Conjugate Gradient Reuse Algorithm with Dynamic Step Size Control

A. N. Birkett and R. A. Goubran

Keywords: Conjugate Gradient, Adaptive Filters, Gradient Reuse, Dynamic Step Size

ABSTRACT

A new algorithm is presented which combines the Fast Conjugate Gradient algorithm (FCGA), the Modified Variable Step Size (MVSS) algorithm, and gradient reuse to provide a convergence/tracking performance/complexity trade-off. The proposed algorithm reuses the estimated conjugate direction vector $\mathbf{d}_k(n)$ at each iteration k to search for the minimum along one particular conjugate direction, and thus performs a one dimensional line search. The variable step size reduces the number of iterations necessary to reach the minimum during the line search portion. Improved convergence and tracking is obtained compared to the NLMS, RLS, FCGA and MVSS algorithms when the input data is correlated and the environment is non-stationary. By restricting the number of iterations performed during the line search, it is possible to achieve the same performance as the FCGA but using a smaller window size, and therefore reduced complexity. A simplified version of the proposed algorithm is also presented that *reuses* weight updates (i.e the gradient) to avoid calculating gradients and conjugate directions at every sample n. This simplified algorithm only invokes the conjugate gradient update every Pth 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 Conjugate Gradient Algorithm (CGA) has been shown to provide convergence speed comparable to the recursive least square (RLS) algorithm even when the input signal autocorrelation matrix is ill conditioned [1]. However, the CGA computational burden is still high compared to variations based on the Least

Mean Square (LMS) algorithm [2]. Boray and Srinath [1] recently developed a *fast conjugate gradient algorithm* (FCGA) for adaptive filtering using an averaged instantaneous gradient over a *window* of past sample values. 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 [4]. 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. Other line search techniques such as the cubic interpolation algorithm [5] and the scaled conjugate gradient algorithm (SCGA) [6] have been proposed in the literature, however, these are formulated for full gradients and not for gradient estimates based on the technique described in [1]. A stochastic line search algorithm has also been presented in [7] which recursively minimizes the sum of squared errors on a linear manifold. It is similar to fast RLS algorithms since it iteratively calculates the optimum step size parameter. However, simulations in [7] are presented for correlated data in a stationary environment only.

In the CGRA, we use the MVSS technique during the line search portion such that successive values of the step size are calculated based on the autocorrelation of adjacent error samples. Thus, when the filter is far from the optimum and the autocorrelation of the error signal is large, the step size is large. This has the effect of reducing the number of iterations needed to find the minimum of a particular conjugate direction by increasing the line search step size where appropriate and in this respect the CGRA is similar to the SCGA. In effect the gradient direction is being reused several times to perform a one dimensional line search. We may also limit the update rate such that only R iterations of the line search are allowed. A reduced R limits the complexity increase, but still provides improvements in performance that are comparable to the FCGA with an increased gradient window size. Simulations in Section 3 illustrate that the

CGRA is capable of achieving the same rate of convergence as the FCGA, but with a smaller gradient window, and thus a lower (overall) computational complexity.

A simplified version of the CGRA is also presented which invokes the conjugate gradient update every *P* samples. In between gradient calculations, the weight updates are reused thus avoiding the calculation of the conjugate gradients altogether. This simplification results in an algorithm with complexity reduced by a factor of *P* that still maintains a performance which is somewhere between the FCGA and NLMS in terms of tracking and convergence, depending on *P.*

The remainder of this paper is organized as follows. In Section 2 the conjugate gradient method is reviewed, the FCGA, MVSS and gradient reuse methods are introduced, the conjugate gradient reuse algorithm and a simplified version of it are formulated and complexity issues are discussed. In Section 3, we present some simulation results comparing the NLMS, RLS, MVSS, FCGA and CGRA algorithms in a system identification context using correlated input data. Conclusions are presented in Section 4.

2.0 THE CONJUGATE GRADIENT ALGORITHM

Consider a transversal filter with *m* taps where $\mathbf{u}(n) = [u(n), u(n-1),...u(n-m+1)]^T$ represent the tap input vector at time *n*, $\mathbf{w}(n) = [w_1(n), w_2(n),...w_m(n)]^T$ represents the tap weight vector at time *n* and $d(n)$ represents the desired response at time *n*. The algorithms proposed have a tap weight update equation based on stepping in the direction of the negative gradient according to;

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)[-\nabla f(\mathbf{w}(n))]^T \quad (1)$$

where $\mu(n)$ is the time varying step size. The LMS algorithm uses an instantaneous value of gradient in place of its ensemble average [8] and in the RLS algorithm, the true gradient is replaced by a data dependent estimate using the inverse of the autocorrelation matrix. This significantly improves the convergence but also adds complexity. The conjugate gradient algorithm is based on updating the tap weights with new directions that are “non-interfering”, in otherwords, conjugate to each other. The CGA can be used to min-

imize a pure or approximated quadratic function by iteratively constructing direction vectors that are mutually conjugate and linearly independent [3]. This results in convergence properties that will minimize a quadratic function of *m* variables (i.e weights) in no more than *m* iterations and provide fast convergence. The concept of "non-interfering" directions can be made mathematically explicit by considering a multidimensional function $f(\cdot)$. Let point $\mathbf{x}$ represent the origin of a particular multidimensional system with a set of linearly independent direction vectors $\mathbf{x}_0,\mathbf{x}_1, \mathbf{x}_2,...\mathbf{x}_{m-1}$ which represent the coordinate system. Let $\Delta\mathbf{x}$ be an arbitrary *m* by 1 vector representing the distance from the origin $\mathbf{x}$ along the direction vectors $\mathbf{x}_i$. Then any function value $f(\mathbf{x}+\Delta\mathbf{x})$ can be approximated by a Taylor series which can be truncated to a quadratic function as follows;

$$\begin{aligned} f(\mathbf{x}+\Delta\mathbf{x}) &= f(\mathbf{x}) + \sum_i \frac{\partial f(\mathbf{x})}{\partial x_i}\Delta x_i + \frac{1}{2}\sum_{ij}\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\Delta x_i \Delta x_j + \ldots \\ &\approx c + \Delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{b} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \Delta\mathbf{x} \end{aligned} \tag{2}$$

where

$$c \equiv f(\mathbf{x}) \qquad \mathbf{b} \equiv \frac{\partial f(\mathbf{x})}{\partial x_i} = \nabla f(\mathbf{x}) \qquad \mathbf{Q} \equiv \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \nabla^2 f(\mathbf{x}) \tag{3}$$

The matrix $\mathbf{Q}$ is the $m \times m$ Hessian matrix of the function at $\mathbf{x}$, and $\mathbf{b}$ is the $m \times 1$ gradient of the function at $\mathbf{x}$. The approach in the conjugate direction method is to obtain a set of linearly independent direction vectors $\mathbf{x}_0,\mathbf{x}_1, \mathbf{x}_2,...\mathbf{x}_{m-1}$ which are conjugate with respect to $\mathbf{Q}$ so that the optimum solution vector $\mathbf{x}^o$ minimizes equation (2). $\mathbf{x}^o$ can be expressed as;

$$\mathbf{x}^o = \alpha_0\mathbf{x}_0 + \alpha_1\mathbf{x}_1 + \ldots + \alpha_{m-1}\mathbf{x}_{m-1} \tag{4}$$

and the constants are given by [9];

$$\alpha_i = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{b}}{\mathbf{x}_i^{\mathrm{T}}\mathbf{Q}\mathbf{x}_i} \tag{5}$$

The conjugate gradient algorithm determines the appropriate orthogonal set of direction vectors and constants α_i. If the direction vectors are mutually conjugate and linearly independent, then the initial guess $\mathbf{x}$

will converge to the optimum $\mathbf{x}^o$ after m steps, that is $\mathbf{x}_m=\mathbf{x}^o$. The conjugate gradient algorithm can be implemented in a block processing form as in [2], or as an on-line method described in this paper.

By modifying the CGA for a *nonquadratic* function as given in p. 138 of [3], an algorithm can be derived that does not require computation of the Hessian matrix **Q**. Essentially the second derivative is replaced by the difference quotient;

$$\mathbf{Q}\mathbf{p}_k = \nabla^2 f(\mathbf{x}_k)\mathbf{p}_k \approx \frac{\nabla f(\mathbf{x}_k + \sigma\mathbf{p}_k) - \nabla f(\mathbf{x}_k)}{\sigma} \tag{6}$$

where σ is some small constant which can be set to one. The penalty for not calculating **Q** is that *two* gradient calculations must be performed per iteration, one at the current value of the vector $\mathbf{x}_k$ and one at $\mathbf{y}_k$ where $\mathbf{y}_k = \mathbf{x}_k + \sigma\mathbf{p}_k$ and $\mathbf{p}_k = -[\nabla f(\mathbf{x}_k)]^T$ is the negative of the gradient at $\mathbf{x}_k$. However, since the computation of the Hessian matrix is of order $O(m^3)$ and the calculation of a single gradient is of order $O(m^2)$, the savings are substantial if the filter order m is large. We now replace the dependent variable $\mathbf{x}_k$ given above with a set of weights $\mathbf{w}_k(n)$ we obtain the following algorithm. Note that n refers to the time index and k refers to the conjugate direction count in the sequel.

<u>*Conjugate Gradient Algorithm Using 2 Gradient Calculations per iteration:*</u>

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

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

Step 1. a) Starting with an initial weight vector $\mathbf{w}_0(n)$ compute the following;

$$\mathbf{g}_0(n) = [\nabla f(\mathbf{w}_0(n))]^T \tag{7}$$

$$\mathbf{y}_0(n) = \mathbf{w}_0(n) - \mathbf{g}_0(n) \tag{8}$$

$$\mathbf{p}_0(n) = [\nabla f(\mathbf{y}_0(n))]^T \tag{9}$$

b) set $\mathbf{d}_0(n) = \mathbf{g}_0(n)$ (10)

Step 2. Repeat for k=0,1,. . . ,m-1

a) set $\mathbf{w}_{k+1}(n) = \mathbf{w}_k(n) + \alpha_k \mathbf{d}_k(n)$ where α_k is the optimum step size; (11)

$$\alpha_k = \frac{-\mathbf{g}_k^T(n)\mathbf{d}_k(n)}{\mathbf{d}_k^T(n)(\mathbf{g}_k(n) - \mathbf{p}_k(n))} \quad (12)$$

b) Compute the gradients at the new weight vector $\mathbf{w}_{k+1}$

$$\mathbf{g}_{k+1}(n) = [\nabla f(\mathbf{w}_{k+1}(n))]^T \quad (13)$$

$$\mathbf{y}_{k+1}(n) = \mathbf{w}_{k+1}(n) - \mathbf{g}_{k+1}(n) \quad (14)$$

$$\mathbf{p}_{k+1}(n) = [\nabla f(\mathbf{y}_{k+1}(n))]^T \quad (15)$$

c) Unless k=m-1, obtain the new direction vector $\mathbf{d}_{k+1}(n) = -\mathbf{g}_{k+1}(n) + \beta_k \mathbf{d}_k(n)$ where; (16)

$$\beta_k = \frac{\mathbf{g}_{k+1}^T(n)\mathbf{g}_{k+1}(n)}{\mathbf{g}_k^T(n)\mathbf{g}_k(n)} \quad (17)$$

and repeat *Step 2 (a)*.

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

β_k gives a measure of the rate of change of successive gradients. If $\beta_k > 1$, then the magnitude of a successive gradient vector is not decreasing, meaning that the minimum has been reached. If $\beta_k > 1$ is a termination condition for conjugate direction k in Step 2). The calculation of β_k is done according to the Fletcher-Reeves method rather than the Polak-Ribiere method [9] since it tends to give a smoother convergence. In the CGA algorithm above, it is assumed that the gradient calculations are true gradients in the sense that all the data is available and that at least m conjugate directions can be calculated. However, this is computationally expensive and impractical if real time processing is required.

2.1 Fast Conjugate Gradient Method

It is shown in [1] that if a gradient estimate is constructed by averaging the instantaneous gradient estimates over a window size n_w of past values, there will be at least min(m,n_w) linearly independent direction

vectors in the gradient estimate, where *m* is the filter order. Specifically the instantaneous estimate at time *n* is replaced by a windowed estimate as follows;

$$[\nabla f(\mathbf{w}(n))]^T = \mathbf{g}(n) \approx \left(\frac{2}{n_w}\right)\left[\sum_{i=0}^{n_w-1} \{[\mathbf{w}^T(n)\mathbf{x}(n-i) - d(n-i)]\mathbf{x}(n-i)\}\right] \quad (18)$$

With this gradient estimate, the CGA will terminate in n_w loops of *Step 2*. At low values of n_w, the gradient estimates are poor resulting in inappropriate values of the step size α_k. In addition, the computation of α_k in equation (12) still requires $2mn_w$ multiplies and one division. By removing the calculation of α_k and replacing it with a constant value, the calculation of **p** and **y** are also no longer required, thus simplifying the algorithm. However, in this paper, instead of using a fixed step size as proposed in [1], a normalized step size is used;

$$\tilde{\alpha}_k(n) = \frac{\alpha}{\mathbf{x}^T(n)\mathbf{x}(n) + \varepsilon} \quad (19)$$

where $0 < \alpha < 2$ and ε is some small value. This slight variation of the FCGA was used in all the simulations. It should be pointed out that by avoiding the calculation of the optimum step size at each sub-iteration, there is no guarantee that the successive direction vectors will be truly conjugate. This will result in reduced convergence rates.

2.2 Dynamic Step Size Line Search Algorithm

Gradient computations are expensive compared to computing the output *y* for a particular set of weights **w**. An improvement in performance over the FCGA can be obtained with a marginal increase in complexity by reusing the estimated gradient direction vector $\mathbf{d}_k$ several times until the resulting weight updates no longer lead to a reduction in error. We search for the minimum of a conjugate direction before evaluating the next conjugate direction and thus maintain true conjugacy of directions. A gradient direction is computed and a search is performed for the *minimum* along that line. This search can be optimized by exploiting additional information about the surface which can be either known *a priori* or estimated as the process

continues. For simplicity, we assume that no additional information is available and that the search consists of a series of steps which terminate near the minimum. There is a trade-off between the step size and the accuracy of the search. If the step size is too small, we have good accuracy but spend a lot of time performing the search, and vice versa. We are concerned more about the number of steps taken to reach the minimum of the line search rather than the accuracy and therefore wish to use the maximum allowable step size when we are far from the minimum of the error surface.

The MVSS is a robust variable step size algorithm for LMS type filters that 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. When the error correlation is small, as is the case in high noise environments, or when the minimum is reached, the step size is reduced correspondingly. Specifically, the step size is updated by the following formulas;

$$\mu(k+1) = \begin{cases} \mu_{max} & ;\mu(k+1) \geq \mu_{max} \\ \mu_{min} & ;\mu(k+1) \leq \mu_{min} \\ \zeta\mu(k) + \gamma\rho^2(k) & ;\mu_{min} < \mu(k+1) < \mu_{max} \end{cases} \quad (20)$$

$$\text{where} \quad \rho(k) = \Gamma\rho(k-1) + (1-\Gamma)e(k)e(k-1) \quad (21)$$

$$\text{and} \quad \begin{cases} 0 < \zeta < 1 \\ \Gamma < 1 \\ \gamma > 0 \end{cases} \quad (22)$$

The parameter γ controls the convergence time as well as the final misadjustment. The parameter ζ controls the averaging of the step size update and Γ controls the averaging time constant of the filtered error update. The parameter ρ gives a short time estimate of the error signal autocorrelation. Typical parameter values are ζ=0.97, Γ=0.99, γ=1e-5 [4]. The advantage of the MVSS algorithm over the standard variable step size (VSS) algorithm is its relative insensitivity to noisy signals due to the time average autocorrela-

tion process. This technique is adopted in the CGRA to dynamically control the step size update during the line search. A measure of the "minimum" of the line search can be obtained by examining $e(j)$ and $e(j\text{-}1)$ 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 $\mathbf{w}_o$ with the one used to generate $e(j\text{-}1)$.

2.3 Maximum and Minimum Step Sizes

Checking for $\beta_k > 1$ is a necessary step in the proposed algorithm. Proakis [11] demonstrated that the conjugate gradient algorithm resembles the operation of a first-order recursive filter whose output $\mathbf{d}_k$ is given by equation (16). An m-dimensional filter is in effect a set of m identical single-pole (low pass) filters operating in parallel which corresponds to filtering the gradients with a time-variant filter. Algorithms which use linearly filtered noisy gradients can provide faster convergence than the conventional LMS algorithm. When $\mathbf{g}_k$ are noisy however, 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. In [11] the author found it necessary to limit the value of $\beta_k < 1$ in a channel equalization experiment and obtained the conditions for stability which can be expressed as follows;

$$0 < \mu_k < \frac{2(1+\beta_k)}{\lambda_{max}} \tag{23}$$

$$\text{where } 0 < \beta_k < 1 \tag{24}$$

where 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.

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.

Conjugate Gradient Reuse Algorithm (CGRA):

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

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

Step 1. *a)* Shift in new data into vector $\mathbf{x}(n)$

b) Starting with an initial weight vector $\mathbf{w}_0(n)$, compute the initial error;

$$e(n) = \mathbf{w}_0^T(n)\mathbf{x}(n) - d(n) \quad (25)$$

c) Compute the maximum step size according to

$$\mu_{max}(n) = \frac{1 + \beta_k(n)}{\mathbf{x}^T(n)\mathbf{x}(n)} \quad (26)$$

d) Compute the initial windowed gradient estimate;

$$\mathbf{g}_0(n) = [\nabla f(\mathbf{w}_0(n))]^T = \left(\frac{2}{n_w}\right)\left[\sum_{i=0}^{n_w-1}\{[\mathbf{w}_0^T(n)\mathbf{x}(n-i) - d(n-i)]\mathbf{x}(n-i)\}\right] \quad (27)$$

e) set $\mathbf{d}_0(n) = -\mathbf{g}_0(n)$ (28)

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

a) 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,. n_w where $n_w \le m$

2b-1) Set $\mathbf{w}_{k,j}(n) = \mathbf{w}_{k,j-1}(n) + \mu_{k,j}(n)\mathbf{d}_k(n)$ (29)

2b-2) Compute the new error output using

$$e_{k,j}(n) = \mathbf{w}_{k,j}^T(n)\mathbf{x}(n) - d(n) \quad (30)$$

2b-3) Adjust the step size

$$\mu_{k,j}(n) = \begin{cases} \mu_{max}(n) & ;\mu_{k,j}(n) \geq \mu_{max}(n) \\ \mu_{min}(n) & ;\mu_{k,j}(n) \leq \mu_{min}(n) \\ \zeta\mu_{k,j-1}(n) + \gamma\rho_j^2(n) & ;\mu_{min}(n) < \mu_{k,j}(n) < \mu_{max}(n) \end{cases} \quad (31)$$

where $\rho_j(n) = \Gamma\rho_{j-1}(n) + (1-\Gamma)e_j(n)e_{j-1}(n)$ (32)

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

c) restore the "optimum" weight vector $\mathbf{w}_{k+1}(n) = \mathbf{w}_{k,j}(n)$ for this direction.

d) Unless $k=m_w$-1, set $\mathbf{d}_{k+1}(n) = -\mathbf{g}_{k+1}(n) + \beta_k(n)\mathbf{d}_k(n)$,where; (33)

$$\beta_k(n) = \frac{\mathbf{g}_{k+1}^T(n)\mathbf{g}_{k+1}(n)}{\mathbf{g}_k^T(n)\mathbf{g}_k(n)} \quad \text{and} \quad (34)$$

$$\mathbf{g}_{k+1}(n) = [\nabla f(\mathbf{w}_{k+1}(n))]^T = \left(\frac{2}{n_w}\right)\left[\sum_{i=n-n_w+1}^{n} \{[\mathbf{w}_{k+1}{}^T(n)\mathbf{x}(i) - d(i)]\mathbf{x}(i)\}\right] \quad (35)$$

I f $\beta_k(n) > 1$, *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*.

The reuse of the gradient is performed in *Step 2b-1)* where the newly computed direction vector $\mathbf{d}_k$ is used to successively update $\mathbf{w}_k$. Note that $\mathbf{d}_k(n) = -\mathbf{g}_k(n) = -[\nabla f(\mathbf{w}_k(n))]^T$ only during the first iteration of the line search when k=0 and that during successive iterations, $\mathbf{d}_k$ will change. During the line search, we have imposed an allowed limit of n_w steps of loop *2b)* to reach the minimum of a particular conjugate direction. This factor was chosen assuming that the lower the n_w, the poorer the estimate of the true gradient and therefore, we wish to limit the number of steps in any particular direction to place a limit on the number of steps taken should they be in the wrong direction. It was often observed in the simulations that fewer than n_w successive steps of *2b)* need to be taken before the minimum is reached and therefore this limit is not too restrictive. If the minimum of a particular conjugate direction has not been reached

before the next conjugate direction is calculated, or if the gradient estimate is poor, there is no guarantee that the new directions will be conjugate with respect to one another. This will slow convergence, however, it is still superior to the NLMS algorithm.

2.4 Simplified CGRA

The CRGA provides an averaged gradient which also points in the optimum direction towards the minimum of the performance surface based on the available information in the gradient window. If we assume that the performance surface does not change too rapidly, then it is safe to assume that 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. 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 by Hush and Salas [10] for reducing the computational complexity of backpropagation weight updates in neural networks where they showed that the convergence rate speed-up or slow-down is a related to the reuse rate. It is also possible to reuse the weight updates several times per sample iteration n, however, for the application described here, we only update the weights once per sample with a gradient calculation every P samples.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 algorithm is the same as the CGRA except for the following changes which are indicated with an asterisk in bold type;

<u>*Simplified Conjugate Gradient Reuse Algorithm (CGRA2)*</u>:

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. *a)* Shift in new data into vector $\mathbf{x}(n)$

b) Starting with an initial weight vector $\mathbf{w}_0(n)$, compute the initial error;

$$e(n) = \mathbf{w}_0^T(n)\mathbf{x}(n) - d(n) \quad (36)$$

******count=count+1***
******if count=P, continue, else goto Step 3)***

Perform rest of *Step 1)* and *Step 2) here*

Step 3. ****** If count=1,*** $\Delta\mathbf{w}_k(n) = \mathbf{w}_k(n) - \mathbf{w}_o(n)$ (37)

$$\mathbf{w}_o(n+1) = \mathbf{w}_k(n) \quad (38)$$

****** else*** $\mathbf{w}_o(n+1) = \mathbf{w}_o(n) + \Delta\mathbf{w}_k(n)$ (39)

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

2.5 Complexity

In the regular CGA, the number of multiplications required in Step 1) is $3m^2$ per gradient calculation or $6m^2$ total. In step 2), the number of multiplications per sample is $m(6m^2 + 6m)$ per sample for an overall total of $6m^3 + 12m^2$. 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 $R \le n_w$. *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;

$$(2mn_w + 1) + n_w R(2m + 6) + (n_w - 1)(2mn_w + 3m) \quad (40)$$

multiplications per sample. For R =1 the CGRA will default to the FCGA algorithm and for n_w=1, it reverts to the NLMS algorithm. The standard RLS algorithm has complexity of ($2m^2$+4m). 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;

$$(m+1) + \frac{2mn_w + n_w R(2m+6) + (n_w - 1)(2mn_w + 3m)}{P} \quad (41)$$

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.

TABLE 1. Comparison of algorithm complexity.

Algorithm	**Mult./sample**	**Mult./sample for m=50, n_w=5**
CGA	$6m^3 + 12m^2$	780,000
FCGA	$(2mn_w + 1) + n_w(2m+6) + (n_w - 1)(2mn_w + 3m)$	3631
CGRA	$(2mn_w + 1) + n_w R(2m+6) + (n_w - 1)(2mn_w + 3m)$	4161
CGRA2	$(m+1) + \frac{2mn_w + n_w R(2m+6) + (n_w - 1)(2mn_w + 3m)}{P}$	1438
RLS	$2m^2 + 4m$	5200

3.0 SIMULATIONS

In this section, we apply the CGRA to the problem of system identification as illustrated in Figure 1. 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 representative 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;

$$y(n) = 0.9y(n-1) + v(n) \quad (42)$$

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 CGRA, 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 RLS, NLMS, FCGA, MVSS and CGRA 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,

$$NMSE(n) = 10\log\left(\frac{\sum_{r=0}^{50}[\overline{e}_r(k)]^2}{\sum_{r=0}^{50}[\overline{d}_r(k)]^2}\right) dB \qquad (43)$$

where $\overline{e}_r(k)$ and $\overline{d}_r(k)$ 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. A summary of the parameters used in the simulations are listed in Table 2.

TABLE 2. List of parameters used during simulations.

Algorithm	***#Taps m***	$\overline{\alpha}$	λ	μ_{max}	μ_{min}	ξ	Γ	γ	n_w	***SNR***
NLMS	50	0.5	0.5							50 dB
RLS	50		0.997							50 dB
MVSS	50			1.0	1e-5	0.97	0.99	1e9		50 dB
FCGA	50	0.5							5	50 dB
CGRA	50			see eqn. (26)	$\frac{\mu_{max}}{10}$	0.4	0.4	1e2	5	50 dB

The normalized step size parameter $\overline{\alpha}$ is set to 0.5 for both the NLMS and modified FCGA algorithms. The exponential forgetting factor λ for the RLS is 0.997 since it was found in the simulations that a lower value

caused instability. The ξ parameter for the MVSS is large since the data sequences used are 16 bit integer which is normalized with respect to 32768. The window sizes for the FCGA and CGRA are both set to 5 which provides a good performance/complexity trade-off. The minimum step size for the line search portion of the CGRA was chosen as $\mu_{min} = \frac{\mu_{max}}{10}$. The signal to noise ratio of the desired signal *d* is set to 50 dB.

3.1 Results

***Experiment #1 Comparison of Algorithms with Correlated Input*:** Figure 2 shows the results when the input is coloured by the first order autoregressive process described by equation (42). During the first part of the training, 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 correlated input data. At iteration 1000, the unknown system is changed and the RLS algorithm has problems tracking due to the forgetting factor λ being close to 1 and only manages to obtain a lower error than the NLMS and MVSS algorithms by iteration 1500. 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.

***Experiment #2 Comparison of FCGA and CGRA*:** In this experiment we compare convergence and tracking performance obtained for the FCGA and CGRA using different window sizes for reduced complexity. The conditions for this experiment are the same as in Experiment #1 (correlated input) and the stepsize limits for the CGRA are listed in Table 2. The results in Figure 3 show the performance of the CGRA with a limited gradient reuse rate *R*, as compared to the FCGA using n_w=5 and n_w=8. In this experiment, the NMSE curves were obtained using the parameters listed in Table 2 which also indicates the relative complexity.

TABLE 3. Parameter and complexity comparison for FCGA (n_w=5 and 8) and CGRA (n_w=5, R=2).

Algorithm	*#Taps m*	$\overline{\alpha}$	ξ	Γ	γ	n_w	*R*	*SNR*	*Complexity (mults/iter)*
FCGA	50	0.5				5		50 dB	3631
FCGA	50	0.5				8		50 dB	8299
CGRA	50		0.4	0.4	1e2	5	2	50 dB	4161

The gradient averaging window n_w in the FCGA had to be increased to 8 in order to obtain the same tracking convergence performance as the CGRA with R=2 and n_w=5. In this example, the complexity of the FCGA(n_w=8) is approximately 99% higher than the CGRA(n_w=5, R=2) for similar performance results using correlated input signals.

Experiment #3 Simplified CGRA performance: The conditions for this experiment are the same as in Experiment #1 with parameters and complexity listed in Table 2. The results in Figure 4 show the performance of CGRA and simplified CGRA (here called CGRA2), as compared to the NLMS and FCGA.

TABLE 4. Comparative complexity using NLMS, FCGA, CGRA and CGRA2 (n_w=R=5).

Algorithm	*#Taps m*	$\overline{\alpha}$	ξ	Γ	γ	n_w	*R*	*P*	*SNR*	*Complexity (mults/iter)*
NLMS	50	0.5							50 dB	102
FCGA	50	0.5				5			50 dB	3631
CGRA	50		0.4	0.4	1e2	5	5	1	50 dB	5751
CGRA2	50		0.4	0.4	1e2	5	5	3	50 dB	1968
CGRA2	50		0.4	0.4	1e2	5	5	5	50 dB	1201

The convergence curves illustrate that the CGRA outperforms all other algorithms. 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. There are some transients during the first few iterations since the initial weight change estimates (which are reused P times) will be inaccurate during this period. The transients are higher for increased P both during initial convergence and when the transfer function is changed at iteration 1000 but will die out as the algorithm converges. The results indicate that depending on the value of P, the convergence rate can be tailored to be fast or slow. Increasing the value of P reduces

the convergence rate (and complexity) such that it falls somewhere between the FCGA and NLMS algorithms.

4.0 CONCLUSIONS

This paper has introduced a new version of the Conjugate Gradient Algorithm which is based on combining the gradient averaging window method with gradient reuse and dynamic step size to replace the optimum step size as calculated in the conventional CGA. The CGRA has reduced complexity as compared to the regular CGA and is slightly more complex than the FCGA depending on the number of iterations performed during the one dimensional line search.The CGRA has been shown to have better convergence and tracking properties than the FCGA in correlated nonstationary environments and can achieve the same performance as the FCGA with reduced complexity. A simplified version of the proposed algorithm has also been presented which computes the gradient every P samples, thus obtaining a reduction in complexity with a corresponding trade-off in convergence rate. The CGRA can be tailored to achieve a convergence rate greater than the RLS algorithm without suffering from stability problems associated with an RLS exponential forgetting factor.Although not described here, the CGRA method can also be extended to nonlinear neural networks as a to improve convergence speed and is the subject of future study.

5.0 ACKNOWLEDGEMENTS

The authors wish to thank NSERC, Carleton University, Nortel and the Telecommunications Research Institute of Ontario (TRIO) for their financial support and also Dr. Heping Ding (Nortel) for his comments on improving the paper.

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, pp. 1-10, Jan. 1992.

[2] J. S. Lim, C. K. Un, “Conjugate Gradient Algorithm for Block FIR Adaptive Filtering”, *Electronic Letters*, Vol. 29, No. 5, pp.428-429, March 1993.

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

[4] K. Mayyas, T. Aboulnasr, “A Robust Variable Step Size LMS-Type Algorithm: Analysis and Simulations”, *ICASSP’95*, pp. 1408-1411.

[5] E.M Johansson, F. U. Dowla, D. M. Goodman, “Backpropagation Learning for Multi-layer Feed-forward Neural Networks Using the Conjugate Gradient Method”, *International Journal of Neural Systems*, Vol.2, No. 4, pp. 291-302, 1991.

[6] M. S. Moller, “A scaled conjugate gradient algorithm for fast supervised learning” *Neural Networks*, Vol. 6, No. 4, pp. 525-534, 1993.

[7] C. E. Davila, “Line Search Algorithms for Adaptive Filtering”, *IEEE Trans. Sig. Proc.*, Vol 41, No. 7, pp. 2490-2494, July 1993.

[8] S. Haykin, *Adaptive Filter Theory 3rd Ed.*, Prentice Hall Information and System Sciences Series, N.J., 1996.

[9] D. G. Leunberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Don Mills, Ont., 1973.

[10] 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.

[11] J. Proakis, “Channel Identification for High Speed Digital Communications”, *IEEE Trans. Automat. Control*, Vol. AC-19, pp. 916-922, Dec. 1974.

[12] A. Gilloire, T. Petillon, “A Comparison of the NLMS and Fast RLS Algorithms for the Identification of Tim-Varying systems with Noisy Outputs: Application to Acoustic Echo Cancellation”, *Signal Processing V:Theories and Applications*, L. Torres et. all. editors, Elsevier, pp. 417-420, 1990.

[13] L. E. Scales, *Introduction to Non-linear Optimization*, Springer-Verlag, New York, 1985.

7.0 ILLUSTRATIONS

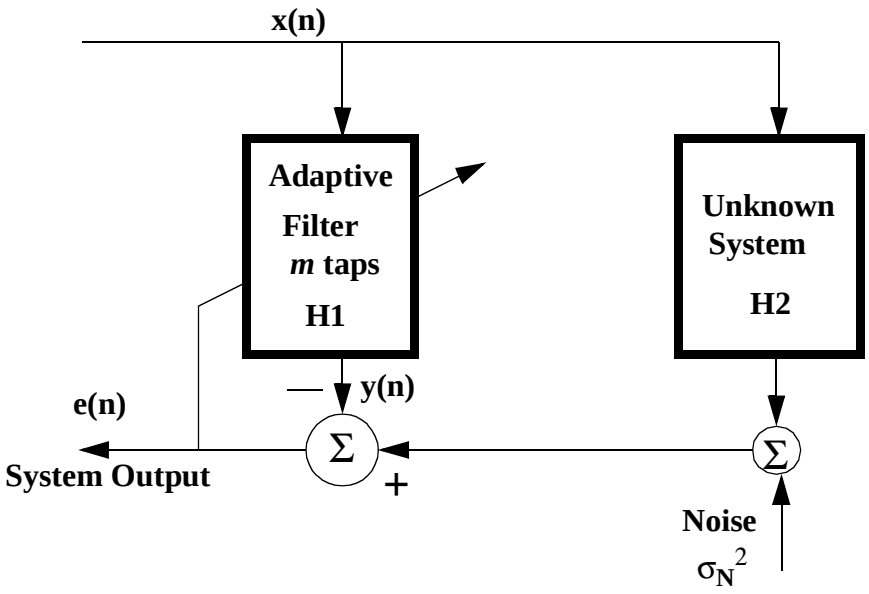

FIGURE 1. System identification model. The input x(n) is either a white noise source w(n) or an AR(1) process. The unknown channel consists of an exponentially decaying impulse 50 taps long. H1 is a 50 tap transversal filter which is updated according to the NLMS, RLS, MVSS, FCGA or CGRA algorithm. An uncorrelated noise source with variance σ_N^2 is added to the adaptive filter output y(n) to produce an SNR of 50 dB.

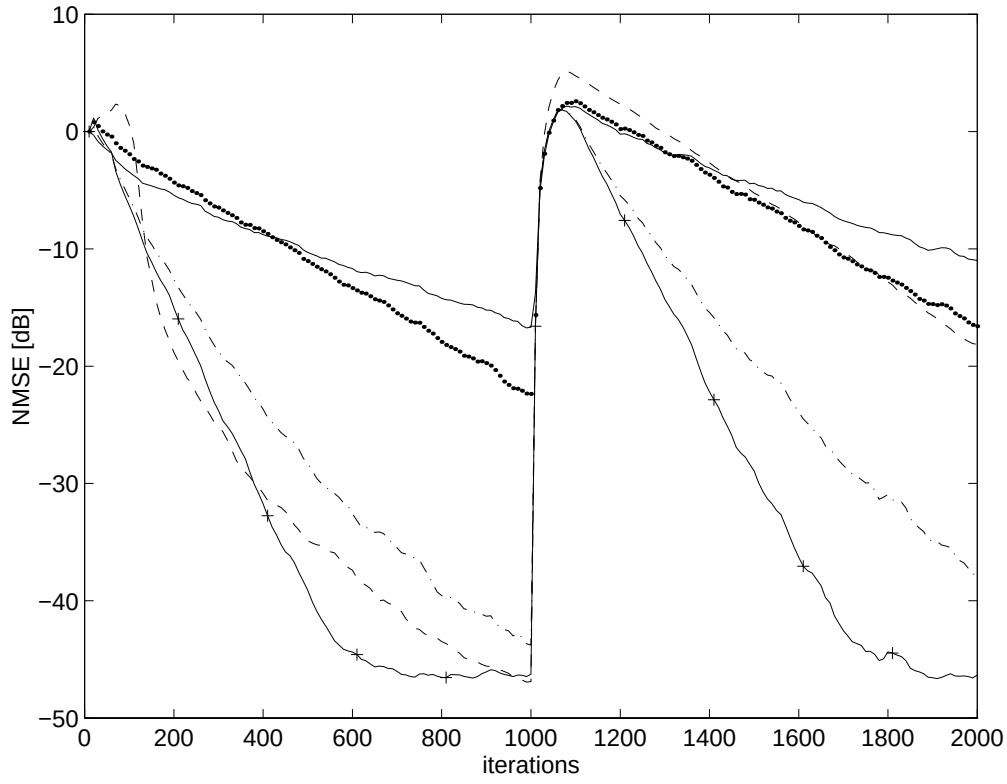

FIGURE 2. Experiment #1 results. Correlated noise input with a sudden change in the unknown system transfer function at iteration 1000.

NLMS

RLS

CGRA with n_w=5

FCGA with n_w=5

MVSS.

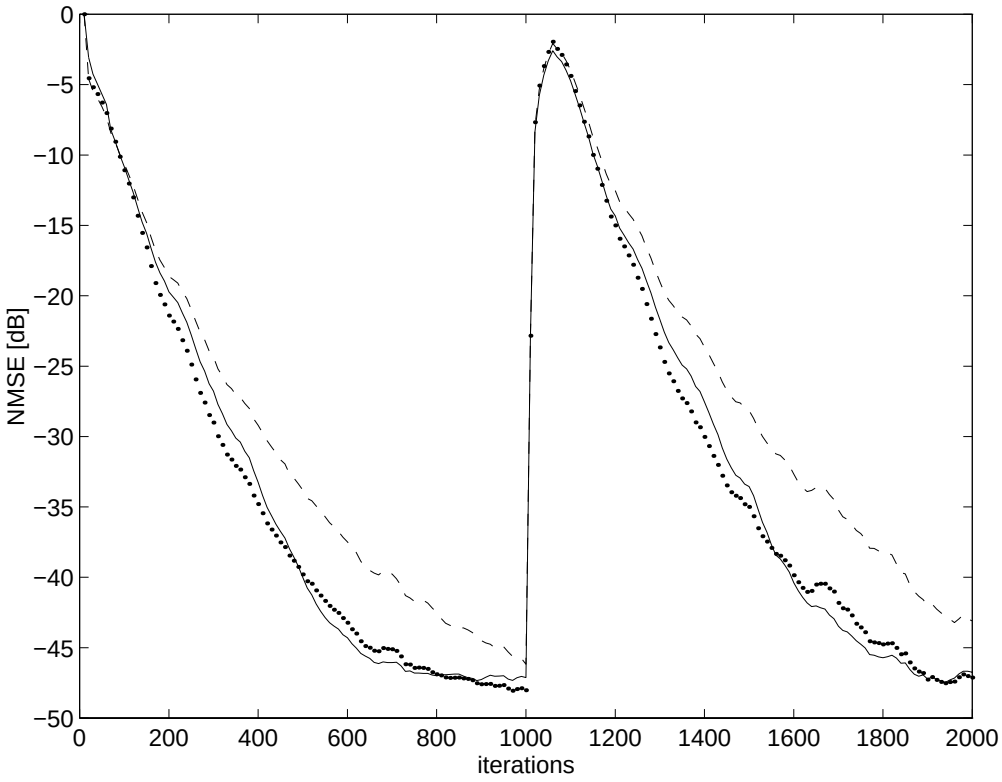

FIGURE 3. Experiment #2 results. Comparison of FCGA and CGRA using a limited gradient reuse rate. Correlated noise input with a sudden change in the unknown system transfer function at iteration 1000.

——— CGRA with R=2 and n_w=5

– – – – FCGA with n_w=5

..... FCGA with n_w=8.

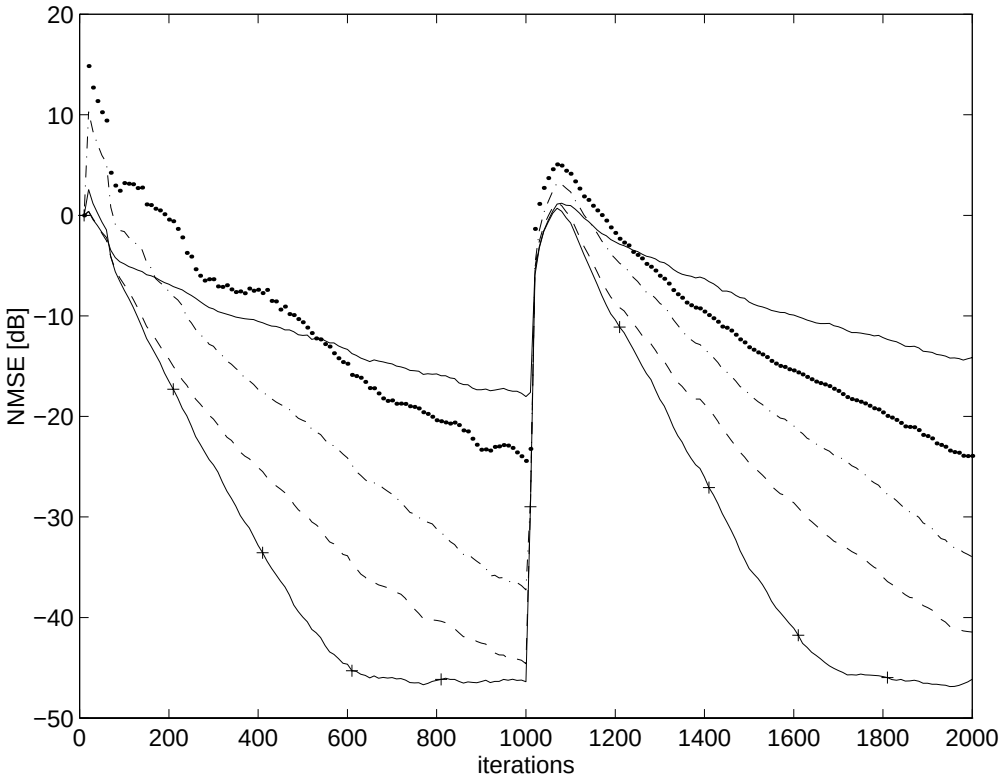

FIGURE 4. Experiment #3 results. Simplified CGRA performance results (CGRA2). Correlated noise input with a sudden change in the unknown system transfer function at iteration 1000.

——— NLMS.

– – – – FCGA with n_w=5.

—+ +— CGRA with n_w=5.

– . – . – CGRA2 with n_w=5 and P=3.

..... CGRA2 with n_w=5 and P=5.