ADAPTIVE FILTERING USING MODIFIED CONJUGATE GRADIENT [*]

Pi Sheng Chang and Alan N. Willson, Jr.

Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90024
e-mail: pschang@ee.ucla.edu, willson@ee.ucla.edu

ABSTRACT

An adaptive filtering algorithm is described that uses the modified Conjugate Gradient (CG) algorithm. It has the ability to perform sample-by-sample updating of the filter coefficients more efficiently than previously described CG methods. Its performance can be comparable to the RLS and LMS-Newton algorithms, giving fast convergence for highly correlated input signals, while maintaining low misadjustment. Simulations demonstrating its performance and the influence of various parameter choices are shown. A convergence criterion is also derived.

1. INTRODUCTION

The Conjugate Gradient (CG) Method can be applied to adaptive transversal filters as shown in [1, 2]. Doing this, the objective becomes the solving of the following equation:

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

where $\mathbf{R}$ is the $N \times N$ correlation matrix of the input data vector $\mathbf{x}_k$ and $\mathbf{b}$ is the cross-correlation vector between the input data and the desired response d_k. If $\mathbf{R}$ and $\mathbf{b}$ are defined as in [4] for the LS problem, the CG method offers an alternative way to solve for $\mathbf{w}$, instead of inverting matrix $\mathbf{R}$. If they are defined as in [1], where a sliding 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 sample. While with previously developed CG algorithms this can be done at the expense of running several iterations per sample, we here propose modifications that will allow the algorithm to run just one iteration per sample while still maintaining performance comparable to RLS or LMS-Newton. One of the main problems that RLS and LMS-Newton have is the necessity to estimate $\mathbf{R}^{-1}$. If the estimated $\mathbf{R}^{-1}$ loses the property of positive definiteness, it will cause the divergence of the algorithm [4]; this doesn't happen with the CG method.

[*]This work was supported in part by CNPq, Brazil, and in part by the U.S. National Science Foundation under Grant MIP 92-01104.

1.1 THE BASIC CONJUGATE GRADIENT METHOD

The basic Conjugate Gradient Algorithm can be stated as follows [5, 7], after some rearrangement for better clarity:

```
$\mathbf{w}_0 = 0,\ \mathbf{g}_0 = \mathbf{b},\ \rho_0 = \mathbf{g}_0^T\mathbf{g}_0,\ \mathbf{p}_1 = \mathbf{g}_0,\ k = 1$
while $k \le k_{max}$
  begin
        $\mathbf{v} = \mathbf{R}\mathbf{p}_k$
        $\alpha_k = \rho_{k-1}/\mathbf{p}_k^T\mathbf{v}$                (2)
        $\mathbf{w}_k = \mathbf{w}_{k-1} + \alpha_k\mathbf{p}_k$          (3)
        $\mathbf{g}_k = \mathbf{g}_{k-1} - \alpha_k\mathbf{v}$            (4)
        $\rho_k = \mathbf{g}_k^T\mathbf{g}_k$
        $\beta_k = \rho_k/\rho_{k-1}$                    (5)
        $\mathbf{p}_{k+1} = \mathbf{g}_k + \beta_k\mathbf{p}_k$           (6)
        $k = k + 1$
  end
```

where α_k is a step size that minimizes a cost function $f(\mathbf{w})$, β_k provides $\mathbf{R}$-conjugacy for the direction vector $\mathbf{p}_k$, and $\mathbf{g}_k$ is the residual vector defined as $\mathbf{g}_k = \mathbf{b} - \mathbf{R}\mathbf{w}_k = -\nabla f(\mathbf{w})^T$.

Variations of this algorithm are presented in [5, 7] where it is shown that one can use an iterative method to terminate the algorithm, instead of the fixed k_{max} iterations, or use different ways to compute α and β. The existence of only one matrix-vector multiplication is possible due to the use of a recursive formulation for the residual [2, 7]:

$$\mathbf{g}_k = \mathbf{g}_{k-1} - \alpha_k\mathbf{R}\mathbf{p}_k. \tag{7}$$

This formulation assumes that $\mathbf{R}$ and $\mathbf{b}$ are constants throughout the k_{max} iterations. It is applicable for the case of block processing and has been used in [2, 3]. Here we propose modifications that allow it to be used in non-block processing or sample-by-sample update.

1.2 CONSIDERATIONS ABOUT THE COST FUNCTION

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

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{R}\mathbf{w} - \mathbf{b}^T\mathbf{w}. \tag{8}$$

The way $\mathbf{R}$ is defined will directly influence the performance of the algorithm. There are two ways that one can compute $\mathbf{R}$ by using different schemes of data windowing, namely:

1) *Finite sliding* data window: Where only the data inside a window of finite length n_w are used. The correlation matrix and the cross-correlation vector are defined as

$$\mathbf{R}_k = \frac{1}{n_w} \sum_{j=k-n_w+1}^{k} \mathbf{x}_j \mathbf{x}_j^T \tag{9}$$

$$\mathbf{b}_k = \frac{1}{n_w} \sum_{j=k-n_w+1}^{k} d_j \mathbf{x}_j. \tag{10}$$

The residual is then computed as

$$\mathbf{g}_k = \mathbf{b}_k - \mathbf{R}_k \mathbf{w}_k \tag{11}$$

$$= \frac{1}{n_w} \sum_{j=k-n_w+1}^{k} [(d_j - \mathbf{w}_j^T \mathbf{x}_j)\mathbf{x}_j]. \tag{12}$$

The formulation in (12) is computationally more efficient if n_w is smaller than N, the length of $\mathbf{x}_k$.

2) *Exponentially decaying* data window: This gives the same correlation function as is used by the RLS algorithm. Using it with the CG algorithm, a performance comparable to RLS can be achieved. The correlation and cross-correlation functions are given by

$$\mathbf{R}_k = \lambda \mathbf{R}_{k-1} + \mathbf{x}_k \mathbf{x}_k^T \tag{13}$$

$$\mathbf{b}_k = \lambda \mathbf{b}_{k-1} + d_k \mathbf{x}_k \tag{14}$$

where λ is the forgetting factor. For sample-by-sample processing, a recursive formulation for the residual can be found by using (13), (14) and (3), resulting in

$$\mathbf{g}_k = \mathbf{b}_k - \mathbf{R}_k \mathbf{w}_k$$

$$= \lambda \mathbf{g}_{k-1} - \alpha_k \mathbf{R}_k \mathbf{p}_k + \mathbf{x}_k (d_k - \mathbf{x}_k^T \mathbf{w}_{k-1}). \tag{15}$$

1.3 TERMINATION

There are many schemes proposed in the literature to terminate the CG algorithm. In [7] an iterative scheme was proposed, based in the norm of the residual and a maximum number of iterations. In [1], the algorithm terminates after $\min(N, n_w)$ iterations, due to the use of the finite sliding data window in the computation of the residual. Either way, the CG algorithm has to run several times in order to converge. This is not a problem when block processing is used, but in sample-by-sample updating this procedure is computationally costly. One way to have just one iteration per coefficient update is to use some *degenerated* scheme. By degeneration we mean that $\mathbf{g}_k$ is not completely orthogonal to the subspace spanned by $\{\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_k\}$ or, in other words, $\mathbf{g}_k^T \mathbf{p}_i = 0, i < k$, does not hold. Another interpretation is to consider that using α given in (2), the algorithm has the property that α minimizes the cost function $f(\mathbf{w})$ on the line $\mathbf{w}_{k-1} + \alpha \mathbf{p}_k$. Using a constant value for α creates a degenerated CG since the cost function $f(\mathbf{w})$ is not completely minimized. Another example of a degenerated scheme is to use a non-constant matrix $\mathbf{R}$ at each iteration. While it degenerates the algorithm, it also allows the algorithm to be used in a non-block adaptation scheme. The new update of the residual is given by (15).

TABLE I. The modified CG algorithm.

$$\boxed{\begin{gathered}
\mathbf{w}_0 = 0,\ \mathbf{g}_0 = \mathbf{b}_0,\ \mathbf{p}_1 = \mathbf{g}_0,\ k = 1 \\[1ex]
\alpha_k = \eta \frac{\mathbf{p}_k^T \mathbf{g}_{k-1}}{\mathbf{p}_k^T \mathbf{R}_k \mathbf{p}_k} \\
\mathbf{w}_k = \mathbf{w}_{k-1} + \alpha_k \mathbf{p}_k \\
\mathbf{g}_k = \lambda \mathbf{g}_{k-1} - \alpha_k \mathbf{R}_k \mathbf{p}_k + \mathbf{x}_k (d_k - \mathbf{x}_k^T \mathbf{w}_{k-1}) \\
\beta_k = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \\
\mathbf{p}_{k+1} = \mathbf{g}_k + \beta_k \mathbf{p}_k
\end{gathered}}$$

1.4 LINE SEARCH

In the CG algorithm, α is a step size used in the update of the weight vector as shown in (3). It is usually chosen so that $f(\mathbf{w}_{k-1} + \alpha \mathbf{p}_k)$ is minimized. Explicitly computing α for the cost function (8) results in (2). This is an exact line search along the direction $\mathbf{p}_k$. Other inexact line search schemes can be used, but they have to satisfy the convergence bound given by (see Appendix)

$$\alpha_k = \eta \frac{\mathbf{p}_k^T \mathbf{g}_{k-1}}{\mathbf{p}_k^T \mathbf{R}_k \mathbf{p}_k}, \quad (\lambda - 0.5) \le \eta \le \lambda. \tag{16}$$

In [1], a method is suggested called "Fast CG," where α is a constant value. Depending on the value used, convergence might not be guaranteed. Also, using $\mathbf{g}_k^T \mathbf{g}_k$ instead of $\mathbf{p}_k^T \mathbf{g}_{k-1}$ as shown in (2) is less effective for the degenerated scheme (see Appendix).

1.5 RESETING THE ALGORITHM

For sample-by-sample processing, it's important to reset the direction vector $\mathbf{p}_k$ to the pure gradient periodically, in order to ensure the convergence of the algorithm, because the degenerated scheme will not allow the algorithm to converge in N steps. How often it is reset will influence the performance of the algorithm. If taking direction $\mathbf{p}_k$ does not increase the cost function, then global convergence can be assured since a pure steepest descent step is taken every time the algorithm is reset. A non-reset method can also be used, but the Polak-Ribiere method [5, 6] for the computation of β, given by

$$\beta_k = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{17}$$

should be used for better performance. Simulations have shown that (17) performs better than (5) when using a degenerated scheme. Table I shows an implementation of the algorithm taking into account some of the considerations discussed.

TABLE II. The CG algorithm using different implementations of β and reset schemes. $\lambda = 0.99$, SNR= 30dB, $\sigma_x^2 = 1$.

	MSE	
CG algo. with:	$\eta = 0.8$	$\eta = 0.99$
reset after N iter.	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 III. Performance for different implementations of α and values of η. SNR= 20dB, $\lambda = 0.99$.

	MSE	
using α_k as in	eq. (20)	eq. (22)
$\eta = 0.6$	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 [4]	1.1216×10^{-2}	

2. SIMULATIONS

Several simulations were performed using two basic configurations [4]: System Identification (SI) and Linear Prediction (LP). All simulations were ensemble averaged over 100 independent trials. First we simulated sevaral implementations discussed previously. Consider an SI configuration where the unknown plant is an FIR filter of order 20, with $\sigma_x^2 = 1$. Table II 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 III 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 (20) is preferable to that of (22). Table IV shows the validity of the convergence criterion. It is shown that this simple criterion is sufficient to guarantee the stability of the algorithm. Table V compares the MSE due to the use of various data windowing schemes, showing that using the exponentially decaying data window gives much better performance than using the finite-length data window. The plant is a 5th order FIR filter with eigenvalue spread of 46. Fig. 1 compares the performance of RLS, CG and Normalized LMS, where $\mu_{NLMS} = 1$ and 0.1. These step sizes give the fastest convergence rate and comparable misadjustment for the NLMS in steady-state, respectively. Finally, simulations were performed using the LP configuration. Fig. 2 shows the simulation results, where $\mu_{NLMS} = 1$ and 0.01, $\eta = \lambda = 0.99$ and SNR= 30dB. The second-order AR model used has an eigenvalue spread of 100. Again, $\mu = 1$ gives the fastest convergence rate while $\mu = 0.01$ gives comparable misadjustment in steady-state.

3. CONCLUSION

A modification to the Conjugate Gradient algorithm for

TABLE IV. Testing the convergence criterion for different values of λ. The algorithm becomes unstable for values of η equal or greater than the ones shown. SNR= 10dB, $\sigma_x^2 = 1$.

	MSE	
λ	η, eq. (20)	η, eq. (22)
0.99	2.0	1.0
0.9	2.0	0.99
0.8	1.8	0.87
0.7	1.53	0.77

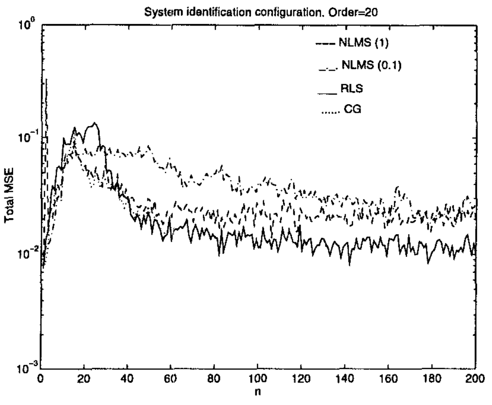

Fig. 1. Simulations using SI configuration. η=λ=0.99, SNR= 20dB.

TABLE V. Performance for different windowing schemes. SNR= 20dB, $\sigma_x^2 = 1$ 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}
$\lambda = 0.99$	1.0403×10^{-2}

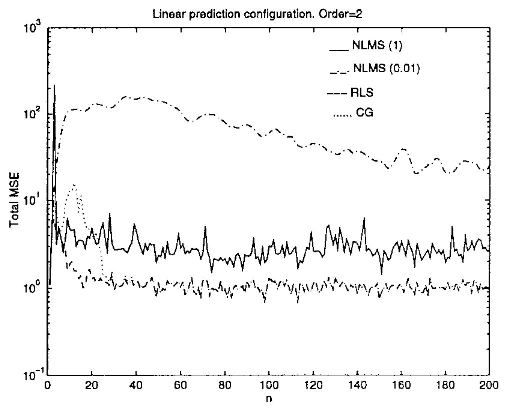

Fig. 2. Simulations using LP configuration. η=λ=0.99, SNR= 30dB.

adaptive filtering is described. This algorithm 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 due to different implementation choices. Two methods of data windowing are considered: the finite sliding window and the exponentially decaying window. With the first, the convergenge rate is fast, but misadjustment is high. By using an exponentially decaying window it is possible to attain fast convergence and low misadjustment at the same time. A convergence criterion is given, which provides a sufficient condition that guarantees the stability of the algorithm.

APPENDIX

For the CG algorithm, a descent property given by $0 \leq \mathbf{p}_k^T\mathbf{g}_k \leq 0.5\ \mathbf{p}_k^T\mathbf{g}_{k-1}$ should hold in order to guarantee convergence [8]. A looser condition can be set if the following is used:

$$0 \leq E[\mathbf{p}_k^T\mathbf{g}_k] \leq 0.5E[\mathbf{p}_k^T\mathbf{g}_{k-1}]. \tag{18}$$

Multiplying (15) by $\mathbf{p}_k^T$ gives

$$\mathbf{p}_k^T\mathbf{g}_k = \lambda\mathbf{p}_k^T\mathbf{g}_{k-1} - \alpha_k\mathbf{p}_k^T\mathbf{R}_k\mathbf{p}_k + \mathbf{p}_k^T\mathbf{x}_k d_k - \mathbf{p}_k^T\mathbf{x}_k\mathbf{x}_k^T\mathbf{w}_{k-1}.$$

Taking the expectation of both sides and considering $\mathbf{p}_k$ uncorrelated with $\mathbf{x}_k$, d_k and $\mathbf{w}_{k-1}$, we have

$$E[\mathbf{p}_k^T\mathbf{g}_k] \approx \lambda E[\mathbf{p}_k^T\mathbf{g}_{k-1}] - E[\alpha_k]E[\mathbf{p}_k^T\mathbf{R}\mathbf{p}_k]$$

$$-E[\mathbf{p}_k^T\mathbf{R}(\mathbf{w}_{k-1} - \mathbf{w}^*)]$$

where the Wiener-Hopf equation [4] has been used. Assuming that the algorithm converges, the last term can be neglected and we should have

$$(\lambda - 0.5)\frac{E[\mathbf{p}_k^T\mathbf{g}_{k-1}]}{E[\mathbf{p}_k^T\mathbf{R}\mathbf{p}_k]} \leq E[\alpha_k] \leq \lambda\frac{E[\mathbf{p}_k^T\mathbf{g}_{k-1}]}{E[\mathbf{p}_k^T\mathbf{R}\mathbf{p}_k]}. \tag{19}$$

A sufficient condition that satisfies (19) is to use

$$\alpha_k = \eta\frac{\mathbf{p}_k^T\mathbf{g}_{k-1}}{\mathbf{p}_k^T\mathbf{R}_k\mathbf{p}_k} \tag{20}$$

where $(\lambda - 0.5) \leq \eta \leq \lambda$. Due to the degeneration scheme, the Expanding Subspace Theorem [5] is not valid and we have, after multiplying (6) in instant $k-1$ by $\mathbf{g}_{k-1}$:

$$\mathbf{p}_k^T\mathbf{g}_{k-1} = \mathbf{g}_{k-1}^T\mathbf{g}_{k-1} + \beta_{k-1}\mathbf{p}_{k-1}^T\mathbf{g}_{k-1} \tag{21}$$

where the last term is not zero due to the use of a non-constant $\mathbf{R}$. So, using

$$\alpha_k = \eta\frac{\mathbf{g}_{k-1}^T\mathbf{g}_{k-1}}{\mathbf{p}_k^T\mathbf{R}_k\mathbf{p}_k} \tag{22}$$

is less effective than using (20). Still, it's possible to use (22), but η must be set smaller in order to compensate for the presence of an extra term in (21).

REFERENCES

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

[2] A. W. Hull and W. K. Jenkins, "Preconditioned conjugate gradient methods for adaptive filtering," *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 540-543, Jun. 1991.

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

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

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

[6] R. Fletcher, *Practical Methods of Optimization.* Chichester, NY: Wiley, 1987, 2nd ed.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore: Johns Hopkins University Press, 1989, 2nd ed.

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