

CONJUGATE GRADIENT REUSE ALGORITHM USING A VARIABLE STEP-SIZE LINE SEARCH

A. Neil Birkett and Rafik A. Goubran

Department of Systems and Computer Engineering
 Carleton University, 1125 Colonel By Drive
 Ottawa, Canada, K1S 5B6
 Tel: (613) 520-5740, Fax: (613) 520-5727
 birkett@sce.carleton.ca goubran@sce.carleton.ca

ABSTRACT

A new algorithm is presented which combines the fast conjugate gradient algorithm and the modified variable step-size algorithm with gradient reuse to provide a convergence/tracking performance trade-off. A simplified version of the proposed algorithm is also presented that reuses weight updates to avoid calculating gradients and conjugate directions at every sample n . This simplified algorithm only invokes the conjugate gradient update every P th sample resulting in an overall complexity reduction by a factor of P as compared to the FCGA. Simulation results are also presented.

1.0 INTRODUCTION

The *fast conjugate gradient algorithm* (FCGA) uses an averaged instantaneous gradient over a *window* of past sample values [1]. The advantages of this windowed approach are (i) better tracking and convergence is achieved in nonstationary environments with correlated data compared to the RLS algorithm, and (ii) there are no stability problems associated with an exponential forgetting factor as in the RLS algorithm.

In this paper, we propose a new algorithm which is based on extending the FCGA to include conjugate direction *reuse* using dynamic step size control based on the Modified Variable Step Size (MVSS) algorithm [2]. We call this new algorithm the conjugate gradient reuse algorithm (CGRA) since it performs a simple one dimensional line search for the minimum along the estimated gradient direction.

The MVSS estimates the autocorrelation of the error signal to determine when the minimum of the performance surface is reached. When the error correlation is large, dynamic step size control adjusts the step size to be large thus speeding up the convergence. A measure of the “minimum” of the line search can be obtained by examining the error signals $e(j)$ and $e(j-1)$ where j is the line search iteration count. If $e(j) < e(j-1)$ then the algorithm is still searching for the minimum of the performance surface along this particular direction. If $e(j) \geq e(j-1)$ then the minimum has been reached, at which point we exit the search and replace the initial weight vector \mathbf{w}_0 with the one used to generate $e(j-1)$.

2.0 DESCRIPTION OF THE LEARNING ALGORITHM

Given an M -th order transversal adaptive filter with input $x(n)$, output $y(n)$ and desired signal $d(n)$, the cost function E to be minimized is a summation of squared errors from a window n_w of past values using the *current* weight vector $\mathbf{w}(n)$;

$$E(n) = \sum_{i=0}^{n_w-1} e^2(n-i) \Big|_{\mathbf{w}(n)} \quad (1)$$

The weights are updated using stochastic gradient method which minimizes the cost function;

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \frac{\partial E}{\partial \mathbf{w}} \quad (2)$$

The gradient estimate is calculated using the conjugate gradient technique. The complete algorithm is summarized below and uses triply indexed parameters. The parameter n refers to the main iteration number, where the data is shifted in on a sample by sample basis, k represents the conjugate direction iteration count and j represents the line search iteration count.

Fast Conjugate Gradient Reuse Algorithm

Initialization: $\mathbf{w}_0(0)=\mathbf{0}$

For each iteration n , do steps 1,2 and 3.

Step 1.

$$e(n) = \mathbf{w}_0^T(n)\mathbf{x}(n) - d(n) \quad (3)$$

$$\mu_{max}(n) = \frac{1 + \beta_k(n)}{\mathbf{x}^T(n)\mathbf{x}(n)} \quad (4)$$

$$\mathbf{g}_0(n) = [\nabla f(\mathbf{w}_0(n))]^T$$

$$= \left(\frac{2}{n_w}\right) \left[\sum_{i=0}^{n_w-1} \{[\mathbf{w}_0^T(n)\mathbf{x}(n-i) - d(n-i)]\mathbf{x}(n-i)\} \right] \quad (5)$$

$$\mathbf{d}_0(n) = \mathbf{g}_0(n) \quad (6)$$

Step 2. Repeat for $k=0,1,\dots,n_w-1$ where $n_w \leq m$

2a) set $\mu_{k,0}(n) = \mu_{max}(n)$ and $\mathbf{w}_{k,0}(n) = \mathbf{w}_k(n)$

Repeat Steps 2b-1) through 2b-4) for $j=1, \dots, n_w$ where $n_w \leq m$

$$\mathbf{2b-1)} \quad \mathbf{w}_{k,j}(n) = \mathbf{w}_{k,j-1}(n) + \mu_{k,j}(n)\mathbf{d}_k(n) \quad (7)$$

$$\mathbf{2b-2)} \quad e_{k,j}(n) = \mathbf{w}_{k,j}^T(n)\mathbf{X}(n) - d(n) \quad (8)$$

2b-3) Adjust the step size

$$\mu_{k,j}(n) = \begin{cases} \mu_{max}(n) & ; \mu_{k,j}(n) \geq \mu_{max}(n) \\ \mu_{min}(n) & ; \mu_{k,j}(n) \leq \mu_{min}(n) \\ \zeta \mu_{k,j-1}(n) + \gamma \rho_j^2(n) & ; \mu_{min}(n) < \mu_{k,j}(n) < \mu_{max}(n) \end{cases} \quad (9)$$

where

$$\rho_j(n) = \Gamma \rho_{j-1}(n) + (1 - \Gamma)e_j(n)e_{j-1}(n) \quad (10)$$

2b-4) if $e_{k,j}(n) > e_{k,j-1}(n)$ then proceed to Step 2c), else goto Step 2b-1).

$$\mathbf{2c)} \quad \mathbf{w}_{k+1}(n) = \mathbf{w}_{k,j}(n) \quad (11)$$

2d) Unless $k=n_w-1$, set

$$\mathbf{d}_{k+1}(n) = -\mathbf{g}_{k+1}(n) + \beta_k(n)\mathbf{d}_k(n) \quad , \text{where;} \quad (12)$$

$$\beta_k(n) = \frac{\mathbf{g}_{k+1}^T(n)\mathbf{g}_{k+1}(n)}{\mathbf{g}_k^T(n)\mathbf{g}_k(n)} \quad \text{and} \quad (13)$$

$$\begin{aligned} \mathbf{g}_{k+1}(n) &= [\nabla f(\mathbf{w}_{k+1}(n))]^T \\ &= \left(\frac{2}{n_w} \right) \left[\sum_{i=n-n_w+1}^n \{ [\mathbf{w}_{k+1}^T(n)\mathbf{X}(i) - d(i)]\mathbf{X}(i) \} \right] \end{aligned} \quad (14)$$

If $\beta_k(n) > 1$, go directly to Step 3), otherwise go to Step 2)

Step 3. Replace $\mathbf{w}_0(n+1)$ by $\mathbf{w}_k(n)$ and go back to Step 1.

β_k gives a measure of the rate of change of successive gradients. The calculation of β_k is done according to the Fletcher-Reeves method rather than the Polak-Ribiere method [3] since it tends to give a smoother convergence. Checking for $\beta_k > 1$ is a necessary step in the proposed algorithm. When \mathbf{g}_k is noisy it is possible that $\mathbf{g}_{k+1} \approx \mathbf{g}_k$ and the new β_k will be close to or larger than 1. Successive iterations of Step 2 will only serve to move the weight vector away from the optimum value and make the algorithm unstable. It necessary to limit the value of $\beta_k < 1$ for stability which can be expressed as follows [4];

$$0 < \mu_k < \frac{2(1 + \beta_k)}{\lambda_{max}} \quad (15)$$

where $0 < \beta_k < 1$ and λ_{max} is the maximum eigenvalue of the input data. Specifically, the gradient averaging extends the upper limit of the region of stability of μ_k from $2/\lambda_{max}$ to

$2(1+\beta_k)/\lambda_{max}$ but β_k must be kept below 1. We use this maximum step size each iteration and then impose a limit on the minimum step size.

2.1 Simplification of the Algorithm

If the performance surface does not change too rapidly, then by reusing the conjugate gradient *weight* updates (as opposed to direction updates), we can still step in the right direction and at the same time avoid the calculation of the true gradient. Thus if we only allow a gradient calculation every P input samples, we obtain a reduction in the complexity by a factor of P over the CGRA. This is the basis of the simplified CGRA. A variation of this idea was proposed in [5] for reducing the computational complexity of backpropagation weight updates in neural networks. The trade-off is that the convergence rate will become *poorer* in correlated environments as P increases. However, it provides a basis for trading computationally complexity for performance in the same way as the gradient window size n_w . The simplified algorithm is the same as the CGRA except for the following changes which are indicated with an asterisk in bold type;

Simplified Conjugate Gradient Reuse Algorithm (CGRA2):

Initialization: $\mathbf{w}_0(0)=\mathbf{0}, \beta_k(0)=0.$

*****count=0;**

For each iteration n, do steps 1,2 and 3.

Step 1. Shift in new data into vector $x(n)$

$$\mathbf{1b)} \quad e(n) = \mathbf{w}_0^T(n)\mathbf{x}(n) - d(n) \quad (16)$$

*****count=count+1**

*****if count=P, continue, else goto Step 3)**

Perform rest of Step 1) and Step 2) here

$$\mathbf{Step 3.} \quad \mathbf{*** If count=1,} \quad \Delta \mathbf{w}_k(n) = \mathbf{w}_k(n) - \mathbf{w}_o(n) \quad (17)$$

$$\mathbf{w}_o(n+1) = \mathbf{w}_k(n) \quad (18)$$

$$\mathbf{*** else} \quad \mathbf{w}_o(n+1) = \mathbf{w}_o(n) + \Delta \mathbf{w}_k(n) \quad (19)$$

Replace $\mathbf{w}_0(n+1)$ by $\mathbf{w}_k(n)$, and go back to Step 1.

3.0 COMPLEXITY

3.1 CGA

In the regular CGA, the number of multiplications required in Step 1) is $3m^2$ per gradient calculation or $6m^2$ in total, where m is the order of the filter. In step 2), the number of multiplications per sample is $m(6m^2 + 6m)$ per sample for an overall total of

$$6m^3 + 12m^2. \quad (20)$$

3.2 CGRA

In the CGRA, the number of multiplications per sample in *Step 1*) is $2mn_w + 1$. *Step 2b*) is done R times resulting in a complexity of $n_w R(2m + 6)$ multiplications per sample where $R \leq n_w$. *Step 2d*) is done $n_w - 1$ times for a complexity of $(n_w - 1)(2mn_w + 3m)$ multiplications per sample. Summing all of these contributions, the overall complexity of the CGRA is equal to;

$$(2mn_w + 1) + n_w R(2m + 6) + (n_w - 1)(2mn_w + 3m) \quad (21)$$

multiplications per sample. For $R = 1$ the CGRA will default to the FCGA algorithm with a complexity of

$$(2mn_w + 1) + n_w(2m + 6) + (n_w - 1)(2mn_w + 3m) \quad (22)$$

and for $n_w = 1$, it reverts approximately to the NLMS algorithm with a complexity of

$$2m + 6 \quad (23)$$

The standard RLS algorithm has complexity of

$$2m^2 + 4m \quad (24)$$

3.3 Simplified CGRA

The CGRA has a slight increase in computational complexity over the FCGA due to the gradient reuse rate R . However, if fewer than R successive steps of *2b*) are needed before the minimum is reached, this estimate of complexity would represent an upper bound. It is possible to limit the value of R to some value smaller than n_w to provide a limited complexity increase. Simulation results will show that by using a restricted R , it is possible to obtain the same performance with the CGRA as with the FCGA, even though the latter requires a larger window size to obtain this performance and is therefore more complex. The simplified CGRA only performs gradient calculations every P samples, and this reduces the complexity to;

$$\frac{(m + 1) + 2mn_w + n_w R(2m + 6) + (n_w - 1)(2mn_w + 3m)}{P} \quad (25)$$

multiplications per sample for $R \geq 2$. For $R = 1$, the algorithm reverts to the CGRA. Table 1 gives comparative complexities of the CGA, FCGA, CGRA, CGRA2 and RLS algorithms for $m=50$, $n_w=5$, $R=2$ and $P=3$.

4.0 SIMULATION RESULTS

In this section we apply the exponential RLS, NLMS, MVSS, FCGA, CGRA and CGRA2 algorithms to the problem of system identification. The unknown system is modelled by an impulse 50 taps long which is obtained from an exponentially decaying set of random values between ± 1 . This choice is repre-

TABLE 1. Comparison of algorithm complexity.

Algorithm	Mult./sample	Mult./sample for $m=50$, $n_w=5$, $R=2$ and $P=3$
CGA	equation (20)	780,000
FCGA	equation (22)	3631
CGRA	equation (21)	4161
CGRA2	equation (25)	1438
RLS	equation (24)	5200

sentative of a typical acoustic impulse response obtained in conference rooms with small reverberation times for applications in acoustic echo cancellers, where both fast convergence and tracking are required. The input to the system is a coloured noise sequence obtained from a single pole autoregressive process described by;

$$y(n) = 0.9y(n-1) + v(n) \quad (26)$$

where $v(n)$ is a unit variance white noise sequence. This signal is then filtered by the unknown system and finally, a small amplitude uncorrelated white gaussian noise signal is then added to the system output to produce a desired signal to noise ratio of 50 dB. In order to demonstrate the tracking capabilities of the algorithms, the unknown system impulse response is changed halfway through the data sequence by multiplying all coefficients by -1.0 . This change in the transfer function will cause a temporary increase in the Mean Squared Error (MSE) as the algorithms try to readjust the weights to the new optimum weight vector. The ability of a particular algorithm to quickly re-adapt its weights is a measure of its tracking performance. The Normalized Mean Squared Error (NMSE) convergence curves for the algorithms are plotted for comparison. The NMSE curves are obtained by averaging the error and desired signals over 100 independent runs and then smoothing according to the following formula,

$$NMSE(n) = 10 \log \left(\frac{\sum_{r=0}^{50} [\bar{e}_r(k)]^2}{\sum_{r=0}^{50} [\bar{d}_r(k)]^2} \right) dB \quad (27)$$

where $\bar{e}_r(k)$ and $\bar{d}_r(k)$ represent the averaged error and desired signals averaged over 100 independent trials and r represents the window values over which these averages are then smoothed, in this case equal to 50.

5.0 DISCUSSION OF SIMULATION RESULTS

In Figure 1 the RLS converges quickly owing to its insensitivity to eigenvalue spread. The FCGA and CGRA also converge quickly but the CGRA is faster than the FCGA. Both the MVSS and NLMS have poor convergence characteristics due to the

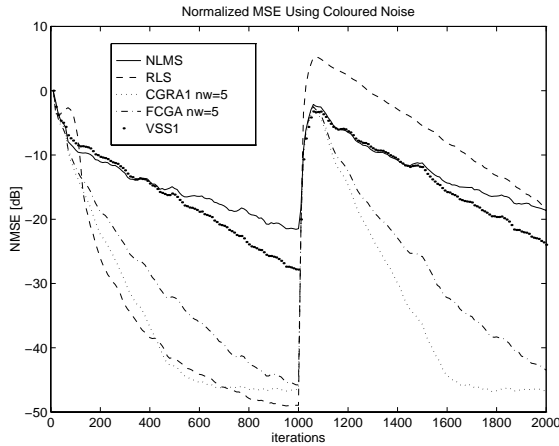


FIGURE 1. Comparison of NMSE using correlated noise input. There is a sudden change in the system transfer function at iteration 1000.

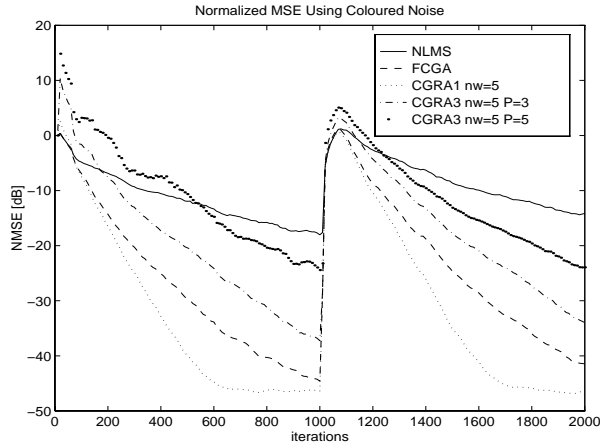


FIGURE 2. Comparison of NMSE using correlated noise showing effect of changing P on CGRA2.

correlated input data. At iteration 1000, the RLS algorithm has problems tracking due to the forgetting factor λ being close to 1. The CGRA and FCGA convergence rates after iteration 1000 are almost identical to the initial convergence rate. The CGRA obtains the best convergence rate of all the above algorithms.

In Figure 2, the convergence of the CGRA2 algorithm ($n_w=5$, $R=P=5$) outperform the NLMS algorithm 300 samples after the transfer function change even with a reduced gradient update rate. The complexity of the CGRA2 ($n_w=5$, $P=5$) for this case is approximately a factor of 5 less than the FCGA. The results indicate that depending on the value of P , the performance (and complexity) can be tailored to fall between the FCGA and NLMS algorithms.

The simplified CGRA can be used effectively in applications where good convergence and good tracking speed are required. Unlike the RLS that suffers from poor tracking due to the exponential least squares formulation, the CGRA can provide

increased convergence without affecting tracking ability when the window sizes are chosen appropriately.

Although not discussed in this paper, the CGRA technique can also be applied in the nonlinear domain as a method of minimizing a nonlinear cost function, similar to the method discussed in [6].

6.0 REFERENCES

- [1] G. K. Boray, M. D. Srinath, "Conjugate Gradient Techniques for Adaptive Filtering", *IEEE Trans. on Circ. and Sys.* Vol. CAS-1, Jan. 1992, pp. 1-10.
- [2] K. Mayyas, T. Aboulnasr, "A Robust Variable Step Size LMS-Type Algorithm: Analysis and Simulations", *Proceedings ICASSP'95*, pp. 1408-1411.
- [3] D. G. Leunberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Don Mills, Ont., 1973.
- [4] J. Proakis, "Channel Identification for High Speed Digital Communications", *IEEE Trans. Automat. Control*, Vol. AC-19, pp. 916-922, Dec. 1974.
- [5] D. R. Hush, J. M. Salas, "Improving the Learning Rate of Back-Propagation with the Gradient Reuse Algorithm", *Proceedings IEEE Int. Conf. on Neural Nets*, Vol.1, 1988, pp. 441-448.
- [6] A.N. Birkett, R. A. Goubran, "Fast nonlinear adaptive filtering using a partial window conjugate gradient algorithm", *Proceedings ICASSP96*, Vol.6, pp. 3542-3545.