

ANALYSIS OF CONJUGATE GRADIENT ALGORITHMS FOR ADAPTIVE FILTERING

Pi Sheng Chang and Alan N. Willson, Jr.

Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90095-1600
e-mail: pschang@ee.ucla.edu, willson@ee.ucla.edu

ABSTRACT

We describe and analyze two approaches to the implementation of the Conjugate Gradient algorithm for adaptive filtering. In particular, their convergence rate and misadjustment are compared. A new analysis approach 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, close to steady-state, the algorithms' behaviors are similar to the Steepest Descent algorithm, where the stalling phenomenon is also observed. Using 16-bit fixed-point number representation, our simulations show that the algorithms are numerically stable.

1. INTRODUCTION

In recent years, many adaptive filtering algorithms based on the Conjugate Gradient (CG) method of optimization have been reported [3, 4, 6, 8, 9, 13]. 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 the Steepest Descent method [1, 10], and it also has lower computational complexity when compared to the classic recursive least squares (RLS) algorithm [3]. But most of the analyses of the CG algorithm can only be found in the optimization and matrix computation literature. Here we will describe two of the CG algorithms that have been proposed in [3, 6], from the signal processing point of view, and we will obtain estimates of their asymptotic performance. Also, their performance under finite word-length effects will be discussed. It will be shown that, due to the highly nonlinear nature of the algorithms, a linearized quantization model, in general, cannot be applied, as has been used for the LMS [5], NLMS [7] or RLS [2] algorithms.

We present the analysis of two approaches to the implementation of the CG algorithm in adaptive filtering. 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 iterations, and N or fewer 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 a certain fixed \mathbf{R} [1, 10]. This is used in algorithm CG2, de-

scribed here and also in [3]. The advantage of this approach is that the convergence rate is independent of the eigenvalue spread of \mathbf{R} , while 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 CG1 algorithm and also in [6], both finite data windows and exponentially decaying data windows can be used, although an exponentially decaying data window gives better performance due to the better estimation of \mathbf{R} and \mathbf{b} that results.

2. THE CG ALGORITHMS

Algorithm CG1: The CG algorithm using the first approach can be found in [6, 13]. In [6], the algorithm minimizes a cost function defined as $f(\mathbf{w}) = \mathbf{w}^T \mathbf{R} \mathbf{w} / 2 + \mathbf{b}^T \mathbf{w}$. It is described by the following equations:

Set initial conditions: $\mathbf{w}_0 = 0$, $\mathbf{g}_0 = \mathbf{b}_0$, $\mathbf{p}_1 = \mathbf{g}_0$, $n = 1$.

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

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \alpha(n) \mathbf{p}(n) \quad (2)$$

$$\mathbf{g}(n) = \lambda_f \mathbf{g}(n-1) - \alpha(n) \mathbf{R}(n) \mathbf{p}(n) + \mathbf{x}(n)(d(n) - \mathbf{x}(n)^T \mathbf{w}(n-1)) \quad (3)$$

$$\beta(n) = \frac{(\mathbf{g}(n) - \mathbf{g}(n-1))^T \mathbf{g}(n)}{\mathbf{g}(n-1)^T \mathbf{g}(n-1)} \quad (4)$$

$$\mathbf{p}(n+1) = \mathbf{g}(n) + \beta(n) \mathbf{p}(n) \quad (5)$$

where $d(n)$ is the desired response, $\alpha(n)$ is the step size that minimizes a cost function $f(\mathbf{w})$, $\beta(n)$ provides quasi- \mathbf{R} -conjugacy for the direction vector $\mathbf{p}(n)$, and $\mathbf{g}(n)$ is the residual vector defined as $\mathbf{g}(n) = -\nabla f(\mathbf{w})^T$. $\mathbf{R}(n)$ is the $N \times N$ sample correlation matrix of the input data vector $\mathbf{x}(n)$, and it is computed as

$$\mathbf{R}(n) = \lambda_f \mathbf{R}(n-1) + \mathbf{x}(n) \mathbf{x}(n)^T \quad (6)$$

where λ_f is the forgetting factor of the exponentially decaying data window. The parameter η in (1) controls the convergence of the algorithm and must be set within the range $(\lambda_f - 0.5) \leq \eta \leq \lambda_f$ [6].

In state-space notation, the algorithm can be written as

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

*This work was supported by the National Science Foundation under Grant MIP-9632698.

Algorithm CG2: Following the same approach used in [3, 6]¹, the next CG algorithm can be described as follows:

Set initial cond.: $\mathbf{w}(0) = 0$. For each instant n compute:
Start:

$$\tilde{\mathbf{R}}(n) = \frac{1}{M} \sum_{i=0}^{M-1} \mathbf{x}(n-i)\mathbf{x}(n-i)^T \quad (7)$$

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

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

for $k = 1$ to N do:

$$\alpha(k) = \frac{\mathbf{g}(k-1)^T \mathbf{g}(k-1)}{\mathbf{p}(k)^T \tilde{\mathbf{R}}(n) \mathbf{p}(k)} \quad (8)$$

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \alpha(k) \mathbf{p}(k) \quad (9)$$

$$\mathbf{g}(k) = \mathbf{g}(k-1) - \alpha(k) \tilde{\mathbf{R}}(n) \mathbf{p}(k) \quad (10)$$

$$\beta(k) = \frac{\mathbf{g}(k)^T \mathbf{g}(k)}{\mathbf{g}(k-1)^T \mathbf{g}(k-1)} \quad (11)$$

$$\mathbf{p}(k+1) = \mathbf{g}(k) + \beta(k) \mathbf{p}(k) \quad (12)$$

After N iterations,

$$\begin{aligned} \mathbf{w}(0) &= \tilde{\mathbf{w}}(n) = \mathbf{w}(N) \\ n &= n+1 \end{aligned}$$

goto Start

Notice that $\tilde{\mathbf{R}}(n)$ is fixed throughout the k iterations and only the final vector $\mathbf{w}(N)$ is of interest.

In state-space notation, CG2 can be written as

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

3. CG ALGORITHM IN SIGNAL-FLOW GRAPH REPRESENTATION AND ASYMPTOTIC ANALYSIS

Using the state-space representation given above, we can view the CG algorithms as nonlinear time-varying digital filters. Consider the CG1 algorithm and, in order to simplify its analysis, let's assume that $E(\alpha(n)) = \bar{\alpha}$, $E(\beta(n)) = \bar{\beta}$, $E(\mathbf{b}(n)) = \mathbf{b}$ and $E(\mathbf{R}(n)) = \mathbf{R}$ are constants. Then we can view the system as linear, time-invariant. Furthermore, let's define $\mathbf{W} = \mathcal{Z}\{E(\mathbf{w}(n))\}$, $\mathbf{G} = \mathcal{Z}\{E(\mathbf{g}(n))\}$, $\mathbf{P} = \mathcal{Z}\{E(\mathbf{p}(n))\}$ and note that (3) can also be written as

$$\mathbf{g}(n) = \mathbf{b}(n)u(n) - \mathbf{R}(n)\mathbf{w}(n). \quad (13)$$

Now we can find the transfer function for \mathbf{W} using (2), (5) and (13). The signal-flow graph representation of these three equations is shown in Fig. 1 where we have, after

¹The 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.

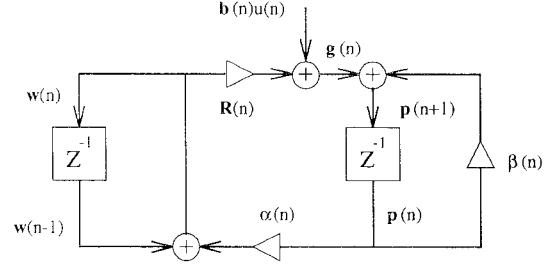


Fig. 1. Signal-flow graph representation of the CG algorithm.

taking the expectation on both sides of the equations and then the z -transform:

$$\mathbf{W} = \mathbf{W}z^{-1} + \bar{\alpha}\mathbf{P} \quad (14)$$

$$z\mathbf{P} = \bar{\beta}\mathbf{P} + \mathbf{G} \quad (15)$$

$$\mathbf{G} = \frac{\mathbf{b}z}{z-1} - \mathbf{R}\mathbf{W}. \quad (16)$$

Solving for \mathbf{W} , we get

$$\mathbf{W} = \left(\frac{(z-1)(z-\bar{\beta})\mathbf{I} + \bar{\alpha}\mathbf{R}z}{z(z-\bar{\beta})} \right)^{-1} \frac{\bar{\alpha}\mathbf{b}z}{(z-\bar{\beta})(z-1)}. \quad (17)$$

Now, since $\mathbf{W}(z) = \mathbf{W}^+(z)$, the one-sided z -transform, we can use the *Final Value Theorem* [14], resulting in

$$\lim_{n \rightarrow \infty} E(\mathbf{w}(n)) = \lim_{z \rightarrow 1} (z-1)\mathbf{W}^+(z) = \mathbf{R}^{-1}\mathbf{b}. \quad (18)$$

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

$$\det((z-1)(z-\bar{\beta})\mathbf{I} + \bar{\alpha}\mathbf{R}z) = 0 \quad (19)$$

must lie inside the unit circle. (18) shows that $E(\mathbf{w}(n))$ will converge to \mathbf{w}_o for $n \rightarrow \infty$, where \mathbf{w}_o is the optimum weight vector. We can apply a unitary transformation to \mathbf{R} so that $\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ and, knowing that $\det(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T) = \det(\mathbf{\Lambda})$, where $\mathbf{\Lambda}$ is a diagonal matrix whose elements λ_i are the eigenvalues of \mathbf{R} , (19) becomes

$$\prod_{i=0}^N (z^2 + (\bar{\alpha}\lambda_i - (\bar{\beta} + 1))z + \bar{\beta}) = 0. \quad (20)$$

If $\bar{\beta} = 0$, we have $z + \bar{\alpha}\lambda_i - 1 = 0$ and $2 > \bar{\alpha}\lambda_i > 0$. A sufficient condition for the stability of the system is

$$\frac{2N}{\text{tr}\mathbf{R}} > \bar{\alpha} > 0.$$

For $0 < \bar{\beta} < 1$, from the placement of the zeros of a second order polynomial equation, we have

$$\bar{\alpha}\lambda_i - (\bar{\beta} + 1) = -2\sqrt{\bar{\beta}} \cos \theta. \quad (21)$$

Then

$$\begin{aligned} -2\sqrt{\bar{\beta}} &< \bar{\alpha}\lambda_i - (\bar{\beta} + 1) < 2\sqrt{\bar{\beta}} \\ (\sqrt{\bar{\beta}} - 1)^2 &< \bar{\alpha}\lambda_i < (\sqrt{\bar{\beta}} + 1)^2 \end{aligned} \quad (22)$$

and a sufficient condition for the stability of the system is

$$\frac{N(\sqrt{\bar{\beta}} - 1)^2}{\text{tr}\mathbf{R}} < \bar{\alpha} < \frac{N(\sqrt{\bar{\beta}} + 1)^2}{\text{tr}\mathbf{R}}. \quad (23)$$

For small values of $\tilde{\beta}$, we have

$$\frac{N}{tr\mathbf{R}} - \epsilon < \bar{\alpha} < \frac{N}{tr\mathbf{R}} + \epsilon$$

where ϵ is a small number, and for $\tilde{\beta} \rightarrow 1$, we have

$$0 < \bar{\alpha} < \frac{4N}{tr\mathbf{R}}.$$

For the CG2 algorithm, since $\tilde{\mathbf{R}}(n)$ and $\tilde{\mathbf{b}}(n)$ are constant throughout the k iterations, the analysis presented in [1, 10] can be applied.

3.1. Convergence and Misadjustment

Consider the algorithm described in [3] and CG2, where a finite-length block of data is used. It has been shown in [3] that the convergence rate doesn't depend on the eigenvalue spread and that the final misadjustment is inversely proportional to the data window length. The conclusion about the convergence rate at first may seem to disagree with what is shown in [1, 10], but one must take into consideration the way the CG algorithm was implemented in [3], where the updated weight vector at each time instant n is the last updated weight vector after k iterations (usually $k = N$), so we have

$$\tilde{\mathbf{w}}(n) = \tilde{\mathbf{R}}(n)^{-1}\tilde{\mathbf{b}}(n). \quad (24)$$

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 . For CG1, where an exponentially decaying data window is used, and the updated weight vector is obtained as a result of a single iteration, the convergence will depend on the eigenvalue spread. Using variable \mathbf{R} reduces substantially the computational complexity of the algorithm. In steady-state, as $\mathbf{R}(n) \rightarrow \mathbf{R}$, the misadjustment for CG1 will be equal to the misadjustment of the RLS algorithm, since both algorithms minimize the same cost function [11].

Now, consider CG2 and the System Identification (SI) configuration described in [11, 14]. The desired response is the output of the FIR filter with optimum weight coefficients $d(n) = \mathbf{x}(n)^T \mathbf{w}_o$ (with no measurement noise). The output mean-squared-error is given by

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

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

where $\mathbf{K}(n) = E(\epsilon(n)\epsilon(n)^T)$ [11]. For the SI configuration with WGN as the input signal, $\mathbf{R} = \sigma_x^2 \mathbf{I}$ and we have

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

When using $\tilde{\mathbf{R}}(n)$ to estimate \mathbf{R} , we have to consider the variance of the estimation, and it can be shown that $var(\tilde{r}_{ij}(n)) \approx m_4/M$ [15], where $m_4 = E[x_i^4]$ is the 4th-order moment of $x_i(n)$ and $\tilde{r}_{ij}(n)$ is a element of $\tilde{\mathbf{R}}(n)$. For Gaussian input signals, the kurtosis of the signal, $\nu_x = E[x_i^4]/\sigma_x^4$, is 3, so we have

$$var(\tilde{r}_{ij}(n)) = 3 \frac{\sigma_x^4}{M}. \quad (25)$$

Table 1. Performance for various lengths of data windowing. $\sigma_x^2 = 0.25$, $N = 5$, $\|\mathbf{w}_o\|^2 = 1.3071$ 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

Using the same reasoning, we have

$$var(\tilde{b}_i(n)) = 3 \frac{\sigma_x^4}{M} \|\mathbf{w}_o\|^2. \quad (26)$$

Referring to (24), we can use a gross approximation and consider that

$$\sigma_x^2 E(\|\epsilon(n)\|^2) \approx \sigma_x^2 var(\tilde{r}_{ij}(n)) var(\tilde{b}_i(n))$$

and

$$E[e(n)^2] \approx \frac{9\sigma_x^{10}}{M^2} \|\mathbf{w}_o\|^2.$$

This shows the dependence of the misadjustment on the length M , as has been shown in the simulation results in [3]. Table 1 shows the performance of CG2 for various values of M .

4. 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. This is due to the fact that quantizing the variables in the CG algorithm will lead some of them to 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 (4), in fixed-point computation. In order to be able to update $\mathbf{p}(n+1)$, it is necessary to have

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

$$\begin{aligned} Q[(g(n) - g(n-1))^T g(n)] &\geq 2^{-B-1} \\ \sum Q[(g_i(n) - g_i(n-1))g_i(n)] &\geq 2^{-B-1} \end{aligned}$$

where B is the number of bits used to represent the fractional part of a number in fixed-point notation (see [5, 7]) and $Q[\cdot]$ is the quantization operation. This implies that each element of $\mathbf{g}(n)$, $g_i(n)$, must satisfy

$$g_i(n) \geq 2^{(-B-1)/2} \quad (27)$$

i.e., 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. 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 $\lambda_f = 0.99$, $\eta = 0.9$, $N = 20$, $\sigma_x^2 = 0.1$, SNR=30 dB, using 10 bits for the fractional part and six bits for the integer part of the number representation.

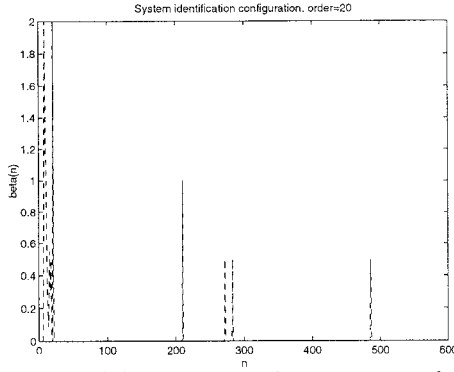


Fig. 2. $\beta(n)$ in fixed-point computation.

4.1. Dynamic Range

As we saw previously, for the computation of $\beta(n)$ only half of the dynamic range is effectively used. This also happens with the computation of $\alpha(n)$ in fixed-point, 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* [5, 7].

Now, consider the computation of $\mathbf{R}(n)$. Rewriting (6), 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 we would require extra bits to correctly compute $\mathbf{R}(n)$ without saturation. A normalized

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

and

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

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

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

$$+ (1 - \lambda_f) \mathbf{x}(n) (d(n) - \mathbf{x}(n)^T \mathbf{w}(n-1)).$$

The vector $\mathbf{p}(n)$ can also be normalized, resulting in the so-called Normalized CG algorithm [12], where we have

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

Note that this normalization is not very effective under fixed-point computation because of the quantization effect explained previously. When the algorithm is close to convergence $\beta(n)$ will be small and, when quantized, will become zero.

5. CONCLUSION

In this paper, we have considered two approaches to the implementation of the Conjugate Gradient algorithm for adaptive filtering. These two approaches have been presented previously in [3, 6]. Here, their convergence rate and misadjustment were compared. A new z -domain approach was used to find the asymptotic performance, and stability bounds for $\bar{\alpha}$ and $\bar{\beta}$ were established. Finally, the behavior of the algorithms in finite word-length computation were described and dynamic range considerations were

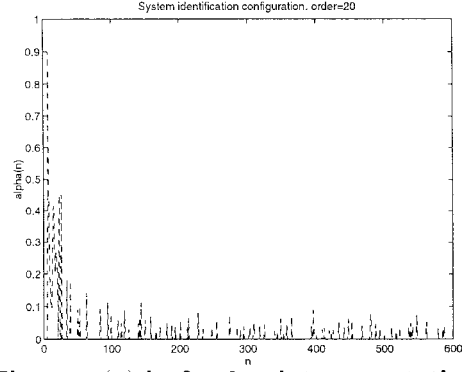


Fig. 3. $\alpha(n)$ in fixed-point computation.

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.

REFERENCES

- [1] O. Axelsson, *Iterative Solution Methods*. New York: Cambridge Univ. Press, 1994.
- [2] 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.
- [3] G. K. Boray and M. D. Srinath, "Conjugate gradient techniques for adaptive filtering," *IEEE Trans. on 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, Jun. 1995.
- [5] C. Caraiscos and B. Liu, "A roundoff error analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 34-41, Feb. 1984.
- [6] P. S. Chang and A. N. Willson, Jr., "Adaptive filtering using modified conjugate gradient," *Proc. 38th Midwest Symp. on Circuits and Systems*, Rio de Janeiro, pp. 243-246, Aug. 1995.
- [7] P. S. Chang and A. N. Willson, Jr., "A roundoff error analysis of the normalized LMS algorithm," *Proc. 29th Asilomar Conf. on Signals, Systems and Comp.*, Pacific Grove, pp. 1337-1341, Oct. 1995.
- [8] P. S. Chang and A. N. Willson, Jr., "Adaptive spectral estimation using the conjugate gradient algorithm," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Atlanta, pp. 2979-2982, May 1996.
- [9] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *IEEE Trans. on Signal Processing*, vol. 43, pp. 1151-1160, May 1995.
- [10] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore: Johns Hopkins University Press, 1989, 2nd ed.
- [11] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1991, 2nd ed.
- [12] M. Hestenes, *Conjugate Direction Methods in Optimization*. New York: Springer-Verlag, 1980.
- [13] K. S. Joo and T. Bose, "A fast conjugate gradient algorithm for 2-D nonlinear adaptive filtering," *Proc. Int. Conf. on Digital Signal Processing*, Limassol, pp. 314-319, Jun. 1995.
- [14] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*. New York: Macmillan, 1988.
- [15] H. Stark and J. W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers*. Englewood Cliffs, NJ: Prentice-Hall, 1986.