FAST CONJUGATE GRADIENT ALGORITHM WITH NORMALIZED STEP SIZE

A. N. Birkett, R. A. Goubran
Department of Systems and Computer Engineering
Carleton University, 1125 Colonel By Drive
Ottawa, Canada, K1S 5B6
Tel: (613) 788-5740, Fax: (613) 788-5727
e-mail: birkett@sce.carleton.ca
EDICS CLASSIFICATION SP 2.6.2

ABSTRACT

A modified conjugate gradient algorithm is proposed which uses a gradient average window to provide a trade-off between convergence rate and complexity. Simplification is obtained by replacing the optimum step size α_k by a normalized step size $\overline{\alpha}$. Improved convergence is obtained with even small choices of window size.

1.0 INTRODUCTION

The limitations of the Least Mean Square (LMS) family of algorithms include sensitivity to parameter selection, sensitivity to the eigenvalue spread of the input data and long training times required to obtain a small error output. The later shortcomings are addressed in this paper. Partial conjugate direction methods can be regarded as being somewhat intermediate between the method of steepest descent and Newton's method, in terms of complexity and convergence properties [1][2]. This gives the designer the option of improving the convergence rate at the expense of increased complexity. Although the algorithms presented here are meant for linear systems, the idea can be extended to nonlinear system identification using neural networks [3][4][5], or as a method to speed convergence in systems limited by the backpropagtion algorithm as in [6]

2.0 DESCRIPTION OF ALGORITHMS

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 $d(n)$ repre-

sents 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;

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla f(\mathbf{w}(n))]^T \quad (1)$$

where μ is the step size. The LMS algorithm uses an instantaneous value of gradient in place of its ensemble average so that we have the following update equation [7];

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{u}(n) \quad (2)$$

where the true gradient $[\nabla f(\mathbf{w}(n))]^T$ has been replaced by the instantaneous estimate $-2e(n)\mathbf{u}(n)$. Note that $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{u}(n)$ is the error signal and μ is the step size. 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.

2.1 Full Conjugate Gradient Algorithm

The full conjugate gradient algorithm is based on updating the tap weights with new directions that are "non-interfering", in otherwords, conjugate to each other. 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 its Taylor series as;

$$\begin{aligned} f(\mathbf{x}+\Delta\mathbf{x}) &\approx f(\mathbf{x}) + \sum_i \frac{\partial f(\mathbf{p})}{\partial x_i} x_i + \frac{1}{2}\sum_{ij} \frac{\partial^2 f(\mathbf{p})}{\partial x_i \partial x_j} x_i x_j + \ldots \\ &= c + \Delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{b} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} \cdot \mathbf{Q} \cdot \Delta\mathbf{x} \end{aligned} \quad (3)$$

where

$$c \equiv f(\mathbf{x}) \qquad \mathbf{b} \equiv \frac{\partial f(\mathbf{x})}{\partial x_i} = \nabla f(\mathbf{x}) \qquad \mathbf{Q} \equiv \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \nabla^2 f(\mathbf{x}) \quad (4)$$

The matrix **Q** is the $m \times m$ Hessian matrix of the function at **x,** and **b** is the $m \times 1$ gradient of the function at **x**. The approach in the conjugate direction method is to obtain a set of linearly independent direction vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_{m-1}$ which are conjugate with respect to **Q** so that the optimum solution vector $\mathbf{x}^o$ minimizes (3). $\mathbf{x}^o$ can be expressed as;

$$\mathbf{x}^o = \alpha_0 \mathbf{x}_0 + \alpha_1 \mathbf{x}_1 + \ldots + \alpha_{m-1} \mathbf{x}_{m-1} \quad (5)$$

and the constants are given by [1];

$$\alpha_i = \frac{\mathbf{x}_i^T \mathbf{b}}{\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i} \quad (6)$$

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 **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 [8], or as an online method described in this paper.

By modifying the CGA for a *nonquadratic* function as given in references [1][2] or [9], we can derive the following algorithm that does not require the knowledge of either the Hessian or a line minimization along a particular conjugate direction;

Full Conjugate Gradient Algorithm:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

and repeat *Step 2 (a)*.

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

The calculation of β_k is done according to the Fletcher-Reeves method rather than the Polak-Ribiere method [1] since it tends to give a smoother convergence. The penalty for not having to calculate the Hessian matrix is that *two* gradient calculations must be performed per iteration, one at $\mathbf{w}_{k+1}$ and one at $\mathbf{y}_{k+1}$ however, since the computation of the Hessian matrix is of order $O(m^2)$ and the calculation of a single gradient is of order $O(m)$ [13], the savings are substantial if the filter order m is large.

2.2 Windowed Conjugate Gradient Algorithms

In the CGA algorithm above, it is assumed that the gradient calculations are true gradients and that at least m conjugate directions can be calculated. However, this is expensive and sometimes impractical if real time processing is required. If we use an instantaneous gradient estimate, as is done in the LMS algorithm, the CGA will terminate in one step. This is because there will not be any more directions conjugate to the initial direction vector. However, a better approximation to the gradient can be obtained by calculating the estimate based on a *window* of past values of inputs. If we construct a gradient estimate by averaging the instantaneous gradient estimates over a specified number n_w of past values, there will be at least n_w lin-

early dependent direction vectors in the gradient estimate. Specifically we replace the instantaneous estimate by a windowed estimate as follows;

$$[\nabla f(\mathbf{w}(n))]^T = \mathbf{g}(n) \approx \left(\frac{2}{n_w}\right)\left[\sum_{i=0}^{n_w-1} \{[\mathbf{w}_0{}^T(n)\mathbf{x}(n-i) - d(n-i)]\mathbf{x}(n-i)\}\right] \quad (18)$$

With this gradient estimate, the CGA will terminate in n_w loops of *Step 2*. Note that the same type of windowed gradient calculation will be used in (7),(9),(13), and (15) except that in (9) and (15) the vector **y** is used in place of **w**.

2.2.1 Simplification of the Algorithm

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 (12) requires $2mn_w$ multiplies and one division for a window of size n_w. This step size can be replaced by a constant value as done in reference [9] or with a normalized step $\bar{\alpha}$ size as is done here according to the following equation;

$$\bar{\alpha} = \frac{\gamma}{\varepsilon + \|\mathbf{x}(n)\|^2} = \frac{\gamma}{\varepsilon + \mathbf{x}^T(n)\mathbf{x}(n)} \quad (19)$$

where

γ is a number between 0 and 2.

ε is a small positive number to prevent the denominator from going to zero.

By removing the calculation of α_k, the calculation of **p** and **y** are also no longer required, thus simplifying the algorithm further. The α_k are therefore replaced by a normalized step size according to (19). The resulting algorithm obtained by combining the CGA with a windowed gradient calculation and normalized step size is called the *Normalized Fast Conjugate Gradient Algorithm* (NFCGA).

Normalized Fast Conjugate Gradient Algorithm:

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

Step 1.

a) Starting with an initial weight vector $\mathbf{w}_0$, compute the following gradient estimate;

$$\mathbf{g}_0 = [\nabla f(\mathbf{w}_0)]^T = \left(\frac{2}{n_w}\right)\left[\sum_{i=0}^{n_w-1}\{[\mathbf{w}_0{}^T(n)\mathbf{x}(n-i) - d(n-i)]\mathbf{x}(n-i)\}\right] \quad (20)$$

b) set $\mathbf{d}_0 = -\mathbf{g}_0$ (21)

c) compute the normalized step size parameter $\bar{\alpha}$ according to (19);

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

a) set $\mathbf{w}_{k+1} = \mathbf{w}_k + \bar{\alpha}\mathbf{d}_k$ (22)

b) Compute an estimate of the gradient $\mathbf{g}_{k+1}$ at $\mathbf{w}_{k+1}$;

$$\mathbf{g}_{k+1} = [\nabla f(\mathbf{w}_{k+1})]^T = \left(\frac{2}{n_w}\right)\left[\sum_{i=n-n_w+1}^{n}\{[\mathbf{w}_{k+1}{}^T(n)\mathbf{x}(i) - d(i)]\mathbf{x}(i)\}\right] \quad (23)$$

c) Unless k=m_w-1, set $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k\mathbf{d}_k$,where; (24)

$$\beta_k = \frac{\mathbf{g}_{k+1}^T\mathbf{g}_{k+1}}{\mathbf{g}_k^T\mathbf{g}_k} \quad (25)$$

Note that if $\beta_k > 1$, *go directly to Step three.*

Repeat *Step 2 a).*

Step 3. Replace $\mathbf{w}_0$ by $\mathbf{w}_k$, shift in a new value to the input data vector $\mathbf{x}$, and go back to *Step 1*.

It should be pointed out that checking for $\beta_k > 1$ is a necessary step in the simplified version of this algorithm. As shown in [11], linearly filtered noisy gradient algorithms can provide faster convergence than (2) however, when $\mathbf{g}_k$ are 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 make the algorithm unstable. Proakis [11] 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;

$$0 < \beta_k < 1$$
$$0 < \alpha < \frac{2(1+\beta_k)}{\lambda_{max}} \quad (26)$$

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 α from $2/\lambda_{max}$ to $2(1+\beta_k)/\lambda_{max}$ but β_k must be kept below 1.

2.3 Complexity

The choice of n_w =1 implies no averaging in the gradient estimate and the NFCGA reverts to the NLMS algorithm. For higher values of n_w the complexity approaches that of the RLS algorithm. The complexity of the NFCGA is $O(mn_w^2)$ since in *Step 2*, the weights are updated n_w times per iteration and the calculation of the averaged gradient is $O(mn_w)$. It is interesting to note that for a window size n_w=3, the complexity is $O(9m)$ is approximately equal to that of the Fast RLS algorithm $O(8m)$ [10]. By comparison, the standard RLS algorithm has complexity $O(m^2)$.

3.0 SIMULATIONS

In this section, we apply the NFCGA 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. The input to the system is noise with a variance of 1.0. Uncorrelated noise is added to the system output. The system is illustrated in Figure 1. Convergence curves showing the Normalized Mean Squared Error (NMSE) are obtained by averaging 200 independent trials. The NLMS and NFCGA algorithms both use a step size parameters $\gamma = 0.5$ and the RLS uses a forgetting factor $\alpha = 0.99$. The results illustrated in Figure 2 show that for the white noise input, the NFCGA converges somewhere between the rate of the NLMS and RLS algorithms, depending on the size of the gradient averaging window. Figure 3 shows the results when the input is coloured by a first order autoregressive process according to $y(n) = 0.9y(n-1) + v(n)$ where $v(n)$ is a unit variance white noise sequence. As can be seen from both Figure 2 and Figure 3, the choice of n_w determines the convergence rate. In all cases, the NFCGA algorithm consistently outperforms the NLMS algorithm.

4.0 CONCLUSIONS

This paper has introduced a variant of the Partial Conjugate Gradient Algorithm based on using a gradient averaging window and normalized step size to replace the optimum step size. The NFCGA has reduced complexity as compared to the regular CGA but still has fast convergence even with low values of gradient

averaging window. Simulations illustrate that the NFCGA performance is between the NLMS and RLS performance depending on the size of the gradient averaging window. The NFCGA can be extended to nonlinear neural networks to improve convergence.

REFERENCES

[1] D. G. Leunberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, 1973

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

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

[4] S. Ergenzinger, E. Thomsen, " An accelerated learning algorithm for multilayer perceptrons: optimization layer by layer", *IEEE Trans. Neural Networks*, Vol. 6, No. 1, pp. 31-42 , Jan. 1995.

[5] R. Battiti, "First and second order methods for learning: Between steepest descent and Newtons method", *Neural Computation*, Vol. 4, No. 2, pp 141-166, 1992.

[6] A.N. Birkett, R. A. Goubran, "Acoustic Echo Cancellation Using NLMS-Neural Network Structures", Proceedings *ICASSP*, pp 3035-3038, Detroit, MI, 1995.

[7] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall, 1986.

[8] 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.

[9] 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.

[10] 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. editiors, Elsevier, pp. 417-420, 1990.

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

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

[13] W. Buntine, A. A. Weigend," Computing Second Derivative in Feed-Forward Networks: A review"; *IEEE Trans. Neural Networks*, Vol. 5, No. 3, pp. 481-488, May 1994.

ILLUSTRATIONS

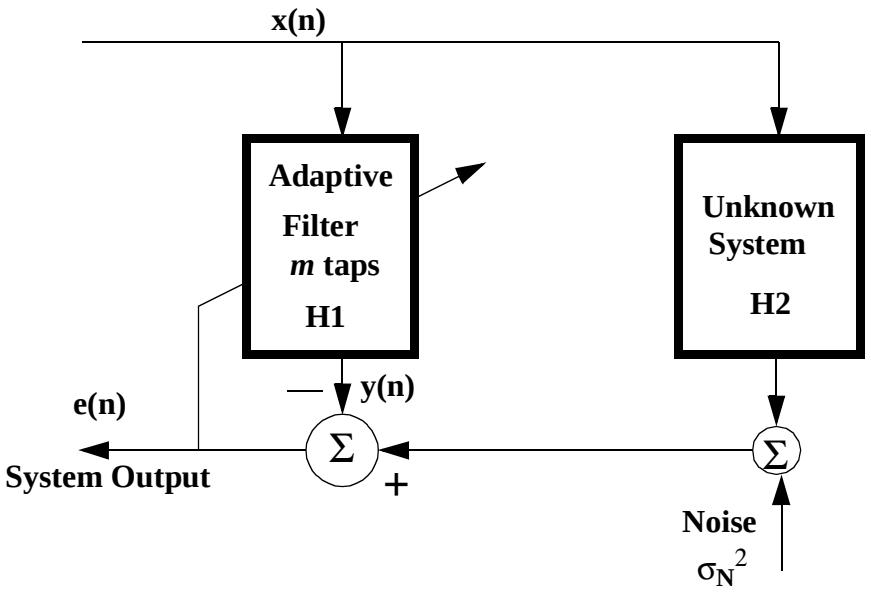

FIGURE 1. System identification model. The input x(n) is either a white noise source w(n) or a first order autoregressive process according to x(n)=0.9x(n-1)+w(n). The unknown channel consists of an exponentially decaying impulse 50 taps long. H1 is a 50 tap transversal filter which is updated according to the NLMS, RLS or NFCGA algorithm. An uncorrelated noise source with variance σ_N^2 is added to the adaptive filter output y(n).

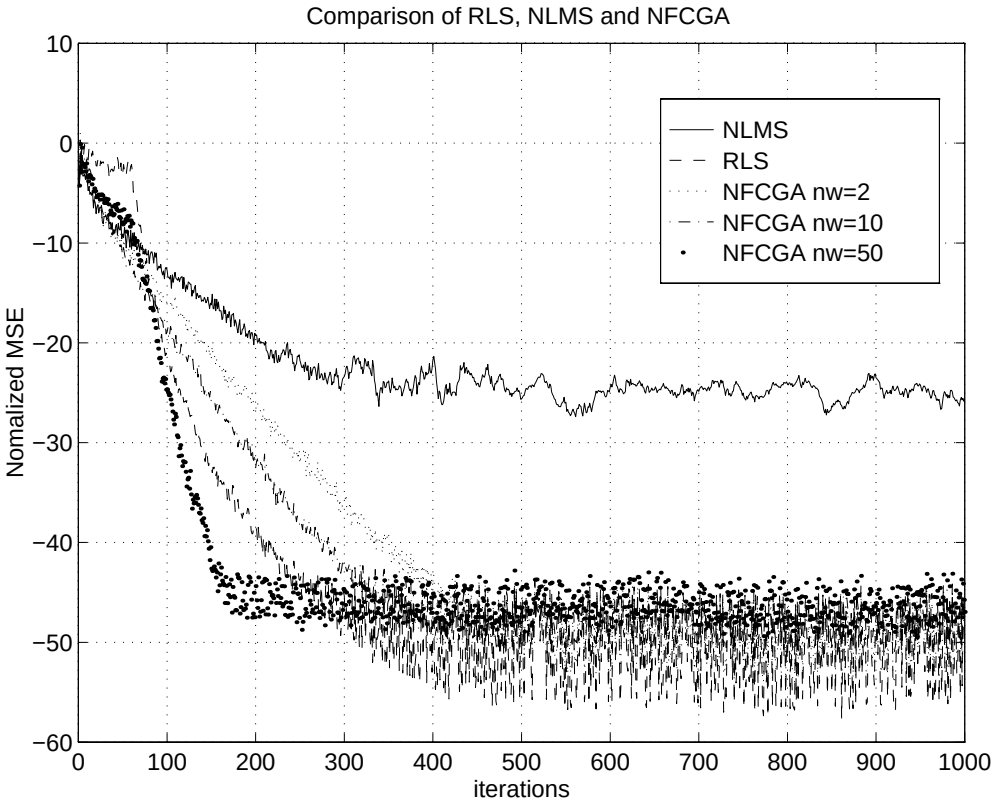

FIGURE 2. Normalized MSE when using a white noise source to model the input signal x(n). Two hundred independent trials are used in the averaging process.

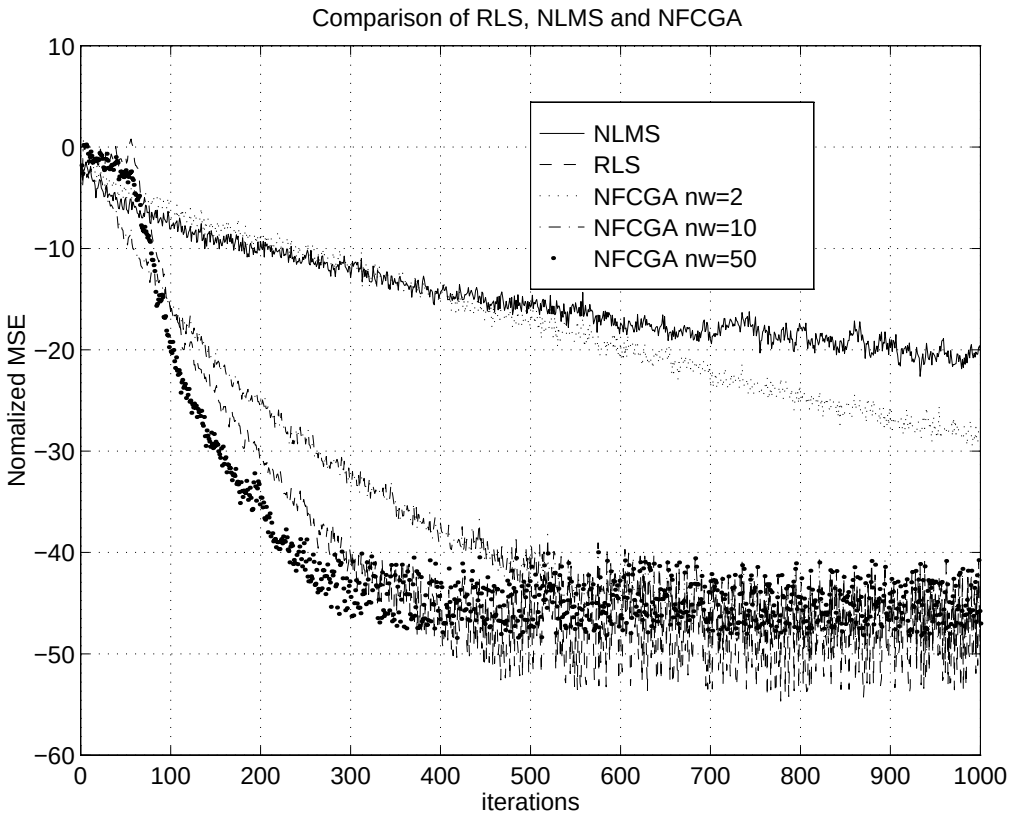

FIGURE 3. Normalized MSE when using a first order autoregressive signal to model the input signal x(n). Two hundred independent trials are used in the averaging process