14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 4-8, 2006, copyright by EURASIP

A FREQUENCY DOMAIN CONJUGATE GRADIENT ALGORITHM AND ITS APPLICATION TO CHANNEL EQUALIZATION

Aris S. Lalos, Kostas Berberidis

Dept of Computer Engineering & Informatics, and RACTI R&D
University of Patras, 26500 Rio-Patras, Greece
E-mails: lalos@ceid.upatras.gr, berberid@ceid.upatras.gr

ABSTRACT

In this paper, a new block adaptive filtering algorithm, based on the Conjugate Gradient (CG) method of optimization, is proposed. A Toeplitz approximation of the autocorrelation matrix is used for the estimation of the gradient vector and the correlation quantities are updated on a block by block basis. Due to this formulation, the algorithm can be implemented in the frequency domain (FD) using the fast Fourier transform (FFT). Efficient recursive relations for the frequency domain quantities updated on a block by block basis have been derived and an appropriate decoupling of the direction vector has been applied. The applicability of the new algorithm to the problem of adaptive equalization is studied. The proposed algorithm exhibits superior convergence properties as compared to existing CG techniques, offering significant savings in computation complexity.

1. INTRODUCTION

Adaptive filtering algorithms has been an area of active research over the last decades due to their wide applicability in many signal processing and communication applications. The performance of an adaptive algorithm can be measured by a number of factors such as accuracy of steady state solution, convergence speed, tracking abilities, computational complexity, numerical robustness, etc [1]. In many real time applications the issues of complexity and convergence speed play a crucial role, therefore many different techniques such as partial updating schemes [2], IIR adaptive filtering [3], and Frequency Domain Adaptive Filtering (e.g [1], [4]) have been employed to reduce the computational complexity, and to accelerate the convergence.

The aim in partial updating schemes, is to reduce the computational complexity of an adaptive filter by adapting a block of the filter coefficients rather than the entire filter at every iteration. In non stationary environments, partial updating adaptive filters might be undesirable as they do not guarantee convergence. The use of IIR adaptive filters dramatically reduces the computational complexity since a good performance can be achieved by estimating a small number of parameters. However, the numerical behavior of IIR schemes in certain implementation platforms is still an issue under investigation. Frequency domain adaptive filters (FDAF) turn out to be a good solution in several practical applications due to their computational efficiency and their good convergence properties.

Most of the existing FDAF algorithms are of the gradient type, that is, their time-domain counterparts are based on some variations of the Least Mean Square (LMS) algorithm. On the contrary, to the best of our knowledge, no work has been done toward developing frequency domain implementations of adaptive algorithms based on the Conjugate Gradient (CG) method. The family of algorithms that are based on the CG method of optimization provides convergence comparable to RLS schemes at a computational complexity that is intermediate between the LMS and the RLS methods. In existing literature, depending upon the CG algorithm under consideration the cost of sequential processing of the data grows subexponentially or quadratically with the filter order.

In this paper, a new block adaptive CG algorithm implemented in the frequency domain is developed. The complexity of the algorithm is considerably reduced by employing the FFT algorithm, to update the Toeplitz approximation of the autocorrelation matrix and the cross correlation vector via fast correlation implementation, and to calculate matrix vector products. Furthermore, the convergence of the algorithm is remarkably improved as compared to existing CG adaptive algorithms [5], by appropriate decoupling the successive direction vectors. The application of the algorithm to Channel Equalization is also pointed out. The resulting algorithm exhibits similar convergence with the RLS algorithm and a computational complexity that is proportional to the logarithm of the filter order per time step.

The paper is organized as follows. In Section 2, the problem is formulated, the conjugate gradient method of optimization is described and ways to implement the algorithm efficiently in adaptive filtering context are also reported. In Section 3, the new block CG algorithm and its frequency domain implementation are derived. In Section 4 the adaptive channel equalization case is treated. Finally simulation results are provided in Section 5, and the work is concluded in Section 6.

2. SOME PRELIMINARIES

2.1 Problem Formulation

Before proceeding further, let us first define the notation used throughout the paper. In the time domain, vectors and matrices are denoted by bold lower case and bold upper case letters, respectively. In the frequency domain, vectors are denoted by calligraphic upper case letters.

Let us now assume that we are given the input $\{x(n)\}$ and a desired output $\{u(n)\}$ of an unknown system. The unknown system is assumed to be a time varying linear system and the task is to obtain at each time n an estimate $\mathbf{w}(n) = [w_1(n), \cdots, w_M(n)]^T$ of the system. This estimate is computed so that its output $y(n)$ given by

$$y(n) = \sum_{i=1}^{M} w_i(n)\, x(n-i)$$

tracks in an optimal way the desired output $u(n)$ of the unknown system.

2.2 The Conjugate Gradient Algorithm

The Conjugate Gradient (CG) Method is an iterative method for finding the minimum of a quadratic cost function. The objective of the method here, is the minimization of a cost function defined as

$$V(\mathbf{w}) = E\left[|u(n)|^2\right] - \mathbf{b}^H\mathbf{w} - \mathbf{w}^H\mathbf{b} + \mathbf{w}^H\mathbf{R}\mathbf{w} \qquad (1)$$

where $\mathbf{R} = E\left[\mathbf{x}^*(n)\,\mathbf{x}^T(n)\right]$ is the $M \times M$ correlation matrix of the system input $x(n)$, and $\mathbf{b} = E\left[u(n)\,\mathbf{x}^*(n)\right]$ is the cross-correlation vector between the desired output $u(n)$ of the system and the input $x(n)$. Vector $\mathbf{x}(n)$ is defined as $[x(n), \cdots, x(n-M+1)]^T$.

This work was partially supported by the General Secretariat for Research and Technology, under grant ΠΕΝΕΔ-No.03ΕΔ838.

INITIAL CONDITIONS:
$\mathbf{w}(0)=0, \mathbf{g}(0)=\mathbf{b}, \mathbf{p}(1)=\mathbf{g}(0), k=1$
WHILE $k \leq k_{max}$

$$a(k)=\frac{\mathbf{p}^H(k)\mathbf{g}(k-1)}{\mathbf{p}^H(k)\mathbf{R}\mathbf{p}(k)} \quad (2)$$

$$\mathbf{w}(k)=\mathbf{w}(k-1)+a(k)\mathbf{p}(k) \quad (3)$$

$$\mathbf{g}(k)=\mathbf{g}(k-1)-a(k)\mathbf{R}\mathbf{p}(k) \quad (4)$$

$$\beta(k)=\frac{\mathbf{g}^H(k)\mathbf{g}(k)}{\mathbf{g}^H(k-1)\mathbf{g}(k-1)} \quad (5)$$

$$\mathbf{p}(k+1)=\mathbf{g}(k)+\beta(k)\mathbf{p}(k) \quad (6)$$

$k=k+1$

END

Table 1: Basic CG Algorithm

The CG method originates from the so-called conjugate direction (CD) method [6]. The main idea in the CD method is to obtain a set of linearly independent direction vectors which are conjugate with respect to $\mathbf{R}$ so that the vector $\mathbf{w}^o$ that minimizes (1) can be expressed as a linear combination of these vectors. The weights of the linear combination are calculated by imposing $\mathbf{R}$-conjugation between the direction vectors. The set of the $\mathbf{R}$-conjugate direction vectors are generated by a set of M linearly independent vectors. In the CG method these M vectors are the successive gradients of the cost function (1), obtained as the method progresses.

The basic CG algorithm consists of equations (2)-(6) summarized in Table 1, where $a(k)$ is the step size that minimizes the cost function $V(\mathbf{w}(k))$ defined similarly as in (1), $\beta(k)$ provides $\mathbf{R}$-orthogonality for the direction vector $\mathbf{p}(k)$, and $\mathbf{g}(k)$ is the negative of the gradient vector of the cost function at $\mathbf{w}(k)$ defined as

$$\mathbf{g}(k)=\mathbf{b}-\mathbf{R}\mathbf{w}(k)=-\frac{1}{2}\left[\nabla_w V(\mathbf{w}(k))\right]. \quad (7)$$

Previously developed CG adaptive algorithms execute several iterations of the CG algorithm of Table 1 to solve the system of equations $\mathbf{R}\mathbf{w}=\mathbf{b}$, per sample or per block update of $\mathbf{R},\mathbf{b}$ [7]. In the case of sample by sample update of $\mathbf{R}$, $\mathbf{b}$, some modifications have been proposed in order to allow the CG algorithm to run one iteration per incoming sample, but still maintain performance comparable with RLS or LMS-Newton [5]. However, the matrix-vector multiplication $\mathbf{R}\mathbf{p}(k)$ that is required in each iteration increases the total complexity to $O\left(M^2\right)$ multiplications per output sample. Therefore, the way that $\mathbf{R}$ and $\mathbf{b}$ are estimated directly influences the performance and the complexity of the algorithm.

In the section that follows, some modifications that allow the algorithm to run just one iteration per incoming block of input data are proposed. $\mathbf{R}$ is approximated by a Toeplitz matrix constructed by a time averaged estimator of the autocorrelation sequence $r_0, r_1, \cdots, r_{M-1}$ of $\{x\}$ i.e. $r_k(n)=\sum_{i=k+1}^{n}\lambda^{n-i}x^*(i)x(i-k)$. Thus, since the matrix vector multiplications can be computed by circular convolutions, the FFT is used throughout the computations and the total complexity is reduced to $O(M\log M)$ multiplications per output block of data samples. Moreover, the convergence speed of the resulting Frequency Domain (FD) algorithm is considerably increased by properly decoupling the successive direction vectors.

3. BLOCK FREQUENCY DOMAIN CG ALGORITHM

3.1 Block CG Algorithm Derivation

In this section, the main task is to derive the block updating equation of the weight coefficient vector, based on the basic CG algorithm. Initially, it is essential to focus on obtaining the block update recursion of the autocorrelation and the cross correlation vectors by using the autocorrelation method of data windowing [1]. Let us first write the symbol-by-symbol update recursion of the vectors $\mathbf{r}(n)$, $\mathbf{b}(n)$ as follows:

$$\mathbf{r}(n)=\lambda\mathbf{r}(n-1)+x^*(n)\mathbf{x}(n)$$
$$\mathbf{b}(n)=\lambda\mathbf{b}(n-1)+u(n)\mathbf{x}^*(n)$$

where $\{x(n)\}$, $\{u(n)\}$ denote the system's input and desired output, respectively. Based on the above recursions, it is straightforward to show that

$$\mathbf{r}(n)=\lambda^L\mathbf{r}(n-L)+\sum_{i=0}^{L-1}\lambda^i x^*(n-i)\mathbf{x}(n-i) \quad (8)$$

$$\mathbf{b}(n)=\lambda^L\mathbf{b}(n-L)+\sum_{i=0}^{L-1}\lambda^i u(n-i)\mathbf{x}^*(n-i) \quad (9)$$

This expression represents a single update of the auto - correlation and cross-correlation vectors from time $n-L+1$ to time n based on the L data samples accumulated, and is thus called a block update. For simplicity and efficiency we are interested in the case of $L=M$. Without loss of generality we can substitute $n=kL$ in (8), (9), where k is a block time index. By factoring the argument kL on the left hand side of (8), (9) and dropping the explicit dependence of correlation vectors on L we have the following equivalent block updates:

$$\mathbf{r}(k)=\lambda^M\mathbf{r}(k-1)+\mathbf{r}_g(k) \quad (10)$$

$$\mathbf{b}(k)=\lambda^M\mathbf{b}(k-1)+\mathbf{b}_g(k) \quad (11)$$

where

$$\mathbf{r}_g(k)=\sum_{i=0}^{M-1}\lambda^i x^*(kM-i)\mathbf{x}(kM-i) \quad (12)$$

$$\mathbf{b}_g(k)=\sum_{i=0}^{M-1}\lambda^i u(kM-i)\mathbf{x}^*(kM-i) \quad (13)$$

To finish up with the correlations update equations, we recall that matrix $\mathbf{R}(k)$ is a Toeplitz symmetric matrix constructed by the elements of vector $\mathbf{r}(k)$, and $T\left(\mathbf{r}_g(k)\right)$ is a Toeplitz symmetric matrix, constructed by the elements of vector $\mathbf{r}_g(k)$.

Let us now assume that vectors $\{\mathbf{r}(k),\mathbf{b}(k)\}$ are fixed from block to block. Then a block recursive formula of vector $\mathbf{g}(k)$ can be found by using (3), (7), (10) and (11) resulting in

$$\begin{aligned}\mathbf{g}(k)&=\mathbf{b}(k)-\mathbf{R}(k)\mathbf{w}(k)=\lambda^M\mathbf{g}(k-1)+\mathbf{b}_g(k)\\&\quad-a(k)\mathbf{R}(k)\mathbf{p}(k)-T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)\\&=\mathbf{g}'(k-1)-a(k)\mathbf{R}(k)\mathbf{p}(k)\end{aligned} \quad (14)$$

where $\mathbf{p}(k)$ is the direction vector related with vector $\mathbf{g}(k)$ via eq. (6), and

$$\begin{aligned}\mathbf{g}'(k-1)&=\lambda^M\mathbf{g}(k-1)+\mathbf{b}_g(k)\\&\quad-T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1).\end{aligned}$$

In the basic CG algorithm, $a(k)$ is a step size used in the update of the weight vector as shown in (3). It is usually chosen so that $V(\mathbf{w}(k-1)+a(k)\mathbf{p}(k))$ is minimized. In other words, $a(k)$ can be computed by setting the gradient of the cost function with respect to $a(k)$ equal to zero:

$$\nabla_a V(\mathbf{w}(k))=0\Rightarrow\mathbf{p}^H(k)\mathbf{g}(k)=0\Rightarrow$$
$$a(k)=\frac{\mathbf{p}^H(k)\mathbf{g}'(k-1)}{\mathbf{p}^H(k)\mathbf{R}(k)\mathbf{p}(k)}$$

Finally, the factor $\beta(k)$ that provides $\mathbf{R}$-orthogonality between the vectors $\mathbf{p}(k)$, cannot be obtained by directly applying equation (5), since a non-constant $\mathbf{R}$ is used at each iteration. One way to tackle this problem is to periodically reset the direction vector to the negative of the gradient vector $\mathbf{p}(k)=\mathbf{g}(k)$ in order to ensure the convergence of the algorithm [5]. Alternatively, the Polak-Ribiere method can also be used [6]. In such a case, $\beta(k)$ can be computed by

$$\beta(k)=\frac{(\mathbf{g}(k)-\mathbf{g}(k-1))^H\mathbf{g}(k)}{\mathbf{g}^H(k)\mathbf{g}(k)}.$$

Having computed $\beta(k)$, the direction vector $\mathbf{p}(k)$ and the weight coefficient vector are updated by using (6) and (3), respectively.

3.2 Frequency Domain Implementation

The block update terms in the right side of the equations (10), (11) are linear correlations. Specifically, the quantity in (12) is a linear correlation between the input signal and the input signal vector and the quantity in (13) is a linear correlation between the desired signal and the input signal vector. Therefore, it is possible to efficiently implement each of these sums in the frequency domain by using the overlap-save sectioning method[4].

If we define the following frequency domain quantities

$$\mathscr{X}(k)=\mathbf{F}\left[0,\cdots,0,\lambda^{M-1}x(kM-M+1),\cdots,x(kM)\right]^T \quad (15)$$

$$\mathscr{U}(k)=\mathbf{F}\left[0,\cdots,0,\lambda^{M-1}u(kM-M+1),\cdots,u(kM)\right]^T \quad (16)$$

$$\mathscr{V}(k)=diag\{\mathbf{F}[x(kM-2M+1),\cdots,x(kM-M),\ x(kM-M+1),\cdots,x(kM)]^T\} \quad (17)$$

where $\mathbf{F}$ is the DFT matrix of order $2M$, then we may write:

$$\begin{bmatrix}\mathbf{r}_g^*(k)\\ \times\end{bmatrix}=\mathbf{F}^{-1}\mathscr{V}^H(k)\mathscr{X}(k) \quad (18)$$

$$\begin{bmatrix}\mathbf{b}_g(k)\\ \times\end{bmatrix}=\mathbf{F}^{-1}\mathscr{V}^H(k)\mathscr{D}(k) \quad (19)$$

Symbol $\times$ denotes a "don't care" $M\times 1$ vector.

It should also be noticed that matrices $\mathbf{R}(k)$ and $T\left(\mathbf{r}_g(k)\right)$ are Toeplitz. Thus we may implement the products $\mathbf{R}(k)\mathbf{p}(k)$, $T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)$ efficiently in the frequency domain by embedding $\mathbf{R}(k)$, $T\left(\mathbf{r}_g(k)\right)$ into $2M\times 2M$ circulant matrices

$$\mathbf{C}_R(k)=\begin{bmatrix}\mathbf{R}(k)&\mathbf{S}(k)\\ \mathbf{S}(k)&\mathbf{R}(k)\end{bmatrix},\ \mathbf{C}_T(k)=\begin{bmatrix}T\left(\mathbf{r}_g(k)\right)&\mathbf{S}'(k)\\ \mathbf{S}'(k)&T\left(\mathbf{r}_g(k)\right)\end{bmatrix}$$

where it is sufficient here to define the first column of $\mathbf{C}_R(k)$ as $\mathbf{c}(k)=\left[\mathbf{r}(k);0;\mathbf{r}^*_{[M:-1:1]}(k)\right]$ and the first column of $\mathbf{C}_T(k)$ as $\mathbf{c}_T(k)=\left[\mathbf{r}_g(k);0;\mathbf{r}^*_{g[M:-1:1]}(k)\right]$ with the help of some Matlab notation. Vectors $\mathbf{r}^*_{[M:-1:1]}(k)$, $\mathbf{r}^*_{g[M:-1:1]}(k)$ consist of the $M-1$ last elements of $\mathbf{r}^*(k)$, $\mathbf{r}^*_g(k)$ respectively, in reverse order. The matrices $\mathbf{C}_R(k)$, $\mathbf{C}_T(k)$ can be diagonalized by using the DFT matrix, and their spectral decomposition is given by

$$\mathbf{C}_R(k)=\mathbf{F}^{-1}\mathbf{D}_R(k)\mathbf{F},\ \ \mathbf{C}_T(k)=\mathbf{F}^{-1}\mathbf{D}_T(k)\mathbf{F}$$

where $\mathbf{D}_R(k)$,$\mathbf{D}_T(k)$ are $2M\times 2M$ diagonal matrices containing the eigenvalues of $\mathbf{C}_R(k)$, $\mathbf{C}_T(k)$ respectively. The eigenvalues of $\mathbf{C}_T(k)$ are equal to the DFT values of the first column of $\mathbf{C}_T(k)$, i.e., $\mathbf{D}_T(k)=diag\{\mathbf{F}\mathbf{c}_T(k)\}$. The eigenvalues of $\mathbf{C}_R(k)$ may be computed in terms of the eigenvalues of $\mathbf{C}_T(k)$. Specifically, based on (10), the following recursion can be easily derived

$$\mathbf{D}_R(k)=\lambda^M\mathbf{D}_R(k-1)+\mathbf{D}_T(k). \quad (20)$$

The matrix vector product $\mathbf{R}(k)\mathbf{p}(k)$ can be computed by the following equation

$$\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}=\begin{bmatrix}\mathbf{I}_M&\mathbf{0}_M\\ \mathbf{0}_M&\mathbf{0}_M\end{bmatrix}\mathbf{F}^{-1}\mathbf{D}_R\mathbf{F}\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}.$$

The product $T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)$ can be expressed in a similar way.

Thus we may augment the coefficient update (3), the update of the vector $\mathbf{g}(k)$ (14) and the direction update (6) schemes as

$$\begin{bmatrix}\mathbf{w}(k)\\ \mathbf{0}_M\end{bmatrix}=\begin{bmatrix}\mathbf{w}(k-1)\\ \mathbf{0}_M\end{bmatrix}+a(k)\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{g}(k)\\ \mathbf{0}_M\end{bmatrix}=\lambda^M\begin{bmatrix}\mathbf{g}(k-1)\\ \mathbf{0}_M\end{bmatrix}-a(k)\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}+\begin{bmatrix}\mathbf{b}_g(k)\\ \mathbf{0}_M\end{bmatrix}-\begin{bmatrix}T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)\\ \mathbf{0}_M\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{p}(k+1)\\ \mathbf{0}_M\end{bmatrix}=\begin{bmatrix}\mathbf{g}(k)\\ \mathbf{0}_M\end{bmatrix}+\beta(k)\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}$$

Taking the DFT of both sides of the above equations we have

$$\mathscr{W}(k)=\mathscr{W}(k-1)+a(k)\mathscr{P}(k) \quad (21)$$

$$\mathscr{G}(k)=\lambda^M\mathscr{G}(k-1)-a(k)\mathbf{F}\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}+\mathbf{F}\mathbf{G}\begin{bmatrix}\mathbf{b}_g(k)\\ \times\end{bmatrix}-\mathbf{F}\begin{bmatrix}T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)\\ \mathbf{0}_M\end{bmatrix} \quad (22)$$

$$\mathscr{P}(k+1)=\mathscr{G}(k)+\beta(k)\mathscr{P}(k) \quad (23)$$

where matrix $\mathbf{G}$ is a time domain constraint matrix defined as $\mathbf{G}=[\mathbf{I}_M,\mathbf{0}_M]$, with $\mathbf{0}_M$ being an $M\times M$ matrix of zeros.

To finish up with the update equations we observe that the time domain step-size $a(k)$ and the time domain factor $\beta(k)$ can be computed by already available frequency domain vectors as

$$a(k)=\frac{\mathbf{p}^H(k)\mathbf{g}'(k-1)}{\mathbf{p}^H(k)\mathbf{R}(k)\mathbf{p}(k)}=\frac{\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}^H\begin{bmatrix}\mathbf{g}'(k-1)\\ \mathbf{0}_M\end{bmatrix}}{\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}^H\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}}$$

$$=\frac{\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}^H\frac{\mathbf{F}^H\mathbf{F}}{2M}\begin{bmatrix}\mathbf{g}'(k-1)\\ \mathbf{0}_M\end{bmatrix}}{\begin{bmatrix}\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}^H\frac{\mathbf{F}^H\mathbf{F}}{2M}\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}}=\frac{\mathscr{P}^H(k)\mathscr{G}'(k-1)}{\mathscr{P}^H(k)\mathbf{F}\begin{bmatrix}\mathbf{R}(k)\mathbf{p}(k)\\ \mathbf{0}_M\end{bmatrix}} \quad (24)$$

where

$$\mathscr{G}'(k-1)=\lambda^M\mathscr{G}(k-1)+\mathbf{F}\mathbf{G}\begin{bmatrix}\mathbf{b}_g(k)\\ \times\end{bmatrix}-\mathbf{F}\begin{bmatrix}T\left(\mathbf{r}_g(k)\right)\mathbf{w}(k-1)\\ \mathbf{0}_M\end{bmatrix}$$

and

$$\beta(k)=\frac{(\mathscr{G}(k)-\mathscr{G}(k-1))^H\mathscr{G}(k)}{\mathscr{G}^H(k-1)\mathscr{G}(k-1)} \quad (25)$$

Convergence acceleration

The convergence speed of the whole scheme might be improved by using in update equations (21), (22), and (23) matrix step sizes instead of scalar ones $a(k)$ and $\beta(k)$, respectively. These matrices, denoted as $\mathbf{M}_a(k)$ and $\mathbf{M}_\beta(k)$, would have a diagonal structure, thus allowing a decoupled update at the different frequency bins. The diagonals of those time varying matrices contain, the step sizes $a_m(k)=a(k)/P_m(k)$ and the factors $\beta_m(k)=\beta(k)/P_m(k)$, $m=0,\cdots 2M-1$, respectively. $P_m(k)$ corresponds to an estimate of the signal power in the m^{th} bin, and may be computed as in [8]

$$P_m(k)=\lambda_p P_m(k-1)+\left(1-\lambda_p\right)|\mathscr{V}_{m,m}(k)|^2$$

INITIALIZATION:

$P_m(0) = \delta_m \mathbf{I}$, $\mathscr{W}(0) = [0,0,\cdots,0]^T$, $\mathscr{P}(1) = \mathscr{G}(0)$

MATRIX DEFINITIONS:

$G = \begin{bmatrix} \mathbf{I}_M & \mathbf{0}_M \end{bmatrix}$, $\tilde{G} = \begin{bmatrix} \mathbf{I}_M & \mathbf{0}_M \\ \mathbf{0}_M & \mathbf{0}_M \end{bmatrix}$

$F = 2M \times 2M$ **DFT matrix**

FOR EACH NEW BLOCK k:

$P_m(k) = \lambda_p P_m(k-1) + (1-\lambda_p)\,|\mathscr{V}_{m,m}(k)|^2$

$M_p = diag\left\{\left[P_0^{-1}(k),\cdots,P_{2M-1}^{-1}(k)\right]\right\}$

$\mathscr{P}(k) = M_p \mathscr{P}(k)$

$\mathscr{X}(k) = \mathbf{F}\left[0,\cdots,0,\lambda^{M-1}x(kM-M+1),\cdots,x(kM)\right]^T$

$\mathscr{U}(k) = \mathbf{F}\left[0,\cdots,0,\lambda^{M-1}u(kM-M+1),\cdots,u(kM)\right]^T$

$\mathscr{V}(k) = diag\left\{\mathbf{F}\left[x(kM-2M+1),\cdots,x(kM)\right]^T\right\}$

$\mathbf{r}_g(k) = G\mathbf{F}^{-1}\mathscr{V}^H(k)\,\mathscr{X}(k)$

$\mathbf{c}_T(k) = \left[\mathbf{r}_g^T(k), 0, \mathbf{r}_{g[M:-1:1]}^H(k)\right]^H$

$\mathbf{D}_T(k) = diag\{\mathbf{F}\mathbf{c}_T(k)\}$

$\mathbf{D}_R(k) = \lambda^M \mathbf{D}_R(k-1) + \mathbf{D}_T(k)$

$\mathscr{J}_1 = \mathbf{F}\tilde{G}\mathbf{F}\mathbf{D}_R\mathbf{F}^{-1}\tilde{G}\mathbf{F}^{-1}\mathscr{P}(k)$

$\mathscr{J}_2 = \mathbf{F}\tilde{G}\mathbf{F}\mathbf{D}_T\mathbf{F}^{-1}\tilde{G}\mathbf{F}^{-1}\mathscr{W}(k-1)$

$\mathscr{G}'(k-1) = \lambda^M\mathscr{G}(k-1) + \mathbf{F}G\mathbf{F}^{-1}\mathscr{V}^H(k)\,\mathscr{D}(k) - \mathscr{J}_2$

$a(k) = \frac{\mathscr{P}^H(k)\mathscr{G}'(k-1)}{\mathscr{P}^H(k)\mathscr{J}_1}$

$\mathscr{W}(k) = \mathscr{W}(k-1) + a(k)\,\mathscr{P}(k)$

$\mathscr{G}(k) = \mathscr{G}'(k-1) - a(k)\,\mathscr{J}_1$

$\beta(k) = \frac{(\mathscr{G}(k)-\mathscr{G}(k-1))^H\mathscr{G}(k)}{\mathscr{G}^H(k-1)\mathscr{G}(k-1)}$

$\mathscr{P}(k+1) = [\mathscr{G}(k) + \beta(k)\,\mathscr{P}(k)]$

$k = k+1$

END

Table 2: FD-CG Algorithm

It is easily seen that using $\mathbf{M}_a(k)$ and $\mathbf{M}_\beta(k)$ is equivalent with the premultiplication of each direction vector $\mathscr{P}(k)$ with a diagonal matrix $\mathbf{M}_p$ whose elements are given by:

$$\mathbf{M}_p = diag\left\{\left[P_0^{-1}(k),\cdots,P_{2M-1}^{-1}(k)\right]\right\}.$$

The latter step sizing approach is the one used in the algorithm summarized in Table 2.

3.3 Complexity Issues

The computational complexity (real multiplications) of the algorithm of Table 2 is now summarized for comparison with the non block time domain modified CG [5] and the classical Frequency Domain Adaptive Filtering (FDAF) algorithm based on the overlap save sectioning method [4].

The modified CG with M complex weights requires $16M^2$ real multiplications for the computation of the matrix-vector product $\mathbf{R}(k)\mathbf{p}(k)$, $16M$ real multiplications to compute the step-size and the factor $\beta(k)$ and another $22M$ real multiplications to update the coefficient, the gradient and the direction vector. Thus, $16M^3 + 38M^2$ are required for every M output samples. The linear convolution overlap-save FDAF requires $10M\log_2(2M) + 24M$ real multiplications per M output samples.

Complexity Ratios	Filiter Size M			
	64	128	256	1024
$\frac{FDCG}{Modified\ CG}$	0.0037	0.0010	0.0003	2e-5
$\frac{FDCG}{Overlap-save\ FDAF}$	3.4964	3.4531	3.4167	3.3591

Table 3: Computational Complexity Ratios

The proposed algorithm requires $30M\log_2(2M)$ real multiplications for the computation of 15 $2M$-point FFT's. The computation of the partial products $\mathscr{J}_1$, $\mathscr{J}_2$ requires $16M$ real additional multiplications. Furthermore, another $24M$ additional multiplications are required for the computation of (15), (16), (18), (19), (20) and finally $68M$ multiplications are required to calculate the step-size, the factor that provides $\mathbf{R}$-conjugacy and to update the coefficient, the gradient and the direction vector. Therefore, the proposed algorithm requires $30M\log_2(2M) + 108M$ real multiplications for every M output samples. The complexity ratios are summarized in Table 3. Clearly, the complexity of the algorithm of Table 2 is of the same order as the overlap-save FDAF, since the complexity ratio of the two algorithms remains the same as the filter length increases. On the other hand, the proposed FDCG algorithm offers significant savings as compared to Modified CG.

4. APPLICATION TO CHANNEL EQUALIZATION

In this section, the applicability of the new algorithm to adaptive channel equalization is pointed out. We are particularly interested in mobile wireless communication systems. Channels that are encountered in such systems may have long Impulse Responses (IR) and change significantly in time. Therefore the involved equalizer should be able to track the channel variations and also have a fast convergence so as a reduced training sequence to be adequate.

4.1 Linear Equalization Case

The main part of a linear equalizer is a transversal filter that combines linearly a number of consecutive channel output samples to provide an estimate of the current symbol. The algorithm of Table 2 is directly applicable to the LE case after taking into account the following remarks:

- The equalizer's coefficients correspond to the unknown parameters of the system $w_i(k)$ that has to be identified.
- Sample $x(kM)$ denotes the channel output at time kM, and sample $u(kM)$ is either a training symbol (during the training mode), or a decision (during decision directed mode). In the latter case, $u(kM)$ will be given by

$$u(kM) = f\left\{\sum_{i=1}^{M} w_i(k)\,x(kM+M_1-i)\right\},$$

where $f\{\cdot\}$ stands for the decision device function and M_1-1 stands for the number of samples that the filter output is delayed with respect to its input in order to provide the desired response to the equalizer.

Thus, the algorithm of Table 2, can be directly applied to the linear equalization case, provided that we replace $\mathscr{X}(k)$ with the vector $\mathscr{X}(k+M_1-1)$ and $\mathscr{V}(k)$ with the vector $\mathscr{V}(k+M_1-1)$.

4.2 DFE Case

The algorithm of Table 2 might also be applied to the adaptive decision feedback equalization problem. The output $y(n)$ of a step by step DFE equalizer at time n in a vector form is given by

$$\mathbf{y}(n) = \mathbf{w}(n)\,\mathbf{x}'(n+M-1)$$

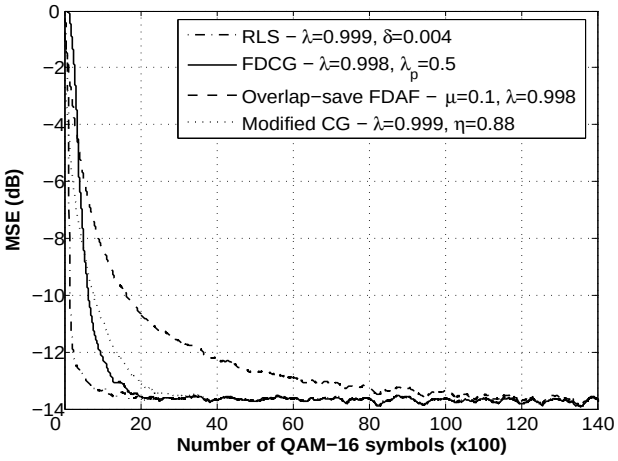

Figure 1: Channel A - LE(128,64)

where

$$\mathbf{w}(n) = \begin{bmatrix} \mathbf{w}_{ff}(n) \\ \mathbf{w}_{fb}(n) \end{bmatrix}, \quad \mathbf{x}'(n) = \begin{bmatrix} \mathbf{x}(n+M-1) \\ \mathbf{u}(n-1) \end{bmatrix}.$$

Let us now denote the DFE equation by using the orthogonal principle:

$$\mathbf{R}^{x'}\mathbf{w} = \mathbf{b}', \quad \mathbf{R}^{x'} = \begin{bmatrix} \mathbf{R}^{x} & (\mathbf{R}^{ux})^{H} \\ \mathbf{R}^{ux} & \mathbf{R}^{u} \end{bmatrix}$$

As we can see, the input autocorrelation matrix $\mathbf{R}^{x'}$ is no longer Toeplitz as in the linear equalization problem. However, by approximating the cross-terms with zero matrices the FF and the FB filters are decoupled and can be treated separately. Thus, the vectors $\mathbf{w}_{ff}(n)$ and $\mathbf{w}_{fb}(n)$ can be updated on a block by block basis by applying a CG scheme as the one described for the LE case.

Unfortunately, there is an inherent "causality" problem in the above block formulation. Some unknown decisions need to be used for the update of the FB filter coefficients. This problem can be overcome by using tentative decisions in place of the unknown ones within each block [9].

5. SIMULATION RESULTS

To illustrate the performance of the algorithm we provide some simulation results. The experiments were carried out on two different wireless channels, named as channel A and channel B [10]. Channel A contained 6 multipath components with amplitudes 0.7490, 1, 0.2290, 0.3160, 0.0550, 0.1580 respectively, and the corresponding time delays with respect to the main peak were T_s, $2T_s$, $25T_s$, $37T_s$, $28T_s$, $57T_s$, respectively. The multipath component phases were chosen randomly. Channel B, contained another 4 components with amplitudes 0.9333, 0.5012, 0.5129, 0.5370 and time delays T_s, $8T_s$, $15T_s$, $22T_s$ respectively. The multipath component phases again were chosen randomly. The input sequence consisted of QAM-16 symbols. Its equalizer's length was set equal to 128 and the filter output is delayed by 64 samples. In the tests we conducted the equalizers operate in training mode. In the first experiment, the equalizer's coefficient vector was set initially equal to the zero vector for all the algorithms. In figure 1, we provide the ensemble averaging for the learning curve over 100 independent trials of the experiment for the FDCG algorithm, the overlap-save FDAF, the Modified-CG and the RLS algorithm. In the second experiment, in order to test the ability of the algorithm to forget old data and hence track, the channel was suddenly changed at time instant 5000. Figure 2 shows the performance of the algorithms when the parameters of the unknown system change abruptly. It can be seen that in both cases (convergence, tracking) the FDCG algorithm exhibits superior convergence rate as compared to the Modified CG algorithm [5] and to the overlap-save FDAF algorithm [4].

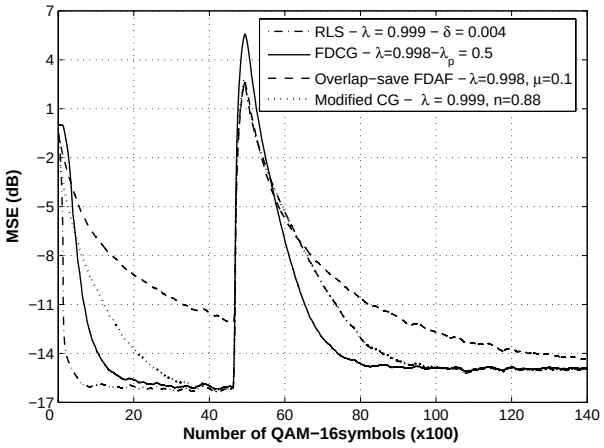

Figure 2: Channel B - Channel A - LE(128,64)

6. CONCLUSION

A new block adaptive CG algorithm implemented in the frequency domain is developed. The correlation quantities are updated by employing the overlap-save sectioning method. Due to the Toeplitz approximation of the autocorrelation matrix, the FFT algorithm is employed for the computation of the matrix vector products that arise. The algorithm enjoys superior convergence properties of existing adaptive algorithms based on the CG method. Its computational complexity is proportional to the logarithm of the filter order. The performance is further enhanced due to frequency domain implementation and the decoupling of the direction vector. A suitable signal power initialization/update, the applicability of the new algorithm to the DFE case and its tracking behaviour in Rayleigh channels are some of the issues that are currently being investigated.

REFERENCES

[1] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NY: Wiley, 2003.

[2] S. C. Douglas, "Adaptive filters employing partial updates," *IEEE Trans. Circuits Syst. II*, vol. 44, no. 3, pp. 209–216, Mar. 1997.

[3] P. A. Regalia, *Adaptive IIR Filtering in Signal Processing and Control.* New York: Marcel Dekker, 1995.

[4] J. J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Mag.*, pp. 15–37, Jan. 1992.

[5] P. S. Chang and A. N. Willson, "Analysis of conjugate gradient algorithms for adaptive filtering," *IEEE Trans. Signal Processing*, vol. 48, no. 2, pp. 409–418, 2000.

[6] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester U.K.: Willey, 1987.

[7] G. K. Boray and M. D. Srinath, "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. Circuits Syst. I*, vol. 39, pp. 1–10, Jan. 1992.

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

[9] K. Berberidis and P. Karaivazoglou, "An efficient block adaptive decision feedback equalizer implemented in the frequency domain," *IEEE Trans. Signal Processing*, vol. 50, no. 9, pp. 2273–2285, Sept. 2002.

[10] ETSI, UMTS, Tech. Rep. 101 112, vers. 3.2.0, 1998-04.