BLOCK ADAPTIVE IIR FILTERS USING PRECONDITIONED CONJUGATE GRADIENTS FOR ORTHOGONALIZATION

Andrew W. Hull and W. Kenneth Jenkins
The Department of Electrical and Computer Engineering
and The Computer and Systems Research Laboratory
University of Illinois at Urbana-Champaign
1308 West Main Street
Urbana, IL 61801
USA

ABSTRACT

The method of Preconditioned Conjugate Gradients (PCG) is introduced as an accelerator for simple IIR algorithms to significantly increase their rate of convergence without dramatically adding to their complexity. This paper presents the IIR extension of [1], and develops a block Newton-type adaptive IIR digital filter with complexity O(log(N)). The proposed algorithm exploits a structured approximation to the Hessian which permits the application of a fast preconditioned conjugate gradient optimization method. In this novel formulation, the identification problem of the IIR coefficients separates into two subproblems, each of which may be solved by application of fast adaptive FIR techniques. Present IIR algorithms require greater computational cost, or converge more slowly. It is the adoption of fast PCG which permits the development of an O(log(N)) adaptive algorithm. The PCG method manipulates an approximation of the Hessian matrix to form an orthogonalizing update term for the IIR LMS algorithm. Rapid convergence follows, and the method is robust with respect to fixed-point instability.

The use of preconditioned conjugate gradients in the Gauss-Newton update leads naturally to the application of the planar least squares inverse to bound the poles of the adaptive system by projecting an unstable denominator onto a stable polynomial. This technique is invoked whenever the output of the adaptive filter exceeds a certain threshold. This approach provides a computationally efficient means to ensure robust IIR adaptive behavior.

1. INTRODUCTION

The use of an Infinite Impulse Response (IIR) adaptive filter structure is a strategy related to block processing which attempts to reduce the computational burden of high-order adaptive FIR filters. The presence of feedback generates an impulse response having large support with significant nonzero magnitude while using substantially fewer parameters than an equivalent FIR adaptive filter. This "parsimony principle" has fueled an interest in IIR adaptive filters which has yet to contribute significantly in practice.

The IIR adaptive filtering problem to be considered here consists of two input signals, x(n), the input signal, and d(n), the desired or "template" signal. The objective of the adaptive filter is to adjust the coefficients of the filter, w(n), so that the error signal, e(n), is minimized. The solution, w, for the optimal filter weights is given by a set of normal equations,

$$\mathbf{R}\,\mathbf{w} = \mathbf{p}, \qquad (1)$$

of common structure. The matrix, **R**, is the autocorrelation of the regressor signal. The vector, **w**, contains the coefficients of the adjustable filter, and **p** is the cross-correlation between the regressor and desired signals.

The object of an adaptive filter is to search over an error surface and locate the values of **w** which yield the minimal error. The properties of the error surface affect the performance of the optimization procedure used by the adaptive filter. For the case of an IIR adaptive filter, the error surface may have several minimizing solutions and possibly also local minima. About those points the surface is well approximated by a quadratic surface [2]. This last fact suggests the use of a Newton-type, or self-orthogonalizing algorithm,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{R}^{-1}\,e(n)\mathbf{x}(n), \qquad (2)$$

to adjust the values of the IIR filter.

The primary contributor to the cost of orthogonal updating is the manipulation of the autocorrelation matrix, **R**. From (2) it can be seen that **R** must be updated, inverted, and used to form a matrix-vector product at each iteration. This represents computations of $O(N^3)$, primarily due to the matrix inversion. Much of the adaptive filtering literature represents a search for less computationally expensive methods of forming $\mathbf{R}(n)^{-1}\,\mathbf{x}(n)$, the so-called "Kalman" gain. The approach examined here is to use the iterative Preconditioned Conjugate Gradient method (PCG) to compute the Kalman gain. The PCG method forms the Kalman gain by solving the system

$$\mathbf{R}(\mathbf{n})\,\mathbf{k}(\mathbf{n}) = \mathbf{x}(\mathbf{n}), \qquad (3)$$

where **k**(n) is a dummy variable representing the Kalman gain. The application of iterative algorithms and, in particular, the preconditioned conjugate gradient method, in this self-orthogonalizing configuration has been overlooked in the adaptive filtering literature.

2. FAST GAUSS-NEWTON ALGORITHMS

"Fast" self-orthogonalizing FIR adaptive algorithms using PCG were developed in [1] by exploiting several properties of circulant matrices and their relationship to Toeplitz forms. The complete quasi-Newton algorithm consists of a series of blocks appended to BLMS. The necessary operations for BLMS consist of steps I, II, III, and V, shown in Figure 1. They consist of the block computation of the adaptive filter output, y, and then the block computation of the average gradient. For an orthogonal algorithm, it is then necessary to compute the autocorrelation lags, $r_x(n)$, which is step IV in Figure 1. The operations of step VI follow by rewriting the correlation estimator in a recursive fashion as a convolution. Finally, the low-computation PCG method is used to compute the Kalman gain by solving a "dummy" system of equations exploiting the assumed Toeplitz structure of **R**.

The autocorrelation matrix, **R**, for an IIR adaptive filter is not Toeplitz. An approximate term which provides effective orthogonalization, particularly near a minimizing solution where the error surface may be accurately modeled as quadratic, employs the Hessian, **H**. This matrix has the form

$$\mathbf{H}(\mathbf{n}) = \begin{pmatrix} \mathbf{R_{xx}} & \mathbf{R_{xy}}(\mathbf{n}) \\ \mathbf{R_{xy}}(-\mathbf{n}) & \mathbf{R_{yy}}(\mathbf{n}) \end{pmatrix}, \qquad (4)$$

where both $\mathbf{R}_{xy}$, and $\mathbf{R}_{yy}$ depend upon on the dynamics of the adaptive filter. Research to fashion IIR algorithms with low complexity generally approximate **H**, and attempt to strike a balance between slower convergence and fewer computations.

The discretization of elliptic PDEs often results in a system of linear equations with a coefficient matrix whose structure is identical to (4). These are typically then solved by the conjugate gradient method. To increase the rate of convergence, a preconditioner is typically employed having the form [3]

$$\mathbf{C} = \begin{bmatrix} \mathbf{M} & 0 \\ 0 & \mathbf{N} \end{bmatrix}. \qquad (5)$$

This immediately suggests approximating **H** by its diagonal Toeplitz components,

$$\mathbf{H}(\mathbf{n}) = \begin{pmatrix} \mathbf{R_{xx}} & 0 \\ 0 & \mathbf{R_{yy}} \end{pmatrix}, \qquad (6)$$

and then employing the PCG method *twice* - once for the subproblem with **Rxx**, and once using **Ryy** - to compute the update, $\mathbf{H}(\mathbf{n})^{-1}\nabla(\mathbf{k})$. The matrix, **C**, follows from (5), and **M** and **N** are circulant approximations of $\mathbf{R}_{xx}$ and $\mathbf{R}_{yy}$. This update is then used in a Gauss-Newton algorithm, denoted FGN,

$$\mathbf{w}(\mathbf{n}) = \mathbf{w}(\mathbf{n}-1) + \mu\, \mathbf{H}(\mathbf{n})^{-1}\nabla(\mathbf{k}). \qquad (7)$$

As in the FIR case, $\mathbf{R}_{xx}$ and $\mathbf{R}_{yy}$ are Toeplitz in structure only in special cases.

3. BLOCK IIR FILTERING

A potentially more lucrative advantage of the IIR FGN algorithm is the ability to exploit block processing. Block IIR adaptive filtering represents the culmination of the promise of IIR adaptive filtering: significantly large filter impulse responses computed using convolutional operators of moderate length, and whose performance is relatively unchanged by input correlation. The block IIR filter is a straightforward generalization of the block FIR filter [4], where the coefficients are formed into infinite-dimensional matrices, and the data occupy columns, as

$$\mathbf{A}\,\mathbf{y} = \mathbf{B}\,\mathbf{x}. \qquad (8)$$

The input vectors are partitioned into sections of L samples, where L $\geq$ { Na + 1, Nb }. **A** and **B** are also partitioned into L x L submatrices. Equation (8) may be written as a singly infinite problem, and following algebraic manipulation has the form:

$$\mathbf{y}(\mathbf{k}) = \mathbf{K}\,\mathbf{y}(\mathbf{k}-1) + \mathbf{H}_0\,\mathbf{x}(\mathbf{k}) + \mathbf{H}_1\,\mathbf{x}(\mathbf{k}-1).$$

It can be shown that all of the matrix operators have convolutional form. Thus a block of output, y(k), can be computed with three FFTs, requiring the storage of three blocks of data.

The block nature of computing the output, y(k), introduces a new element to the IIR nature of the filtering problem. The poles of the block IIR system are $(p_i)^L$, where p_i is a pole of the sequential IIR filter. This suggests that block IIR adaptive filters may be more unstable than sequential IIR counterparts. This behavior is not widely observed in simulation. The averaging of the descent directions in the block FGN algorithm compensates for the increased instability. The convergence behavior of block IIR algorithms degrades analogously to block FIR algorithms: performance is equivalent for white noise, but the step size must be reduced as the input becomes more correlated. This results in slowed convergence.

By resorting to a block adaptive framework it becomes computationally feasible to project the poles of the IIR adaptive filter (the zeros of $A(z^{-1})$) within the unit circle, as necessary. Rather than costly pole monitoring – which motivates the use of the parallel form and the second-order section IIR filter structure – bounds may be placed on the magnitude of the output of the adaptive filter . When an element of **Y**(k) exceeds a threshold, it is concluded that one or more zeros of $A(z^{-1})$ exceed unity. The double planar least-squares inverse of $A(z^{-1})$ will then be used to compute **Y**(k), instead of $A(z^{-1})$ itself.

The planar LS inverse, $B(z^{-1})$, of order M of an Nth degree polynomial, $A(z^{-1})$, is determined from their convolution $A(z^{-1})B(z^{-1}) = C(z^{-1})$, which is chosen to minimize the least squares criterion [5]

$$J = (1 - c_0)^2 + \sum_{k=1}^{M+N} c_k^2 .$$

The unknown parameters, b_i, are determined by setting to zero their partial derivatives with respect to J. If b_0 and a_0 are set to unity, this generates the set of linear equations

$$\begin{pmatrix} q_0 & q_1 & q_2 & \cdots \\ q_1 & q_0 & q_1 & \cdots \\ q_2 & q_1 & \ddots & \ddots \\ \vdots & \ddots & \ddots & q_0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_M \end{pmatrix}. \qquad (9)$$

The q_k are "autocorrelation lags" given by

$$q_k = \sum_{i=0}^{M} a_{i+k}\, a_i,\; k = 0, ..., M . \qquad (10)$$

The system matrix in (9) is Toeplitz, and its elements are determined from a series of convolutions. Robinson shows [6] that the double planar least squares inverse is minimum phase. Thus it is sufficient to solve (9) twice, using fast PCG, to restore $A(z^{-1})$ to a minimum phase polynomial. Unfortunately, the projection operation performed using the double planar LS method perturbs even the minimum phase component of a mixed-phase system. When employed following every block parameter update, the effect of this projection operation is to raise the noise floor to unacceptable levels, typically less than -20 dB. This motivates the output monitoring approach explored herein.

4. COMPUTER SIMULATION

Computer simulations were used to clarify the effectiveness of the Hessian approximation (6) in the Gauss-Newton algorithm (7). Figure 2 shows the results of a system identification experiment where the unknown plant was a lowpass filter, having poles at $z = 0.75$ and $z = 0.65$, and one zero at $z = 1.0$. The input was colored noise. The three traces compare IIR LMS with the learning curves of (7) using the full Hessian, and the approximation (6). The use of the approximate Hessian did not change the convergence "rate", but did delay the algorithms descent into the error surface "bowl". This slight delay in the onset of convergence makes possible a reduction in computational complexity from $O(N^2)$ to $O(N\log(2N))$. These computations in the FPCG algorithm are numerically robust, while the $O(N^2)$ computations of the full Hessian algorithm require use of the matrix inversion lemma. Use of that technique usually requires the use of floating-point arithmetic. Through the choice of preconditioner, the IIR FGN algorithm also permits easier incorporation of *a priori* information than the GN algorithm. The preconditioner, k1, of [7] may be modified to incorporate knowledge of the location of the poles, or other a priori information of the system structure to increase further the convergence rate.

The performance of block IIR adaptive filters and the use of the IIR FPCG GN algorithm were similarly explored via simulation. Figure 3 plots the learning curves of block GN, block IIR FGN, and IIR BLMS. The block update scheme delays further the onset of convergence of the Gauss-Newton algorithms using the full and approximate Hessian, but changes little the rates of convergence of the two algorithms

The ability of output monitoring coupled with planar LS projection was also explored via computer simulation. In all simulations, divergence was detected and the adaptive system restored to a minimum phase condition. The algorithm continued along a stable trajectory. The results were most dramatic when the "unprotected" IIR algorithm experienced explosive divergence. In these cases, no unusual behavior was even noted in the IIR algorithm monitoring output magnitude and projecting the system poles as necessary. A striking example is shown in Figure 4. Here the unknown plant had an individual pole at $z = 0.98$. The IIR LMS algorithm failed to converge, but periodic application of planar LS prevented explosive divergence.

5. CONCLUSTIONS

The central theme of this work has been to explore a technique which reduces the dependence of simple adaptive algorithms on the correlation of the input signal. The "fast" method of preconditioned conjugate gradients was used to iteratively construct the Kalman gain, introducing a family of fast Gauss-Newton algorithms. These algorithms employ an iterative technique to perform the matrix manipulation necessary for an orthogonal adaptive update. This results in superior computational efficiency compared with those of conventional fast LS algorithms. Their complexity is comparable to that of block LMS algorithms, but their self-orthogonalizing behavior permits much more rapid convergence. Furthermore, with an iterative process the filter parameters are valid after any iteration. Thus processing may be interrupted momentarily to dedicate the ALU to more important tasks without serious harm to convergence or other algorithm performance. This approach may, for example, permit architectures differing from the typical one processor per channel configuration used in array processing systems. These benefits of iterative calculation naturally extend to two or more dimensions. This permits the development of fast multidimensional processing algorithms using fast 2-D PCG based adaptive filters.

As in the FIR case, the computational reduction of the FPCG GN approach is directly connected with its ability to replace matrix-vector operations with convolutional operators. This exhaustive application of convolutional structures may be contrasted with previous attempts at fast adaptive algorithms which seek computational reduction by eliminating redundancy and using structured, fast algorithms. By employing an iterative algorithm, it is possible to exploit most fully convolutional structures, permitting $O(\log(N))$ operations. Computation is significantly reduced in applications not requiring rapid tracking of unknown parameters and tolerate block processing.

The iterative computation of the new algorithm facilitates the easy passing information from one block to the next. This ability to pass the previous value of the filter parameters to the present block is of particular interest in several applications, particularly speech processing and disk-drive read/write channel equalization. Each of these scenarios is intrinsically oriented to block processing. The passing of channel parameters is inherently difficult in direct solution techniques, frequently introducing bias. The PCG-GN algorithm suffers from no such difficulties.

REFERENCES

[1] A.W. Hull, W. K. Jenkins, "Iterative Calculation of the Kalman Gain for Self-Orthogonalizing Adaptive Algorithms," Submitted to *IEEE Transactions on Signal Processing.*

[2] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification.* Cambridge: MIT Press, 1983.

[3] G.H. Golub and C.F. Van Loan, *Matrix Computations.* Baltimore, MD: The Johns Hopkins University Press, 1983.

[4] R. King et al., *Digital Filtering in One and Two Dimensions.* New York: Plenum Press, 1989.

[5] N.K. Bose, *Digital Filters.* New York: North-Holland, 1985.

[6] E.A. Robinson, *Statistical Communication and Detection.* New York: Hafner, 1964.

[7] T. Ku and C. Kuo, "Design and analysis of Toeplitz preconditioners," *IEEE Trans. Sig. Proc.*, vol. 40, no. 1, pp. 129-141, Jan. 1992.

```
I. Collect: x(n), d(n)

II. Compute y(k)
    $x_b = [x^r_{k-1}\ x^r_k]$
    $X = FFT(x_b)$
    $w_b = [w\ \underline{0}]$
    $W = FFT(w_b)$
    $Y = X .* W$
    $y = FFT^{-1}(Y)$

III. Compute ∇(k)
    $e_b = [\underline{0}\ e(k)]$
    $E = FFT(e_b)$
    $G = X .* E$
    $\nabla(k) = FFT^{-1}(G)$

IV. Compute R(k)
    $x_1 = [\underline{0}\ x^r_k]$
    $X_1 = FFT(x_1)$
    $\Delta_{rx} = X .* X_1$
    $\delta_{rx} = FFT^{-1}(\Delta_{rx})$
    $\delta'_{rx}(k) = \delta'_{rx}(k-1) + \delta_{rx}/L$
    $r_x(k) = (1/k)\,\delta'_{rx}(k)$
    $R(k) = Toeplitz(r_x(k))$

V. Compute Kalman gain
    Solve: $R(k)\,g(k) = \nabla(k)$
        using FPCG

VI. Update Filter Weights
    $w(k) = w(k-1) + \mu\, g(k)$
```

Figure 1 FAST PCG QUASI-NEWTON ALGORITHM

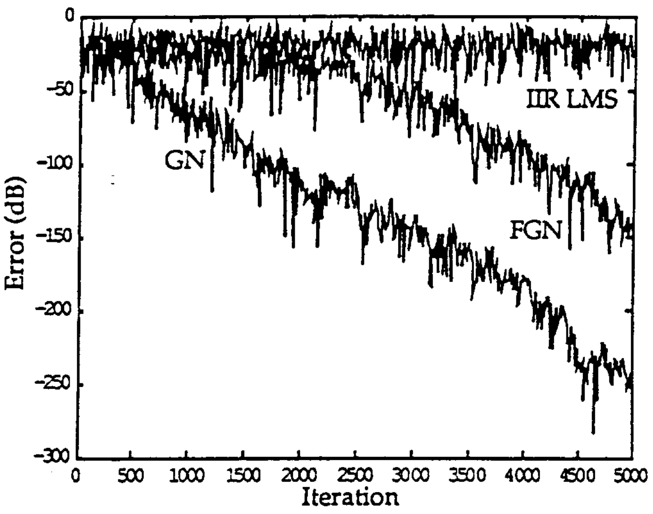

Figure 3 BLOCK IIR SIMULATION: GN, FGN, LMS

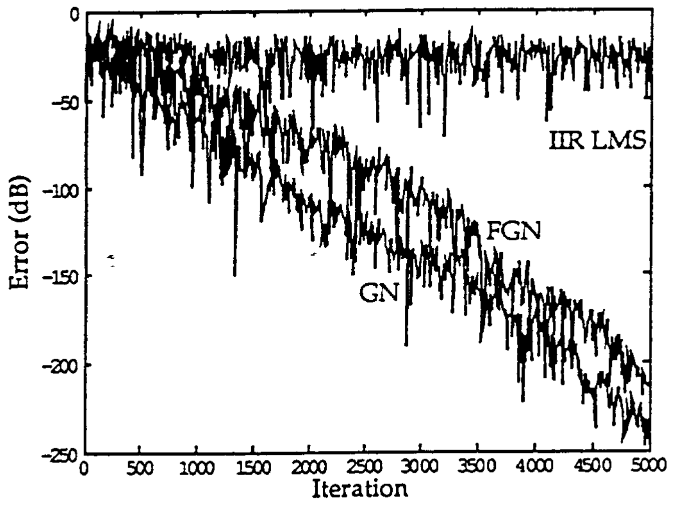

Figure 2 IIR Simulation: GN, FGN, LMS, clr #1

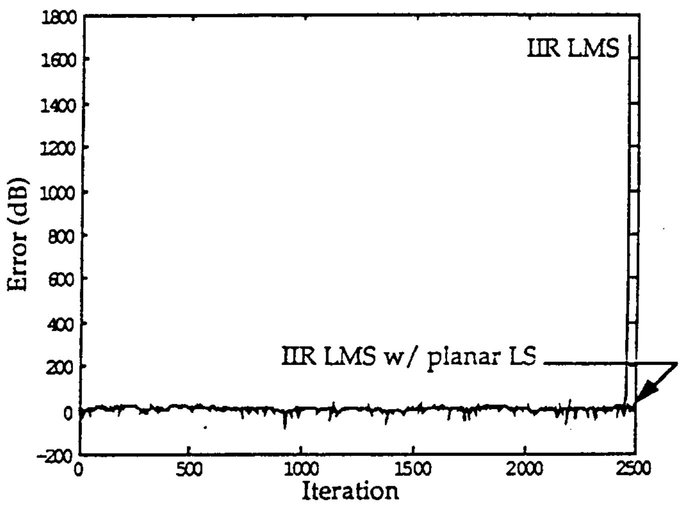

Figure 4 IIR SIMULATION: LMS without PLS, LMS with PLS