic echo cancellation," *Signal Processing*, vol. 61, pp. 213-223, 1997.

[131] F. Resende, P.S.R. Diniz, K. Tokuda, M. Kaneko, and A. Nishihara, "New adaptive algorithms based on multi-band decomposition of the error signal," *IEEE Trans. Circuits and Syst. II*, vol. 45, pp. 592-599, May 1998.

[132] G.A. Clark, S.K. Mitra, S.R. Parker, "Block adaptive filtering," *Proc. Of the IEEE Int. Symp. On Circuits and Systems, ISCAS-80*, pp. 384-387, Houston, TX, April 1980.

[133] D. Mansour, A.H. Gray, "Unconstrained frequency-domain adaptive filter," *IEEE Trans. Acous., Speech, and Signal Processing*, vol. 30, pp. 726-734, Oct. 1982.

[134] Z.J. Mou, D. Duhamel, "Fast FIR filtering: algorithms and implementation," *Signal Processing*, vol. 13, pp. 377-384, Dec. 1987.

[135] K. Berberidis and J. Palicot, "A block quasi-Newton algorithm implemented in the frequency domain," *Proc. IEEE ICASSP*, Atlanta, GA, pp. 1732-1735, May 1996.

[136] D.T.M. Slock and K. Maouche, "The fast subsampled-updating recursive least squares (FSU RLS) and fast trasversal filter (FSU FTF) algorithms for adapting long FIR filters," in *Proc. 3rd Int. Workshop on Acoustic Echo Control*, pp. 75-86, Plestin les Greves, France, Sept. 1993.

[137] D.T.M. Slock and K. Maouche, "The fast subsampled-updating stabilized fast transversal filter (FSU SFTF) RLS algorithm for adapting long FIR filters," in *Proc. VII European Signal Processing Conference*, pp. 740-743, Edinburg, Sept. 1994.

[138] D.T.M. Slock and K. Maouche, "The fast subsampled-updating recursive least squares (FSU RLS) algorithm for adaptive filtering based on displacement structure and the FFT," *Signal Processing*, vol. 40, number 2, pp. 5-20, 1994.

[139] S. Hossur, A.H. Tayfik, "Wavelet transform domain FIR filtering," *IEEE Trans. Signal Processing*, vol. 45, pp. 617-630, May 1997.

[140] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Acous., Speech, and Signal Processing*, vol. 34, pp. 285-295, April 1986.

[141] R. Kumar, "A fast algorithm for solving Toeplitz system of equations," *IEEE Trans. Acous., Speech, and Signal Processing*, vol. 33, pp. 254-267, Feb. 1985.

[142] C.H. Yon and C.K. Un, "Fast multidelay block transform domain adaptive filters based on 2D optimum block algorithm," *IEEE Trans. Circuits and Systems*, vol. 41, May 1994.

[143] T. Furukawa, H. Kubotta, S. Tsujii, "The orthogonal projection algorithm for block adaptive signal processing," *Proc. of the IEEE Int. Conf. On Acoustics, Speech, and Signal Processing*, ICASSP-89, pp. 1059-1062, Glasgow, May 1989.

[144] B. Friedlander, M. Morf, T. Kailath, L. Ljung, "New inversion for matrices in terms of their distance from Toeplitz matrices, *Linear Algebra and Its Applications*, vol. 27, pp. 31-60, 1979.

[145] K. Berberidis and J. Palicot, "A frequency domain quasi-Newton algorithm," *Signal Processing*, vol. 47, pp. 235-238, 1995.