Analysis of Conjugate Gradient Algorithms for Adaptive Filtering

Pi Sheng Chang, *Member, IEEE*, and Alan N. Willson, Jr., *Fellow, IEEE*

***Abstract*—This paper presents and analyzes two approaches to the implementation of the conjugate gradient (CG) algorithm for adaptive filtering where several modifications to the original CG method are proposed. The convergence rates and misadjustments for the two approaches are compared. An analysis in the z-domain is used in order to find the asymptotic performance, and stability bounds are established. The behavior of the algorithms in finite word-length computation are described, and dynamic range considerations are discussed. It is shown that in finite word-length computation and close to steady state, the algorithms' behaviors are similar to the steepest descent algorithm, where the stalling phenomenon is observed. Using 16-bit fixed-point number representation, our simulations show that the algorithms are numerically stable.**

***Index Terms*—Adaptive filtering algorithms, conjugate gradient method.**

I. Introduction

In RECENT years, many adaptive filtering algorithms based on the conjugate gradient (CG) method of optimization have been reported [3], [4], [7], [9], [11], [15], [16]. In these works, several modifications have been proposed to improve the performance of the CG algorithm for various applications, but usually, the analysis of the proposed algorithms has not been shown. It is well known that the CG algorithm has a faster convergence rate than steepest descent [2], [12] and that it also has lower computational complexity when compared with the classic recursive least squares (RLS) algorithm [3], but mostly, its analysis can only be found in the optimization and matrix computation literature. Here, we will describe, from the signal processing point of view, two of the CG algorithm implementations and analyze their performance in steady state. Some related implementation ideas can also be found in [3] and [7]. In addition, their performance under finite word-length effects will be discussed. Due to the highly nonlinear nature of the algorithms, a linearized quantization model as used in the analysis of the LMS [6], NLMS [8] and RLS [5] algorithms, in general, cannot be applied.

In Section II, properties of the CG method of optimization will be discussed, and ways to implement the algorithm efficiently in the adaptive filtering context will be described. Section III compares and analyzes two implementations, and Section IV shows our simulation results.

Manuscript received February 17, 1998; revised May 21, 1999. This work was supported by the National Science Foundation under Grant MIP-9632698. The associate editor coordinating the review of this paper and approving it for publication was Dr. Hitoshi Kiya.

P. S. Chang was with the Integrated Circuits and Systems Laboratory, Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA. He is now with VTEL Corporation, Sunnyvale, CA 94086 USA.

A. N. Willson, Jr. is with the Integrated Circuits and Systems Laboratory, Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1600 USA.

Publisher Item Identifier S 1053-587X(00)00987-9.

II. Derivation of the Algorithm

The CG method can be applied to adaptive transversal filters as shown in [3], [15]. Doing this, the objective becomes the solving of

$$\mathbf{R}\mathbf{w} = \mathbf{b} \tag{1}$$

where $\mathbf{R}$ is the $N \times N$ correlation matrix of the input data vector $\mathbf{x}(n)$, and $\mathbf{b}$ is the cross-correlation vector between the input data and the desired response $d(n)$. If $\mathbf{R}$ and $\mathbf{b}$ are estimated as in [13] for the least-squares (LS) problem, the CG method offers an alternative way to solve for $\mathbf{w}$ instead of inverting the matrix $\mathbf{R}$. If they are estimated as in [3], where a sliding data window is used, then the CG method can be viewed as a stochastic gradient-based method. Many adaptive filtering applications require the weight coefficients to be updated at each incoming data sample. Although, with previously developed CG algorithms, this can be done at the expense of running several iterations per sample, we propose modifications here that will allow the algorithm to run just one iteration per sample but still maintain performance comparable with RLS or LMS-Newton. One of the main difficulties of the RLS and the LMS-Newton algorithms is the necessity to estimate $\mathbf{R}^{-1}$. If the estimated $\mathbf{R}^{-1}$ loses the property of positive definiteness, that will cause the algorithm to diverge [13]. This does not happen with the CG method since there is no need to compute the inverse of $\mathbf{R}$. The basic CG algorithm can be described as follows [12], [17] after some rearrangement for improved clarity:

Initial conditions:

$\mathbf{w}(0) = 0, \mathbf{g}(0) = \mathbf{b}, \rho(0) = \mathbf{g}(0)^T\mathbf{g}(0), \mathbf{p}(1) = \mathbf{g}(0), k = 1$

while $k \leq k_{\max}$,

begin

$$\mathbf{v}(k) = \mathbf{R}\mathbf{p}(k) \tag{2}$$

$$\alpha(k) = \rho(k-1)/\mathbf{p}(k)^T\mathbf{v}(k) \tag{3}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \alpha(k)\mathbf{p}(k) \tag{4}$$

$$\mathbf{g}(k) = \mathbf{g}(k-1) - \alpha(k)\mathbf{v}(k) \tag{5}$$

$$\rho(k) = \mathbf{g}(k)^T\mathbf{g}(k)$$

$$\beta(k) = \rho(k)/\rho(k-1) \tag{6}$$

$$\mathbf{p}(k+1) = \mathbf{g}(k) + \beta(k)\mathbf{p}(k) \tag{7}$$

$$k = k+1$$

end

where $\alpha(k)$ is the step size that minimizes the cost function

$\mathcal{F}(\mathbf{w}(k))$ (see Section II-A), $\beta(k)$ provides $\mathbf{R}$-orthogonality for the direction vector $\mathbf{p}(k)$, $\mathbf{g}(k)$ is the residual vector defined as

$$\mathbf{g}(k) = \mathbf{b} - \mathbf{R}\mathbf{w}(k) = -\nabla\mathcal{F}(\mathbf{w}(k)) \tag{8}$$

with ∇f denoting the gradient of the function f and k denoting the iteration number.

The formulation above has several desirable properties such as the following.

1) $\mathbf{R}$-orthogonality or conjugacy with respect to $\mathbf{R}$ of the vectors $\mathbf{p}(k)$ [2], [12], [17], i.e.,

$$\mathbf{p}(k)^T\mathbf{R}\mathbf{p}(i) = 0, \quad \text{for all } k \neq i. \tag{9}$$

2) Orthogonality of the gradient (residual) vectors [2], [12]:

$$\mathbf{g}(k)^T\mathbf{g}(i) = 0, \quad \text{for } i = 0, \cdots, k-1. \tag{10}$$

3) The so-called expanding subspace theorem [17], where the residual vectors $\mathbf{g}(k)$, $k = 0, \cdots, k_{\max}$, satisfy

$$\mathbf{g}(k)^T\mathbf{p}(i) = 0, \quad \text{for } i \leq k. \tag{11}$$

4) The finite termination property [2], [14], [17], where

$$k_{\max} \leq N \tag{12}$$

is sufficient for the algorithm to minimize $\mathcal{F}(\mathbf{w}(k))$.

5) The descent property, which is given by [10]:

$$\mathbf{p}(k+1)^T\mathbf{g}(k) \geq 0. \tag{13}$$

To see that the CG algorithm satisfies the descent property, we post-multiply the transpose of (7) by $\mathbf{g}(k)$

$$\mathbf{p}(k+1)^T\mathbf{g}(k) = \mathbf{g}(k)^T\mathbf{g}(k) + \beta(k)\mathbf{p}(k)^T\mathbf{g}(k) \geq 0 \tag{14}$$

and recognize that the second term on the right-hand side is zero, due to (11).

Variations of the algorithm described in (2)–(7) can be found in [2], [12], [14], and [17], where it is shown that one can use an iterative method to terminate the algorithm, instead of using the fixed $k_{\max}$ iterations, or use different ways to compute α and β.

An alternative expression for the computation of α is given by

$$\alpha(k) = \frac{\mathbf{g}(k-1)^T\mathbf{p}(k)}{\mathbf{p}(k)^T\mathbf{R}\mathbf{p}(k)}. \tag{15}$$

This expression is obtained by post-multiplying the transpose of (5) by $\mathbf{p}(k)$, resulting in

$$\mathbf{g}(k)^T\mathbf{p}(k) = \mathbf{g}(k-1)^T\mathbf{p}(k) - \alpha(k)\mathbf{p}(k)^T\mathbf{R}\mathbf{p}(k) \tag{16}$$

which leads to (15) after using (11). If we further substitute (14) into (15) and use (11) one more time, we get (3). In the presence of computational errors, (11) will not be *exactly* zero, and using (15) rather than (3) will result in less computational error in the algorithm.

The existence of only one matrix-vector multiplication in (2)–(7) is possible due to the use of a recursive formulation for the residual vector [12], [15]

$$\mathbf{g}(k) = \mathbf{g}(k-1) - \alpha(k)\mathbf{R}\mathbf{p}(k). \tag{17}$$

This expression can be found by substituting (4) into (8) and by assuming that $\mathbf{R}$ and $\mathbf{b}$ are constant throughout the $k_{\max}$ iterations, which is applicable for the case of block processing. It has been used in [15] and [18]. Here, we propose modifications that allow it to be used in nonblock processing or sample-by-sample updating. Notice that the iteration number k will be replaced by the time instant n since, after the proposed modifications, only one iteration will be performed per time instant.

A. Considerations About the Cost Function

When the CG algorithm is used to solve (1), it is indirectly minimizing a cost function defined as

$$\mathcal{F}(\mathbf{w}(n)) = \tfrac{1}{2}\mathbf{w}(n)^T\mathbf{R}\mathbf{w}(n) - \mathbf{b}^T\mathbf{w}(n). \tag{18}$$

The way $\mathbf{R}$ and $\mathbf{b}$ are estimated will directly influence the performance of the algorithm. There are two ways that we can compute $\mathbf{R}$ and $\mathbf{b}$ by using different schemes of data windowing.

1) Finite Sliding Data Window: In this case, only the data samples inside a window of finite length M are used. The correlation matrix and the cross-correlation vector are estimated by the time ensemble averages

$$\mathbf{R}(n) = \frac{1}{M}\sum_{j=n-M+1}^{n}\mathbf{x}(j)\mathbf{x}(j)^T$$

$$\mathbf{b}(n) = \frac{1}{M}\sum_{j=n-M+1}^{n}d(j)\mathbf{x}(j).$$

The residual vector is then computed as

$$\mathbf{g}(n) = \mathbf{b}(n) - \mathbf{R}(n)\mathbf{w}(n) \tag{19}$$

$$= \frac{1}{M}\sum_{j=n-M+1}^{n}[(d(j) - \mathbf{w}(j)^T\mathbf{x}(j))\mathbf{x}(j)]. \tag{20}$$

The formulation in (20) is computationally more efficient than (19) if M is smaller than N, which is the length of the input data vector $\mathbf{x}(n)$.

2) Exponentially Decaying Data Window: By using the exponentially decaying data window, the resulting correlation function is the same as the one used by the RLS algorithm. When used with the CG algorithm, a performance comparable to the RLS algorithm can be achieved. The correlation and cross-correlation functions are given by

$$\mathbf{R}(n) = \lambda_f\mathbf{R}(n-1) + \mathbf{x}(n)\mathbf{x}(n)^T \tag{21}$$

and

$$\mathbf{b}(n) = \lambda_f\mathbf{b}(n-1) + d(n)\mathbf{x}(n) \tag{22}$$

where λ_f is the forgetting factor. For sample-by-sample processing, a recursive formulation for the residual vector can be found by using (4), (8), (21) and (22), resulting in

$$\begin{aligned}\mathbf{g}(n) &= \mathbf{b}(n) - \mathbf{R}(n)\mathbf{w}(n)\\ &= \lambda_f\mathbf{g}(n-1) - \alpha(n)\mathbf{R}(n)\mathbf{p}(n)\\ &\quad + \mathbf{x}(n)[d(n) - \mathbf{x}(n)^T\mathbf{w}(n-1)].\end{aligned} \tag{23}$$

B. Termination

There are many schemes proposed in the literature to terminate the CG algorithm. In [12], an iterative scheme was proposed, based on the norm of the residual and a maximum number of iterations. In [3], the algorithm terminates after $\min(N, M)$ iterations, due to the use of a finite sliding data window in the computation of $\mathbf{R}$ and, consequently, the residual vector. Either way, the CG algorithm has to run several iterations per data update in order to converge. This is not a problem when block processing is used, but in sample-by-sample updating, the procedure is computationally costly. One way to employ just one iteration per coefficient and data update is to use some *degenerated* scheme. By degeneration, we mean that $\mathbf{g}(n)$ will not be completely orthogonal to the subspace spanned by $\{\mathbf{p}(0), \mathbf{p}(1), \cdots, \mathbf{p}(n)\}$ or, in other words, $\mathbf{g}(n)^T\mathbf{p}(i) = 0$, for $i < n$, will not hold. Some other examples of degenerated schemes are a) using a constant value for α and b) using a nonconstant matrix $\mathbf{R}$ at each iteration. For the former, it is well-known that using α, as given in (3), minimizes the cost function $\mathcal{F}(\mathbf{w}(n))$ on the line $\mathbf{w}(n-1) + \alpha\mathbf{p}(n)$, whereas $\mathcal{F}(\mathbf{w}(n))$ will not be completely minimized using a constant value for α. By using a nonconstant matrix $\mathbf{R}$ at each iteration, the algorithm can be used in a nonblock adaptation scheme. The new update of the residual in this case is given by (23).

C. Line Search

In the CG algorithm, α is the step size used in the update of the weight vector, as shown in (4). The value of α is usually chosen so that $\mathcal{F}(\mathbf{w}(n-1) + \alpha\mathbf{p}(n))$ is minimized. Explicitly computing α for the cost function shown in (18) results in (3). This is an exact line search along the direction $\mathbf{p}(n)$. Inexact line search schemes with reduced complexity can also be used, but they must satisfy the convergence bound given in the Appendix. When the exponentially decaying data window is used, we have

$$\alpha(n) = \eta\frac{\mathbf{p}(n)^T\mathbf{g}(n-1)}{\mathbf{p}(n)^T\mathbf{R}(n)\mathbf{p}(n)}, \quad (\lambda_f - 0.5) \le \eta \le \lambda_f. \quad (24)$$

Notice that using $\mathbf{g}(n-1)^T\mathbf{g}(n-1)$ instead of $\mathbf{p}(n)^T\mathbf{g}(n-1)$, as shown in (3), is less effective for the degenerated scheme (see the Appendix).

Another expression for $\alpha(n)$, which preserves orthogonality or the so-called *expanding subspace theorem* [17] by ensuring that $\mathbf{p}(n)^T\mathbf{g}(n) = 0$, is given by

$$\alpha(n) = \frac{\lambda_f\mathbf{p}(n)^T\mathbf{g}(n-1) + \mathbf{p}(n)^T\mathbf{x}(n)[d(n) - \mathbf{x}(n)^T\mathbf{w}(n-1)]}{\mathbf{p}(n)^T\mathbf{R}(n)\mathbf{p}(n)}.$$

This is obtained by premultiplying (23) by $\mathbf{p}(n)^T$ and applying (11). Note that this $\alpha(n)$ also minimizes $\mathcal{F}(\mathbf{w}(n-1) + \alpha(n)\mathbf{p}(n))$.

D. Resetting the Algorithm

For sample-by-sample processing, it is important to periodically reset the direction vector $\mathbf{p}(n)$ to the true gradient in order to ensure the convergence of the algorithm. The degenerated scheme will not allow the algorithm to converge in N steps.

TABLE I
MODIFIED CG ALGORITHM

Set initial conditions:
$\mathbf{w}(0) = 0,\ \mathbf{g}(0) = \mathbf{b}(0),\ \mathbf{p}(1) = \mathbf{g}(0),\ n = 1.$

$$\alpha(n) = \eta\frac{\mathbf{p}(n)^T\mathbf{g}(n-1)}{\mathbf{p}(n)^T\mathbf{R}(n)\mathbf{p}(n)} \quad (50)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \alpha(n)\mathbf{p}(n) \quad (51)$$

$$\mathbf{g}(n) = \lambda_f\mathbf{g}(n-1) - \alpha(n)\mathbf{R}(n)\mathbf{p}(n) + \mathbf{x}(n)(d(n) - \mathbf{x}(n)^T\mathbf{w}(n-1)) \quad (52)$$

$$\beta(n) = \frac{(\mathbf{g}(n) - \mathbf{g}(n-1))^T\mathbf{g}(n)}{\mathbf{g}(n-1)^T\mathbf{g}(n-1)} \quad (53)$$

$$\mathbf{p}(n+1) = \mathbf{g}(n) + \beta(n)\mathbf{p}(n) \quad (54)$$

How often the algorithm is reset will influence its performance. If using a certain set of direction vectors $\mathbf{p}(n)$ does not increase the cost function, then global convergence can be assured since a true steepest-descent step is taken every time the algorithm is reset. A non-reset method can also be used, but the Polak-Ribiere method [10], [17], [20] for the computation of β, which is given by

$$\beta(n) = \frac{(\mathbf{g}(n) - \mathbf{g}(n-1))^T\mathbf{g}(n)}{\mathbf{g}(n-1)^T\mathbf{g}(n-1)} \quad (25)$$

should be used for improved performance. Simulations have shown that (25) performs better than (6) when using a degenerated scheme because $\mathbf{g}(n-1)^T\mathbf{g}(n)$ will not be exactly zero. Table I shows an implementation of the algorithm, taking into account some of the considerations discussed.

III. Analysis of the Conjugate Gradient Algorithm

In the previous section, we presented several approaches to the implementation of the CG algorithm in adaptive filtering. Here, we will analyze two of the proposed approaches. The first one, which we call CG1, assumes a variable autocorrelation matrix $\mathbf{R}$ and cross-correlation vector $\mathbf{b}$, which are updated for each input data sample, and only one iteration of the algorithm is performed per time instant. The second approach assumes constant $\mathbf{R}$ and $\mathbf{b}$ within the internal iterations, and N or fewer internal iterations are performed per input data sample, where N is the dimension of $\mathbf{R}$. We call this algorithm CG2.

It has been shown in the quadratic optimization literature that the CG algorithm converges in finite steps for constant $\mathbf{R}$ and $\mathbf{b}$ [2], [12]. This is used in the analysis of the algorithm CG2 and the algorithm presented in [3]. The advantage of CG2 is that the convergence rate is independent of the eigenvalue spread of $\mathbf{R}$, whereas the disadvantage is that, when a finite data window is used to estimate the autocorrelation and cross-correlation, the

output mean-squared error is dependent on the length of the data window.

In the algorithm CG1, which was previously reported in [7], both finite data windows and exponentially decaying data windows can be used, although the latter gives better performance due to the resulting better estimation of $\mathbf{R}$ and $\mathbf{b}$, as will be shown in Section IV.

A. The Conjugate Gradient Algorithms

Algorithm CG1: The CG algorithm using the first approach is shown in Table I, where it minimizes a cost function defined as $\mathcal{F}(\mathbf{w}(n)) = \mathbf{w}(n)^T\mathbf{R}(n)\mathbf{w}(n)/2 - \mathbf{b}(n)^T\mathbf{w}(n)$. $\mathbf{R}(n)$ is the $N \times N$ sample correlation matrix of the input data vector $\mathbf{x}(n)$ computed as in (21), and $\mathbf{b}(n)$ is the $N \times 1$ cross-correlation vector computed as in (22).

In state-space notation, the algorithm CG1 can be written as shown in the first equation at the bottom of the page.

Algorithm CG2: Following the same approach used in [3][1] and in Section II, the second CG algorithm can be described as follows:

Set initial condition: $\mathbf{w}(0) = 0$.

For each time instant n, compute:

Start:

$$\tilde{\mathbf{R}}(n) = \frac{1}{M}\sum_{i=0}^{M-1}\mathbf{x}(n-i)\mathbf{x}(n-i)^T \tag{26}$$

$$\tilde{\mathbf{b}}(n) = \frac{1}{M}\sum_{i=0}^{M-1}d(n-i)\mathbf{x}(n-i)$$

$$\mathbf{g}(0) = \tilde{\mathbf{b}}(n) - \tilde{\mathbf{R}}(n)\mathbf{w}(0), \quad \mathbf{p}(1) = \mathbf{g}(0)$$

for $k = 1$ to $k_{\max}$ do:

$$\alpha(k) = \frac{\mathbf{g}(k-1)^T\mathbf{g}(k-1)}{\mathbf{p}(k)^T\tilde{\mathbf{R}}(n)\mathbf{p}(k)}$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \alpha(k)\mathbf{p}(k)$$

$$\mathbf{g}(k) = \mathbf{g}(k-1) - \alpha(k)\tilde{\mathbf{R}}(n)\mathbf{p}(k)$$

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

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

After $k_{\max}$ iterations, do:

$$\tilde{\mathbf{w}}(n) = \mathbf{w}(k_{\max})$$

$$\mathbf{w}(0) = \tilde{\mathbf{w}}(n)$$

$$n = n+1$$

goto Start.

Notice that $\tilde{\mathbf{R}}(n)$ is fixed throughout the k iterations, and only the final vector $\mathbf{w}(k_{\max})$ is of interest. Here, $k_{\max} = \min(N, M)$ since if $M < N$, there are only M distinct direction vectors [3].

In state-space notation, CG2 can be written as the second equation at the bottom of the page, where $\delta(k)$ is the unit-sample sequence.

B. CG Algorithm in Signal-Flow-Graph Representation and Asymptotic Analysis

Using the state-space representation given previously, we can view the CG algorithms as nonlinear time-varying digital filters. First consider the algorithm CG1, and in order to simplify its analysis, it is assumed that the input signal is wide-sense stationary and ergodic, $E[\alpha(n)] = \overline{\alpha}$, $E[\beta(n)] = \overline{\beta}$, $E[\mathbf{b}(n)] = \mathbf{b}$, and $E[\mathbf{R}(n)] = \mathbf{R}$, where $E[x]$ denotes the expected value of x, and $\alpha(n), \beta(n)$, $\mathbf{p}(n), \mathbf{w}(n)$, $\mathbf{g}(n), \mathbf{b}(n)$, and $\mathbf{R}(n)$ are assumed to be statistically independent with respect to each other. With the expectation operator applied to the state variables, we can view the system as being linear and time invariant. Furthermore, let us define $\mathbf{W}(z) = \mathcal{Z}\{E[\mathbf{w}(n)]\}$, $\mathbf{G}(z) = \mathcal{Z}\{E[\mathbf{g}(n)]\}$, and $\mathbf{P}(z) = \mathcal{Z}\{E[\mathbf{p}(n)]\}$, where $\mathcal{Z}\{x\}$ is the z-transform of x, and note that (52) can also be written as

$$\mathbf{g}(n) = \mathbf{b}(n)u(n) - \mathbf{R}(n)\mathbf{w}(n) \tag{27}$$

where $u(n)$ is the unit-step sequence.

Now, we can find the transfer function for $\mathbf{W}(z)$ using (51) and (54) in Table I and (27). The signal-flow-graph representation of these three equations is shown in Fig. 1, where we have, after taking the expectation on both sides of these equations and then the z-transform:

$$\mathbf{W}(z) = \mathbf{W}(z)z^{-1} + \overline{\alpha}\mathbf{P}(z) \tag{28}$$

$$z\mathbf{P}(z) = \overline{\beta}\mathbf{P}(z) + \mathbf{G}(z) \tag{29}$$

$$\mathbf{G}(z) = \frac{\mathbf{b}z}{z-1} - \mathbf{R}\mathbf{W}(z). \tag{30}$$

[1]The algorithm presented in [3] has a different formulation for $\alpha(k)$, $\mathbf{g}(k)$, and $\mathbf{p}(k+1)$, but, computationally, it has the same behavior as the algorithm CG2 described here.

$$\begin{bmatrix}\mathbf{w}(n)\\ \mathbf{g}(n)\\ \mathbf{p}(n)\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \alpha(n)\mathbf{I} & \alpha(n)\beta(n-1)\mathbf{I}\\ -\mathbf{x}(n)\mathbf{x}(n)^T & \lambda_f\mathbf{I} - \alpha(n)\mathbf{R}(n) & -\alpha(n)\beta(n-1)\mathbf{R}(n)\\ \mathbf{0} & \mathbf{I} & \beta(n-1)\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{w}(n-1)\\ \mathbf{g}(n-1)\\ \mathbf{p}(n-1)\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{x}(n)d(n)\\ \mathbf{0}\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{w}(k)\\ \mathbf{g}(k)\\ \mathbf{p}(k)\end{bmatrix} = \begin{bmatrix}\mathbf{I} & \alpha(k)\mathbf{I} & \alpha(k)\beta(k-1)\mathbf{I}\\ \mathbf{0} & \mathbf{I} - \alpha(k)\tilde{\mathbf{R}}(n) & -\alpha(k)\beta(k-1)\tilde{\mathbf{R}}(n)\\ \mathbf{0} & \mathbf{I} & \beta(k-1)\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{w}(k-1)\\ \mathbf{g}(k-1)\\ \mathbf{p}(k-1)\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ (\tilde{\mathbf{b}}(n) - \tilde{\mathbf{R}}(n)\mathbf{w}(0))\delta(k)\\ \mathbf{0}\end{bmatrix}$$

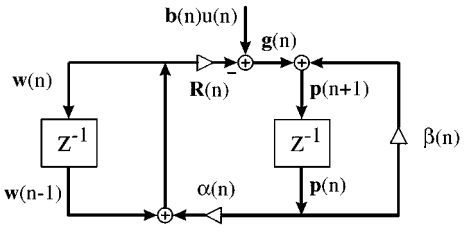

Fig. 1. Signal-flow graph representation of the CG algorithm.

Solving for $\mathbf{W}(z)$, we get

$$\mathbf{W}(z) = ((z-1)(z-\overline{\beta})\mathbf{I} + \overline{\alpha}\mathbf{R}z)^{-1}\,\frac{\overline{\alpha}\mathbf{b}z^2}{(z-1)}$$

and knowing that $\mathbf{W}(z) = \mathbf{W}^+(z)$, which is the one-sided z transform, we can use the *final value theorem* [19], which leads to

$$\lim_{n\to\infty} E[\mathbf{w}(n)] = \lim_{z\to 1}(z-1)\mathbf{W}^+(z) = \mathbf{R}^{-1}\mathbf{b}. \tag{31}$$

The above limit exists if $(z-1)\mathbf{W}^+(z)$ is stable; therefore, we must have $|\overline{\beta}| < 1$ and the roots of

$$\det((z-1)(z-\overline{\beta})\mathbf{I} + \overline{\alpha}\mathbf{R}z) = 0 \tag{32}$$

must lie inside the unit circle. Expression (31) shows that $E[\mathbf{w}(n)]$ will converge to $\mathbf{w}^*$ for $n \to \infty$, where $\mathbf{w}^*$ is the optimum weight vector. We can apply a unitary transformation to $\mathbf{R}$ so that $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$ and, knowing that $\det(\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T) = \det\boldsymbol{\Lambda}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix whose elements λ_i are the eigenvalues of $\mathbf{R}$, (32) becomes

$$\prod_{i=0}^{N-1}(z^2 + (\overline{\alpha}\lambda_i - (\overline{\beta}+1))z + \overline{\beta}) = 0. \tag{33}$$

Looking at each second-order term of (33) to ensure that the roots of the second-order term lie inside the unit circle in the z-plane, we must have

$$\begin{aligned} -1 &\le \overline{\beta} + \overline{\alpha}\lambda_i - \overline{\beta} - 1 \\ -1 &\le \overline{\beta} - \overline{\alpha}\lambda_i + \overline{\beta} + 1 \\ -1 &\le \overline{\beta} \le 1. \end{aligned}$$

A sufficient condition for the stability of the system described by (28)–(30), where the poles of the system are the roots of (33), is

$$0 \le \overline{\alpha} \le \frac{2\overline{\beta}+2}{\lambda_{\max}}. \tag{34}$$

For $\overline{\beta} \to 0$, we have

$$0 \le \overline{\alpha} \le \frac{2}{\lambda_{\max}} \tag{35}$$

which agrees with the results obtained for the *steepest descent* algorithm [13]. For Algorithm CG2, since $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{b}}(n)$ are constant throughout the k iterations, the analysis presented in [2] and [12] can be readily applied.

C. Convergence and Misadjustment

First, consider the algorithm CG1, where an exponentially decaying data window is used for the computation of $\mathbf{R}(n)$ and $\mathbf{b}(n)$, and the updated weight-vector is obtained as the result of a single iteration. The convergence rate will depend on the eigenvalue spread. Using variable $\mathbf{R}(n)$ for each iteration reduces substantially the computational complexity of the algorithm. In steady state, as $\mathbf{R}(n) \to \mathbf{R}$, the misadjustment for the algorithm CG1 will be equal to the misadjustment of the RLS algorithm since both algorithms minimize a cost function given by

$$\begin{aligned} \mathcal{F}(\mathbf{w}(n)) &= \tfrac{1}{2}\mathbf{w}(n)^T\mathbf{R}(n)\mathbf{w}(n) - \mathbf{b}(n)^T\mathbf{w}(n) \\ &= \tfrac{1}{2}\mathbf{w}(n)^T\sum_{i=0}^{n}\lambda_f^{n-i}\mathbf{x}(i)\mathbf{x}(i)^T\mathbf{w}(n) \\ &\quad - \sum_{i=0}^{n}\lambda_f^{n-i}\,d(i)\mathbf{x}(i)^T\mathbf{w}(n) \\ &= \tfrac{1}{2}\sum_{i=0}^{n}\lambda_f^{n-i}e(i)^2 - \tfrac{1}{2}\sum_{i=0}^{n}\lambda_f^{n-i}d(i)^2 \end{aligned}$$

where $e(i) = d(i) - \mathbf{w}(n)^T\mathbf{x}(i)$.

Next, consider Algorithm CG2, where a finite-length data window is used for the estimation of $\mathbf{R}$ and $\mathbf{b}$. The convergence rate does not depend on the eigenvalue spread of the correlation matrix because of the way the algorithm is implemented, where the updated weight vector at each time instant n is the last updated weight vector after $k_{\max}$ iterations. Therefore, we have, considering that the algorithm converges after the $k_{\max}$ iterations

$$\tilde{\mathbf{w}}(n) \approx \tilde{\mathbf{R}}(n)^{-1}\tilde{\mathbf{b}}(n). \tag{36}$$

This means that at each time instant n, $\tilde{\mathbf{w}}(n)$ is the optimum solution for the given $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{b}}(n)$, so that the convergence of the algorithm in n will not depend on the convergence of the algorithm in k.

Now, for the analysis of the misadjustment of Algorithm CG2, consider it being used in the system identification (SI) configuration. The desired response is the output of the FIR filter with optimum weight coefficients (plant) given by $d(n) = \mathbf{w}^{*T}\mathbf{x}(n)$ when there is no measurement noise. The output error of the system is

$$\begin{aligned} e(n) &= d(n) - \tilde{\mathbf{w}}(n)^T\mathbf{x}(n) \\ &= (\mathbf{w}^* - \tilde{\mathbf{w}}(n))^T\mathbf{x}(n) = \boldsymbol{\epsilon}(n)^T\mathbf{x}(n) \end{aligned}$$

and the mean-squared error is

$$E[e(n)^2] = E[\mathbf{x}(n)^T\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}(n)^T\mathbf{x}(n)] = \mathrm{tr}[\mathbf{R}\mathbf{K}(n)]$$

where $\mathbf{K}(n) = E[\boldsymbol{\epsilon}(n)\boldsymbol{\epsilon}(n)^T]$, and $\boldsymbol{\epsilon}(n) = \mathbf{w}^* - \tilde{\mathbf{w}}(n)$ [13]. For the SI configuration with white Gaussian noise as the input signal, $\mathbf{R} = \sigma_x^2\mathbf{I}$, and we have

$$E[e(n)^2] = \sigma_x^2\ \mathrm{tr}\ \mathbf{K}(n) = \sigma_x^2 E[\|\boldsymbol{\epsilon}(n)\|^2]. \tag{37}$$

When using $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{b}}(n)$ to estimate $\mathbf{R}$ and $\mathbf{b}$, we have to consider the variance of the estimators. The greater the variance,

the "noisier" the resulting weight-vector will be. After $k_{\max}$ iterations, we have

$$\tilde{\mathbf{R}}(n)\tilde{\mathbf{w}}(n) \approx \tilde{\mathbf{b}}(n). \tag{38}$$

This is equivalent to saying that the norm of the residual vector satisfies

$$||\mathbf{g}(k_{\max})|| = ||\tilde{\mathbf{b}}(n) - \tilde{\mathbf{R}}(n)\mathbf{w}(k_{\max})|| < \epsilon$$

where ϵ can be made arbitrarily small [2]. Consider the case when $\mathbf{R} = \sigma_x^2\mathbf{I}$ so that $\tilde{r}_{ij}(n) \approx 0$, for $i \neq j$, where $\tilde{r}_{ij}(n)$ is an element of $\tilde{\mathbf{R}}(n)$. Then, (38) becomes

$$\tilde{r}_{jj}(n)\tilde{w}_j(n) \approx \tilde{b}_j(n)$$

where $\tilde{w}_j(n)$ and $\tilde{b}_j(n)$ are the elements of $\tilde{\mathbf{w}}(n)$ and $\tilde{\mathbf{b}}(n)$, respectively.

Now, consider the inequality presented in [12] that shows a bound for the norm of the weight-error vector

$$||\mathbf{w}^* - \mathbf{w}(k)||_{R_1} \leq 2||\mathbf{w}^* - \mathbf{w}(0)||_{R_1}\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{39}$$

where $||\mathbf{w}||_{R_1} = \sqrt{\mathbf{w}^T\mathbf{R}_1\mathbf{w}}$, $\kappa = ||\mathbf{R}_1||_2||\mathbf{R}_1^{-1}||_2$ is the condition number, and $\mathbf{R}_1 = \tilde{\mathbf{R}}(n)$ in the context presented here. Using $\mathbf{R} = E[\tilde{\mathbf{R}}(n)] = \sigma_x^2\mathbf{I}$ and taking the expectation of both sides of (39), we have $E[\mathbf{w}(k)] = \mathbf{w}^*$ and since $\tilde{\mathbf{w}}(n) = \mathbf{w}(k_{\max})$, we can conclude that $E[\tilde{\mathbf{w}}(n)] = \mathbf{w}^*$, which shows that the weight-vector is convergent in the mean.

The variance of the weight-vector $\tilde{\mathbf{w}}(n)$ is given by

$$\begin{aligned}\text{var}(\tilde{w}_j(n)) &= E[(\tilde{w}_j(n) - E[\tilde{w}_j(n)])^2] \\ &= E[((w_j^* - \epsilon_j(n)) - E[\tilde{w}_j(n)])^2] \\ &= E[(w_j^* - \epsilon_j(n) - w_j^*)^2] = E[\epsilon_j(n)^2].\end{aligned}$$

Due to the computation of $\tilde{w}_j(n)$ using the CG algorithm, it is difficult to find the variance of $\tilde{w}_j(n)$ directly from the algorithm because of its iterative nature. Here, we will consider

$$\text{var}(\tilde{w}_j(n)) \approx \text{var}(\tilde{r}_{jj}(n))\text{var}(\tilde{b}_j(n)) \tag{40}$$

which provides a good approximation for the variance of $\tilde{w}_j(n)$, as shown in our simulations.

Consider now the notation of the input data vector as

$$\mathbf{x}(n-i) = \begin{bmatrix} x_0(n-i) \\ x_1(n-i) \\ \vdots \\ x_{N-1}(n-i) \end{bmatrix} = \begin{bmatrix} x(n-i) \\ x(n-i-1) \\ \vdots \\ x(n-i-N+1) \end{bmatrix}.$$

Then, we can write

$$\begin{aligned} r_{jj} &= E[x(n-j)^2] \\ \tilde{r}_{jj}(n) &= \frac{1}{M}\sum_{i=0}^{M-1} x(n-i-j)^2 \end{aligned} \tag{41}$$

and

$$E[\tilde{r}_{jj}(n)] = r_{jj}.$$

The variance of $\tilde{r}_{jj}(n)$ is given by[2]

$$\begin{aligned}\text{var}(\tilde{r}_{jj}(n)) &= E[(\tilde{r}_{jj}(n) - r_{jj})^2] \\ &= E[\tilde{r}_{jj}(n)^2 + r_{jj}^2 - 2\tilde{r}_{jj}(n)r_{jj}] \\ &= E\left[\frac{1}{M^2}\left\{\sum_{i=0}^{M-1} x(n-i-j)^4 \right.\right. \\ &\quad \left.\left. + \sum_{i\neq k}^{M-1}\sum^{M-1} x(n-i-j)^2x(n-k-j)^2\right\}\right. \\ &\quad \left. + \sigma_x^4 - \frac{2\sigma_x^2}{M}\sum_{i=0}^{M-1} x(n-i-j)^2\right] \\ &= \frac{1}{M^2}[Mm_4 + M(M-1)\sigma_x^4] \\ &\quad + \sigma_x^4 - \frac{2\sigma_x^2}{M}M\sigma_x^2 \approx \frac{m_4}{M}\end{aligned} \tag{42}$$

where $m_4 = E[x(n)^4]$ is the fourth-order central moment [21], considering that $E[x(n)] = 0$.

To compute the variance of $\tilde{b}_j(n)$, we first notice that

$$\begin{aligned}\tilde{\mathbf{b}}(n) &= \frac{1}{M}\sum_{i=0}^{M-1} d(n-i)\mathbf{x}(n-i) \\ d(n-i) &= \mathbf{w}^{*T}\mathbf{x}(n-i) = \sum_{k=0}^{N-1} w_k^*x(n-i-k)\end{aligned}$$

so that

$$\begin{aligned}\tilde{b}_j(n) &= \frac{1}{M}\sum_{i=0}^{M-1} d(n-i)x(n-i-j) \\ &= \frac{1}{M}\sum_{i=0}^{M-1}\sum_{k=0}^{N-1} w_k^*x(n-i-k)x(n-i-j).\end{aligned}$$

Now, consider

$$\frac{1}{M}\sum_{i=0}^{M-1} x(n-i-k)x(n-i-j) \approx 0, \qquad k \neq j$$

because $r_{kj} = 0$, for $k \neq j$ when $\mathbf{R} = \sigma_x^2\mathbf{I}$. Therefore, we have

$$\begin{aligned}\tilde{b}_j(n) &= \frac{1}{M}\sum_{i=0}^{M-1} w_j^*x(n-i-j)^2 \\ &= \frac{w_j^*}{M}\sum_{i=0}^{M-1} x(n-i-j)^2 = w_j^*\tilde{r}_{jj}(n).\end{aligned} \tag{43}$$

Comparing (43) with (42), it is easy to see that

$$\text{var}(\tilde{b}_j(n)) = w_j^{*2}\frac{m_4}{M}. \tag{44}$$

[2]A similar procedure used to determine the variance of an estimator is shown in [21].

For Gaussian input signals, the kurtosis of the signal, $\nu_x = E[x_i^4]/\sigma_x^4$, is 3, so we have $m_4/M = 3\sigma_x^4/M$, and the mean-squared error is given by

$$\begin{aligned} E[e(n)^2] &= \sigma_x^2 E[\|\boldsymbol{\epsilon}(n)\|^2] \\ &= \sigma_x^2 E\left[\sum_{j=0}^{N-1} \epsilon_j(n)^2\right] = \sigma_x^2 \sum_{j=0}^{N-1} \text{var}(\tilde{w}_j(n)) \\ &\approx \sigma_x^2 \sum_{j=0}^{N-1} \frac{9\sigma_x^8}{M^2} w_j^{*2} = \frac{9\sigma_x^{10}}{M^2}\|\mathbf{w}^*\|^2. \end{aligned}$$

This shows the dependence of the misadjustment on the length M, as has been suggested in the simulation results in [3]. Table II shows the performance of CG2 for various values of M. As M increases, the theoretical results converge to the simulation results, showing that (40) is a suitable approximation for the analysis developed here. Further results are shown in the next section.

D. Finite Word-Length Effects

Due to the nonlinear nature of the CG algorithm, it is not possible to use additive quantization noise to model the quantization effects, as has been done in [5], [6], and [8]. Quantizing the variables in the CG algorithm will lead some of them to become zero, changing completely the behavior of the algorithm. This is particularly true for the variables $\alpha(n)$ and $\beta(n)$. Consider, for example, $\beta(n)$ as in (53), in fixed-point computation. In order to be able to update $\mathbf{p}(n+1)$, it is necessary to have

$$Q[\beta(n)] > 0$$

or

$$Q\left[Q[(\mathbf{g}(n)-\mathbf{g}(n-1))^T\mathbf{g}(n)]Q\left[\frac{1}{Q[\mathbf{g}(n-1)^T\mathbf{g}(n-1)]}\right]\right] \geq 2^{-B-1}$$

which implies that we should have

$$Q[(\mathbf{g}(n)-\mathbf{g}(n-1))^T\mathbf{g}(n)] \geq 2^{-B-1}$$

or

$$\sum Q[(g_i(n)-g_i(n-1))g_i(n)] \geq 2^{-B-1}$$

where B is the number of bits used to represent the fractional part of a number in fixed-point notation [6], [8], and $Q[\cdot]$ is the quantization operation. This implies that each element of $\mathbf{g}(n)$, given by $g_i(n)$, must satisfy

$$g_i(n) \geq 2^{(-B-1)/2}$$

which means that only half of the dynamic range of $g_i(n)$ is used in the computation of $\beta(n)$. Usually, when the algorithm converges, the residual vector $\mathbf{g}(n)$ will be close to zero. Due to quantization, the new value of $\beta(n)$ will be zero, and $\mathbf{p}(n+1)$ will not be updated by the additional term. The algorithm, under these circumstances, will behave like the steepest descent algorithm, where $\mathbf{p}(n+1)$ will be equal to the residual vector $\mathbf{g}(n)$. Figs. 2 and 3 show the values of $\alpha(n)$ and $\beta(n)$ for a single run of CG1 in fixed-point arithmetic with $\lambda_f = 0.99$, $\eta = 0.9$, $N = 20$, $\sigma_x^2 = 0.1$, and SNR = 30 dB, using 10 bits for the fractional part and six bits for the integer part of the number representation.

TABLE II
PERFORMANCE FOR VARIOUS LENGTHS OF DATA WINDOWING $\sigma_x^2 = 0.25$, $N = 5$, $\|\mathbf{w}^*\|^2 = 1.3071$ WITH NO MEASUREMENT NOISE, AND WITH RESULTS AVERAGED OVER 50 INDEPENDENT TRIALS

M	MSE (Simul.)	MSE (Theor.)
5	−25.53 dB	−33.38 dB
10	−33.30 dB	−39.40 dB
15	−39.09 dB	−42.92 dB
20	−43.02 dB	−45.42 dB
25	−46.04 dB	−47.36 dB

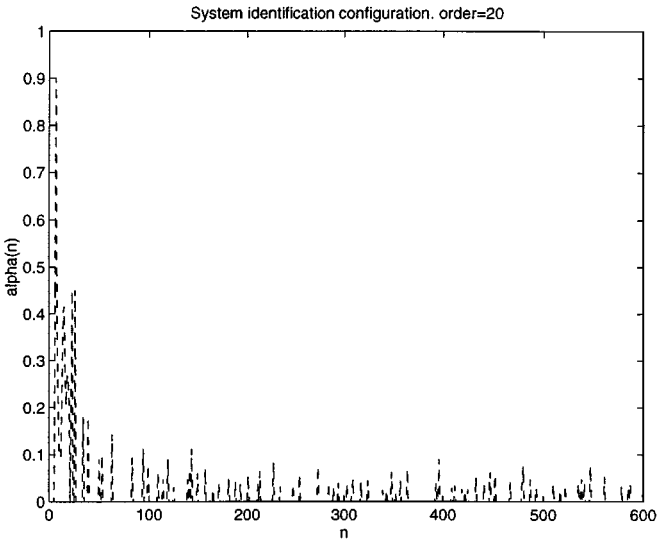

Fig. 2. $\alpha(n)$ in fixed-point computation.

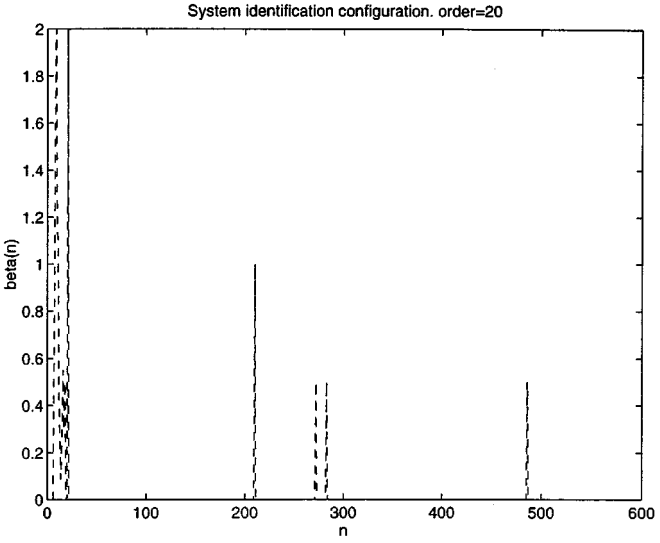

Fig. 3. $\beta(n)$ in fixed-point computation.

TABLE III
CG ALGORITHM USING DIFFERENT IMPLEMENTATIONS OF β AND RESET SCHEMES. $\lambda_f = 0.99$, SNR = 30 dB, $\sigma_x^2 = 1$

	MSE	
CG algorithm with:	$\eta = 0.8$	$\eta = 0.99$
reset after N iterations	1.1278×10^{-3}	1.1290×10^{-3}
non-reset	7.3268×10^{-2}	7.2802×10^{-2}
non-reset Polak-Ribiere	1.1221×10^{-3}	1.1226×10^{-3}
RLS	1.1216×10^{-3}	

TABLE IV
PERFORMANCE FOR DIFFERENT IMPLEMENTATIONS OF α AND VALUES OF η. SNR = 20 dB, $\lambda_f = 0.99$

	MSE	
using α_k as in:	equation (47)	equation (49)
$\eta = 0.6$	1.1213×10^{-2}	1.1210×10^{-2}
$\eta = 0.8$	1.1221×10^{-2}	1.1225×10^{-2}
$\eta = 0.99$	1.1227×10^{-2}	unstable
RLS [13]	1.1216×10^{-2}	

TABLE V
TESTING THE CONVERGENCE CRITERION FOR DIFFERENT VALUES OF λ_f. THE ALGORITHM BECOMES UNSTABLE FOR VALUES OF η GREATER THAN OR EQUAL TO THE ONES SHOWN. SNR = 10 dB, $\sigma_x^2 = 1$

λ_f	η, equation (47)	η, equation (49)
0.99	2.0	1.0
0.9	2.0	0.99
0.8	1.8	0.87
0.7	1.53	0.77

E. Dynamic Range

As previously observed, for the computation of $\beta(n)$, only half of the dynamic range is effectively used. This also happens with the fixed-point computation of $\alpha(n)$ due to the inner product appearing in its numerator. When $\alpha(n)$ is zero due to quantization, the algorithm stops updating the weight-vector. This is known as the *stalling phenomenon* [6], [8].

Now, consider the computation of $\mathbf{R}(n)$. Rewriting (21), we can see that

$$\mathbf{R}(n) = \sum_{i=0}^{n} \lambda_f^i \mathbf{x}(n-i)\mathbf{x}(n-i)^T$$

and

$$E[\mathbf{R}(\infty)] = \sum_{i=0}^{\infty} \lambda_f^i \mathbf{R} = \frac{1}{1-\lambda_f}\mathbf{R}.$$

For values of λ_f close to one, $E[\mathbf{R}(n)]$ is large, and extra bits would be required to correctly compute $\mathbf{R}(n)$ without saturation. A normalized version given by

$$\mathbf{R}(n) = \lambda_f \mathbf{R}(n-1) + (1-\lambda_f)\mathbf{x}(n)\mathbf{x}(n)^T$$
$$\mathbf{b}(n) = \lambda_f \mathbf{b}(n-1) + (1-\lambda_f)d(n)\mathbf{x}(n)$$

is preferred in this case, and the new residual vector will become

$$\mathbf{g}(n) = \lambda_f \mathbf{g}(n-1) - \alpha(n)\mathbf{R}(n)\mathbf{p}(n) + (1-\lambda_f)\mathbf{x}(n)(d(n) - \mathbf{x}(n)^T\mathbf{w}(n-1)).$$

It has been shown in the literature that the vector $\mathbf{p}(n)$ can also be normalized, resulting in the so-called normalized CG algorithm [14], where we have

$$\mathbf{p}(n+1) = \frac{\mathbf{g}(n) + \beta(n)\mathbf{p}(n)}{1+\beta(n)}.$$

While this normalization might be useful in floating-point computation by limiting the dynamic range, it is not very effective under fixed-point computation due to the quantization effect explained previously. When the algorithm is close to convergence, $\beta(n)$ will be small and, when quantized, will become zero.

IV. SIMULATIONS

Several simulations were performed using the two basic configurations [13]: system identification (SI) and linear prediction (LP). All simulations were ensemble averaged over 100 independent trials.

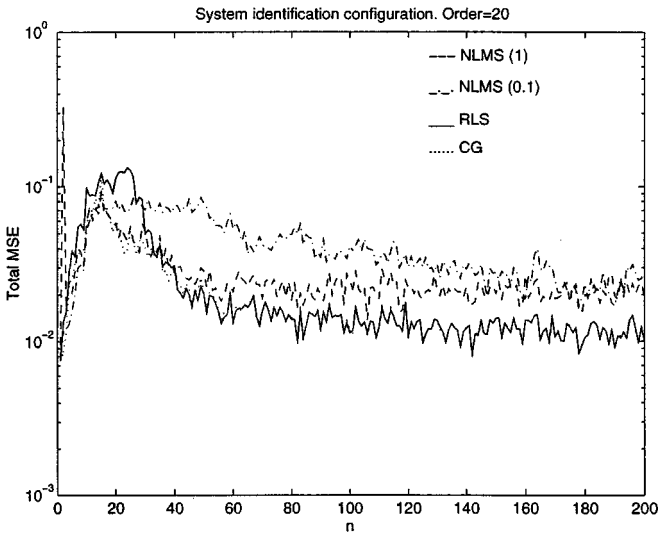

Fig. 4. Simulations using SI configuration. $\eta = \lambda_f = 0.99$, SNR = 20 dB.

First, we simulated several implementations discussed in Section II, where we considered an SI configuration with the unknown plant being an FIR filter of order 20 and the variance of the white Gaussian input signal $\sigma_x^2 = 1$. Table III compares the performance for different reset schemes. It is shown that using a non-reset scheme the algorithm performs badly due to the loss of orthogonality between $\mathbf{g}_k$ and $\mathbf{p}_k$. Table IV compares the performance for different implementations of α and values of η, showing that when orthogonality is not attained, such as in the degenerated scheme, the formulation of α in (47), given in the Appendix, is preferable to that of (49). Table V shows the validity of the convergence criterion given in the Appendix. It can be seen that this simple criterion is sufficient to guarantee the stability of the algorithm. Fig. 4 compares the performance of the RLS, the CG and the normalized LMS algorithms, where

TABLE VI
PERFORMANCE FOR DIFFERENT WINDOWING SCHEMES. SNR = 20 dB, $\sigma_x^2 = 1$ AND $\eta = 0.7$

n_w	MSE
2	1.6455×10^{-1}
5	7.6890×10^{-2}
10	2.2636×10^{-2}
15	1.5678×10^{-2}
20	1.3744×10^{-2}
25	1.2771×10^{-2}
$\lambda_f = 0.99$	1.0403×10^{-2}

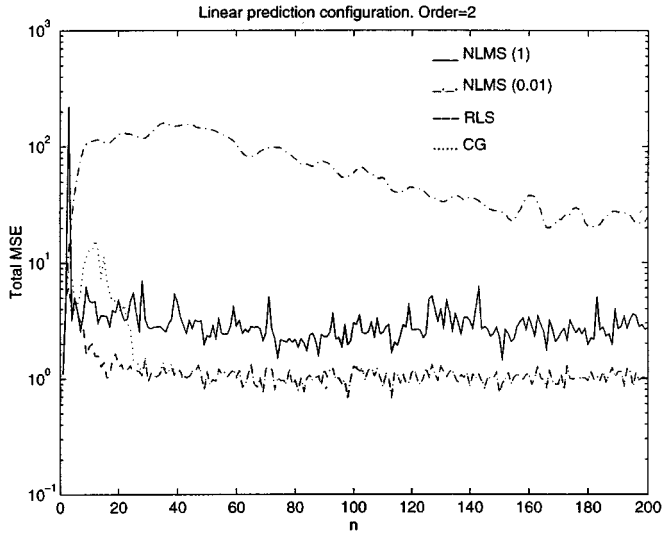

Fig. 5. Simulations using LP configuration. $\eta = \lambda_f = 0.99$, SNR = 30 dB.

$\mu_{NLMS} = 1$ and 0.1. These step sizes give the fastest convergence rate and give misadjustment comparable with the RLS algorithm in steady state, respectively. Table VI compares the MSE due to the use of various data windowing schemes. In the case shown, using the exponentially decaying data window gives a better performance result than using the finite-length data window. Here, the plant used is a fifth-order FIR filter with an eigenvalue spread of 46.

Finally, simulations were performed using the LP configuration. Fig. 5 shows the simulation results, where $\mu_{NLMS} = 1$ and 0.01, with $\eta = \lambda_f = 0.99$ and SNR = 30 dB. The second-order AR model used has an eigenvalue spread of 100. Again, $\mu_{NLMS} = 1$ gives the fastest convergence rate, whereas $\mu_{NLMS} = 0.01$ gives comparable misadjustment to that of the RLS algorithm in steady state.

V. CONCLUSION

Several modifications to the conjugate gradient algorithm for adaptive filtering have been described. The algorithms can have the same performance as some high-convergence-rate algorithms such as the RLS and LMS-Newton, with the advantage that there is no need to perform matrix inversion or to estimate $\mathbf{R}^{-1}$. It has been shown that there are several ways to implement the algorithm, leading to different results. Several simulations were carried out to illustrate the performance for different implementation choices. Two methods of data windowing were considered: the finite sliding window and the exponentially decaying window. With the first, the convergence rate is fast, but misadjustment is high. By using an exponentially decaying window, it is possible to simultaneously attain a fast convergence rate and low misadjustment. A convergence criterion has been given, which provides a sufficient condition that guarantees the stability of the algorithm.

Two approaches to the implementation of the conjugate gradient algorithm have been analyzed, and their convergence rate and misadjustment were compared. A z-domain approach was used to find the asymptotic performance, and stability bounds for $\overline{\alpha}$ and $\overline{\beta}$ were established. Finally, the behavior of the algorithms in finite word-length computation were described, and dynamic range considerations were discussed. It has been shown that close to steady state, the algorithms' behaviors are similar to the steepest descent algorithm, where the stalling phenomenon has also been observed. Using 16-bit fixed-point number representation, the simulations have shown that the algorithms are numerically stable.

APPENDIX

For the CG algorithm, a descent property given by

$$0 \le \mathbf{p}(n)^T \mathbf{g}(n) \le 0.5 \mathbf{p}(n)^T \mathbf{g}(n-1)$$

should hold in order to guarantee convergence [1]. A looser condition can be set if the following is used:

$$0 \le E[\mathbf{p}(n)^T \mathbf{g}(n)] \le 0.5 E[\mathbf{p}(n)^T \mathbf{g}(n-1)].$$

Premultiplying (23) by $\mathbf{p}(n)^T$ gives

$$\begin{aligned}\mathbf{p}(n)^T \mathbf{g}(n) = {} & \lambda_f \mathbf{p}(n)^T \mathbf{g}(n-1) \\ & - \alpha(n) \mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n) + \mathbf{p}(n)^T \mathbf{x}(n)\, d(n) \\ & - \mathbf{p}(n)^T \mathbf{x}(n) \mathbf{x}(n)^T \mathbf{w}(n-1).\end{aligned}$$

Taking the expectation of both sides and considering $\mathbf{p}(n)$ uncorrelated with $\mathbf{x}(n)$, $d(n)$, and $\mathbf{w}(n-1)$ yields

$$\begin{aligned}E[\mathbf{p}(n)^T \mathbf{g}(n)] \approx {} & \lambda_f E[\mathbf{p}(n)^T \mathbf{g}(n-1)] \\ & - E[\alpha(n)] E[\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)] \\ & - \mathbf{p}(n)^T \mathbf{R} E[\mathbf{w}(n-1) - \mathbf{w}^*] \end{aligned} \quad (45)$$

where the Wiener-Hopf equation $\mathbf{R}\mathbf{w}^* = \mathbf{b}$ [13] has been used. Assuming that the algorithm converges, the last term of (45) can be neglected, and we should have

$$E[\alpha(n)] = \frac{E[\mathbf{p}(n)^T \mathbf{g}(n)] - \lambda_f E[\mathbf{p}(n)^T \mathbf{g}(n-1)]}{E[\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)]}$$

and

$$\begin{aligned}(\lambda_f - 0.5) \frac{E[\mathbf{p}(n)^T \mathbf{g}(n-1)]}{E[\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)]} & \le E[\alpha(n)] \\ & \le \lambda_f \frac{E[\mathbf{p}(n)^T \mathbf{g}(n-1)]}{E[\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)]}. \end{aligned} \quad (46)$$

The inequalities in (46) are satisfied if we use

$$\alpha(n) = \eta \frac{\mathbf{p}(n)^T \mathbf{g}(n-1)}{\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)} \tag{47}$$

where $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$. Due to the degeneration scheme, the expanding subspace theorem [17] is not valid, and we have, after multiplying (7) at instant $(n-1)$ by $\mathbf{g}(n-1)$

$$\begin{aligned} \mathbf{p}(n)^T \mathbf{g}(n-1) = & \mathbf{g}(n-1)^T \mathbf{g}(n-1) \\ & + \beta(n-1) \mathbf{p}(n-1)^T \mathbf{g}(n-1) \end{aligned} \tag{48}$$

where the last term is not zero, due to the use of a nonconstant $\mathbf{R}$. Therefore, using

$$\alpha(n) = \eta \frac{\mathbf{g}(n-1)^T \mathbf{g}(n-1)}{\mathbf{p}(n)^T \mathbf{R}(n) \mathbf{p}(n)} \tag{49}$$

is less effective than using (47). Still, it is possible to use (49), but η must be set smaller in order to compensate for the presence of an extra term in (48).

REFERENCES

[1] M. Al-Baali, "Descent property and global convergence of the Fletcher-Reeves method with inexact line search," *IMA J. Numer. Anal.*, vol. 5, pp. 121–124, Jan. 1985.

[2] O. Axelsson, *Iterative Solution Methods*. New York: Cambridge Univ. Press, 1994.

[3] 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.

[4] T. Bose and M. Q. Chen, "Conjugate gradient method in adaptive bilinear filtering," *IEEE Trans. Signal Processing*, vol. 43, pp. 1503–1508, Jan. 1995.

[5] G. E. Bottomley and S. T. Alexander, "A novel approach for stabilizing recursive least squares filters," *IEEE Trans. Signal Processing*, vol. 39, pp. 1770–1779, Aug. 1991.

[6] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 34–41, Feb. 1984.

[7] P. S. Chang and A. N. Willson, Jr., "Adaptive filtering using modified conjugate gradient," in *Proc. 38th Midwest Symp. Circuits Syst.*, Rio de Janeiro, Brazil, Aug. 1995, pp. 243–246.

[8] ——, "A roundoff error analysis of the normalized LMS algorithm," in *Proc. 29th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Oct. 1995, pp. 1337–1341.

[9] ——, "Adaptive spectral estimation using the conjugate gradient algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, GA, May 1996, pp. 2979–2982.

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

[11] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *IEEE Trans. Signal Processing*, vol. 43, pp. 1151–1160, May 1995.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations, 2nd Ed.*. Baltimore, MD: Johns Hopkins Univ. Press, 1989.

[13] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[14] M. Hestenes, *Conjugate Direction Methods in Optimization*. New York: Springer-Verlag, 1980.

[15] A. W. Hull and W. K. Jenkins, "Preconditioned conjugate gradient methods for adaptive filtering," in *Proc. IEEE Int. Symp. Circuits Syst.*, Singapore, June 1991, pp. 540–543.

[16] K. S. Joo and T. Bose, "A fast conjugate gradient algorithm for 2-D nonlinear adaptive filtering," in *Proc. Int. Conf. Digital Signal Process.*, Limassol, Cyprus, June 1995, pp. 314–319.

[17] D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

[18] R. J. Plemmons, "FFT-based RLS in signal processing," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, Minneapolis, MN, Apr. 1993, pp. 571–574.

[19] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*. New York: Macmillan, 1988.

[20] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Math. Oper. Res.*, vol. 3, pp. 244–256, Aug. 1978.

[21] H. Stark and J. W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

Pi Sheng Chang (S'90–M'98) received the B.S. degree from Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, in 1988, the M.S. degree from Pontifical Catholic University, Rio de Janeiro, in 1990, and the Ph.D. degree from University of California, Los Angeles, in 1998, all in electrical engineering.

From 1994 to 1996, he was a Research Assistant with the Department of Neurology, University of California, Los Angeles, where he worked on eye movement studies. Since 1996, he has been with VTEL Corporation, Sunnyvale, CA, working on acoustic echo cancellation and talker location using microphone arrays. His current research interests include audio coding, adaptive signal processing and array processing with applications to acoustics, echo cancellation, microphone arrays, and spectral and DOA estimation.

Alan N. Willson, Jr. (S'66–M'67–SM'73–F'78) received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, in 1961 and the M.S. and Ph.D. degrees from Syracuse University, Syracuse, NY, in 1965 and 1967, respectively.

From 1961 to 1964, he was with IBM, Poughkeepsie, NY. He was an Instructor in electrical engineering at Syracuse University from 1965 to 1967. From 1967 to 1973, he was a Member of Technical Staff at Bell Laboratories, Murray Hill, NJ. Since 1973, he has been on the Faculty of the University of California, Los Angeles (UCLA), where he is now Professor of Engineering and Applied Science in the Electrical Engineering Department. In addition, he served the UCLA School of Engineering and Applied Science as Assistant Dean for Graduate Studies from 1977 through 1981 and is currently Associate Dean of Engineering. He has been engaged in research concerning computer-aided circuit analysis and design, the stability of distributed circuits, properties of nonlinear networks, theory of active circuits, digital signal processing, analog circuit fault diagnosis, and integrated circuits for signal processing. He is editor of the book *Nonlinear Networks: Theory and Analysis* (New York: IEEE, 1974).

Dr. Willson is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, the Society for Industrial and Applied Mathematics, and the American Society for Engineering Education. From 1977 to 1979, he served as Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. In 1980, he was General Chairman of the 14th Asilomar Conference on Circuits, Systems, and Computers. During 1984, he served as President of the IEEE Circuits and Systems Society. He was the recipient of the 1978 and 1994 Guillemin–Cauer Awards of the IEEE Circuits and Systems Society, the 1982 George Westinghouse Award of the American Society for Engineering Education, the 1982 Distinguished Faculty Award of the UCLA Engineering Alumni Association, the 1984 Myril B. Reed Best Paper Award of the Midwest Symposium on Circuits and Systems, and the 1985 and 1994 W. R. G. Baker Awards of the IEEE.