A Recursive Block Incomplete Factorization Preconditioner for Adaptive Filtering Problem

Shazia Javed
School of Mathematical Sciences
Universiti Sains Malaysia
11800 Penang, Malaysia
shaziafateh@hotmail.com

Noor Atinah Ahmad
School of Mathematical Sciences
Universiti Sains Malaysia
11800 Penang, Malaysia
atinah@cs.usm.my

Copyright © 2013 Shazia Javed and Noor Atinah Ahmad. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

In this paper a recursive incomplete factorization preconditioner for the iterative solvers of adaptive filtering problem is proposed. The preconditioner is able to reduce the eigenvalue spread of the input signals. The technique involves QR factorization of a toeplitz block sub matrix of the input data matrix. Incomplete factorization is realized by block-diagonalization of the upper triangular factor. The preconditioner obtained by update of the incomplete upper triangular factor, with no fill-ins, has low computational cost as compared with complete factor update. With appropriate choice of the number of diagonal blocks, the computational cost of the resulting preconditioner can be reduced significantly. Simulations results show noticeable decrease in the spectral condition number of the autocorrelation matrix by the application of resulting preconditioner.

Mathematics Subject Classification: 65F08
Keywords: Incomplete Cholesky’s factor, QRD-RLS, computational complexity

1. Introduction

Adaptive filtering problem can be considered as an adaptive least squares problem of the form:

$$\min_{x \in \mathbb{R}^N} J_n(x) = \sum_{i=1}^{N} \lambda^{n-i} (e_n(i))^2$$

(1)

where $e_n(i)$, for $i=1,2,\ldots, n$, are the samples of error estimates obtained when the filter of order $N$ has run from $i=1$ to $i=n$, using the set of filter parameters that are computed at time $n$, while $0 < \lambda \leq 1$ is the forgetting factor [1]. Methods used for the solution of adaptive filtering problem are either direct or iterative[7]. Direct methods include RLS based algorithms and have fast convergence speed, but are computationally intensive. Iterative methods, including LMS based algorithms, are computationally simple, but their convergence is highly dependent on the spectral condition number of the input autocorrelation matrix. Two of the most important characteristics to judge the performance of an algorithm are convergence rate and computational cost [2,3]. If the spectral condition number is high, algorithm takes more iteration to access a steady state solution and converges slowly. In such a situation, preconditioning is necessary to make the algorithm robust by decreasing spectral condition number. It means that a preconditioned algorithm is required to have convergence rate of direct method, but with low computational cost. A robust direct method is QRD-RLS algorithm [8] which involves rank one update of the data matrix. A new block-adaptive QRD-RLS algorithm was presented in [6] by introducing a non-orthogonal transformation into recursive update. This algorithm adapts an upper triangular block diagonal matrix.

Some well known preconditioners for adaptive filtering algorithms include preconditioners derived directly from the autocorrelation matrix [4,5] and clamped lattice preconditioner [2].

In this paper, we are intending to develop a computationally simple incomplete factorization preconditioner for the iterative solvers of adaptive filtering problem which should be able to reduce the spectral condition number of the input autocorrelation matrix. The preconditioner is formed by recursive update of the block upper triangular diagonalization of the upper triangular QR-factor. Rank one update of the incomplete QR factors improves the robustness of the preconditioner. The sparsity pattern with no fill-ins significantly reduces the computational complexity of the resulting preconditioner.

2. Incomplete QR Factorization

Consider the standard least squares problem of the form:
A recursive block incomplete factorization preconditioner

\[
\min_{x \in \mathbb{R}^n} \|b - Ax\|^2.
\]

Where \( A \) is a \( n \times N \) (\( n \geq N \)) full rank matrix, with the corresponding normal equation

\[
A^T Ax = A^T b
\]

Since the condition number of the coefficient matrix of (2) is the square of that of \( A \), therefore a slight increase in condition number of \( A \) can make the normal equation very ill-conditioned. In such a situation, preconditioning is necessary for robustness. Incomplete factorization provides a good factorization preconditioner to reduce the condition number and improve the convergence speed of the system. Incomplete QR factorization is an approximation of complete QR factorization of the form:

\[
A = QR + E
\]

Where \( Q \in \mathbb{R}^{n \times n} \) may not have orthonormal columns, \( R \in \mathbb{R}^{n \times N} \) is an upper triangular matrix and \( E \in \mathbb{R}^{n \times N} \) is the error matrix which can be made as sparse as we please. The matrix \( R \) here is the Cholesky’s factor of the autocorrelation matrix \( A^T A \). If \( A \approx QR \), then \( A^T A \approx R^T R \) and the matrix \( R \) can be used as a preconditioning matrix for the normal equation, that is,

\[
(R^T A^T AR^{-1})Rx = R^T A^T b
\]

Or

\[
\tilde{A}x = \tilde{b},
\]

where

\[
\tilde{A} = R^T A^T AR^{-1} = (AR^{-1})^T (AR^{-1})
\]

\[
\tilde{x} = Rx,
\]

\[
\tilde{b} = R^T A^T b = (AR^{-1})^T b.
\]

The closer the incomplete QR decomposition to the complete QR decomposition, the closer the condition number of \( \tilde{A} \) is to 1. It is important to note that \( Q \) is totally ignored here, because our interest is in the triangular factor \( R \), which is actually the transpose of the Cholesky’s factor of the autocorrelation matrix \( A^T A \). Since preconditioned autocorrelation matrix (3) involves the inverse \( R^T \) of the preconditioner matrix \( R \), therefore complexity \( O(N^2) \) of inversion of \( R \) has great importance in the computation of the preconditioner.

3. Formulation Of Adaptive Least Squares Filtering

In order to solve the adaptive least squares problem (1), we form the vectors \( a_i \in \mathbb{R}^N \) using input signals \( u(i) \) in such a way that
If vector $x_i \in \mathbb{R}^N$ is an estimate of the filter coefficient vector, $s(i) \in \mathbb{R}$ the desired signal, and, $y(i) = a_i^T x_i$ is the filter output at time $i$; $1 \leq i \leq n$, then error incurred from predicting $s(i)$ by $y(i) = a_i^T x$ is

$$e(i) = s(i) - a_i^T x_i$$

Defining

$$e_n = [e(1), e(2), \ldots, e(n)]^T$$

and

$$d_n = [s(1), s(2), \ldots, s(n)]^T,$$

error equation (4) can be written in vector form as:

$$e_n = d_n - A_n x_n$$

where $A_n$ is the $n \times N$ data matrix given by:

$$A_n = \begin{bmatrix}
    u(1) & u(2) & \cdots & u(i-N+1) \\
    u(2) & u(1) & \cdots & u(i-N+2) \\
    \vdots & \vdots & \ddots & \vdots \\
    u(n-i) & u(n-i-1) & \cdots & u(n-N)
\end{bmatrix}$$

Setting $\lambda = 1$, the problem (1) takes the form:

$$J_n(x) = \| d_n - A_n x_n \|^2$$

(5)

The toeplitiz structure of $A_n$ allows the rank-one updating of the matrix which is the key feature of the QRD-RLS algorithm [8]. Overall QRD-RLS updation step has a complexity of $O(N^2)$.

We begin with a toeplitiz square block submatrix $A_{ik}$, of size $N$, of full data matrix $A_i$ at the $k$ th sample instant. To maintain the square structure of $A_{ik}$, we are confined to the most recent $N$ input vectors only. In the next section we show that how special structure of $A_{ik}$ is manipulated to update the incomplete QR factorization of $A_{ik}$.

4. **Recursive Block Incomplete QR Factorization Preconditioner (RBIFP)**

A new block incomplete QR factorization based preconditioner is derived in this section. The approach is recursive, similar to the one employed in QRD-RLS algorithm, in conjunction with sliding windows involving rank-1 updating and downdating computations. Beginning with a toeplitiz square block submatrix $A_{ik}$, of size $N$, of full data matrix $A_i$ at the $k$ th instant, consider the sliding window
recursive least squares (RLS) computations of $\hat{A}_k$, accomplished by modifying the Cholesky’s factor.

The toeplit $N \times N$ block submatrix $\hat{A}_k$, at $k$ th instant, can be written as:

$$\hat{A}_k = \begin{pmatrix}
a_{k-N+1}^T \\
a_{k-N+2}^T \\
\vdots \\
a_k^T
\end{pmatrix}$$ (6)

The complete QR factorization of $\hat{A}_k$ gives an orthogonal $N \times N$ matrix $\hat{Q}_k$ and $N \times N$ non-singular upper triangular matrix $\hat{R}_k$ such that

$$\hat{A}_k = \hat{Q}_k \hat{R}_k$$ (7)

Such QR factorization is realized by Givens rotations, so that orthogonality of $\hat{Q}_k$ mean rows as well as columns are orthogonal. If $q_i(k)$ denotes the $i$ th row of $\hat{Q}_k$ for $i=1,2,\ldots,N$, then

$$\hat{Q}_k = \begin{bmatrix} q_1(1) & q_2(2) & \cdots & q_N(N) \end{bmatrix}^T$$

The upper triangular matrix $\hat{R}_k$ is the Cholesky’s factor of the autocorrelation matrix of $A_k$. We will recursively update it to get an approximation of the Cholesky’s factor of the autocorrelation matrix of $A_n$. There are have two mains objectives:

- To get an approximation of the Cholesky’s factor of the autocorrelation matrix of $A_n$, which serves as a preconditioner to increase the convergence rate of the iterative algorithms by decreasing the spectral condition number of the autocorrelation matrix.
- To improve the performance of the preconditioner by updating it according to new input signals, but at low computationally cost.

To overcome the problem of high computational cost of recursion, we use the block-adaptive approach of [6] to set a sparsity pattern of $\hat{R}_k$ before starting recursion.

Choose $p$ positive integers $N_1, N_2, \ldots, N_p$ such that

$$N = \sum_{j=1}^{p} N_j$$ (8)

Then partition the block diagonal of $\hat{R}_k$ into $p$ upper triangular submatrices $\hat{R}_k^{(j)}$; $(1 \leq j \leq p)$ of size $(N_j \times N_j)$ in such a way that
where \( \mathbf{R}_i^{(N_p)} \) is an \( N_j \times \left( N - \sum_{i=1}^{j} N_j \right) \) matrix, for all \( 1 \leq j \leq p - 1 \). With definitions:

\[
\mathbf{U}_i = \begin{pmatrix}
\mathbf{R}_i^{(0)} & \mathbf{O} \\
\mathbf{R}_i^{(1)} & \ddots \\
\mathbf{O} & \ddots & \ddots \\
\mathbf{O} & \ddots & \ddots & \mathbf{R}_i^{(p)}
\end{pmatrix}
\]

and

\[
\mathbf{R}_i^{(-)} = \begin{pmatrix}
0 & \mathbf{R}_i^{(N_p)} \\
0 & \mathbf{R}_i^{(N_p)} \\
\mathbf{O} & \ddots & \ddots \\
\mathbf{O} & \ddots & \ddots & 0
\end{pmatrix}
\]

We are now in a position to define the incomplete QR-factorization of \( \mathbf{A}_k \) as:

\[
\mathbf{A}_k = \mathbf{Q}_i \mathbf{U}_i + \mathbf{Q}_i \mathbf{R}_i^{(-)} = \mathbf{B}_k + \mathbf{E}_k
\]

where \( \mathbf{E}_k = \mathbf{Q}_i \mathbf{R}_i^{(-)} \) is the error matrix and is highly ill conditioned. The matrix \( \mathbf{U}_i \) is sparser and requires less computations for update as compared with \( \mathbf{R}_i \).

After \( k + 1 \) sample update, \( \mathbf{A}_k \) is updated with input vector \( \mathbf{a}_{k+1}^T \), using sliding data window of size \( N \), as displayed in figure-1. Here updating and downdating takes place at the same time. Removal of \( \mathbf{a}_{k-N+1}^T \) downdates \( N \times N \) data matrix \( \mathbf{A}_k \) to

Figure 1. Sliding Data window - updating and downdating process.
A recursive block incomplete factorization preconditioner

\[(N-1) \times N\] matrix \(\tilde{A}^{(\cdot)}\), while addition of \(a_{k+1}^T\) updates it to \(N \times N\) data matrix \(\tilde{A}_{k+1}\).

The corresponding update of incomplete matrix \(\tilde{B} = \tilde{Q} \tilde{U}\), gives

\[
\tilde{B}_{k+1} = \begin{pmatrix}
\tilde{B}^{(\cdot)}_k \\
\vdots \\
a_{k+1}^T
\end{pmatrix}
\]  

(10)

where,

\[
\tilde{B}^{(\cdot)}_k = \tilde{Q}^{(\cdot)}_k \tilde{U}_k = \begin{pmatrix}
q_k^{(2)} \\
q_k^{(3)} \\
\vdots \\
q_k^{(N)}
\end{pmatrix} \tilde{U}_k
\]

is a truncated version of \(\tilde{B}_k\) obtained by removing its first row. Using value of \(\tilde{B}^{(\cdot)}_k\), equation (10) can be written as:

\[
\tilde{B}_{k+1} = \begin{pmatrix}
\tilde{Q}^{(\cdot)}_k & \tilde{Q}_{(N+1)\cdot} \\
0_{(N+1)\cdot} & 1
\end{pmatrix} \begin{pmatrix}
\tilde{U}_k \\
a_{k+1}^T
\end{pmatrix}
\]  

(11)

It is straightforward from equation (11) that,

\[
\begin{pmatrix}
\tilde{Q}^{(\cdot)}_k & \tilde{Q}_{(N+1)\cdot} \\
0_{(N+1)\cdot} & 1
\end{pmatrix} \begin{pmatrix}
\tilde{U}_k \\
a_{k+1}^T
\end{pmatrix} = \begin{pmatrix}
\tilde{U}_{k+1} \\
a_{k+1}^T
\end{pmatrix}
\]  

(12)

The matrix on the right hand side of (12) is partially triangularized in a way that its \((N+1)th\) row consists of nonzero elements. In order to update \(\tilde{U}_k\) to \(\tilde{U}_{k+1}\), we need to zero out the last row of the right hand matrix of (12). Such an update is possible [5], since there exists and \((N+1) \times (N+1)\) orthogonal transformations matrix \(T(k+1)\) such that

\[
T(k+1) \begin{pmatrix}
\tilde{U}_k \\
a_{k+1}^T
\end{pmatrix} = \begin{pmatrix}
\tilde{U}_{k+1} \\
0^T
\end{pmatrix}
\]

The transformation matrix \(T(k+1)\), used here, is a product of Givens rotations and is orthogonal, being the product of orthogonal matrices. To preserve the sparsity pattern of \(\tilde{U}_k\) no fill-in strategy is used, which means that the transformation \(T(k+1)\) is applied in such a way that it does not change the zeros of \(\tilde{U}_k\). The update procedure is summarized in Algorithm 1.

Application of \(T(k+1)\) on left hand side of (12) generates the matrix \(\tilde{Q}_{k+1}\). Since
\[ T(k+1) \begin{pmatrix} \hat{Q}_k^{(1)T} & \mathbf{0}_{Nk} \\ \vdots & \vdots & \vdots \\ \mathbf{0}_{1(N-1)} & 1 \end{pmatrix} = \begin{pmatrix} \hat{Q}_k^T \\ \vdots \\ \mathbf{0}_{k+1}^{(1)T} \end{pmatrix} \]

is an \((N+1) \times N\) matrix having transpose of \(N \times 1\) matrix \( \hat{q}_{k+1}^{(1)} \) in the last row, equation (12) gives:

\[
\mathbf{B}_{k+1} = \begin{pmatrix} \hat{Q}_k^T & \mathbf{0}_{k+1}^{(1)} \\ \vdots \\ \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{u}}_{k+1} \\ \vdots \end{pmatrix} 
\]

After multiplication the above equation reduces to

\[
\mathbf{B}_{k+1} = \hat{Q}_{k+1}^T \hat{\mathbf{u}}_{k+1}
\]

(13)

It is worth noting that \( \hat{Q}_{k+1}^T \) is not an orthogonal matrix, hence equation (13) gives an incomplete QR-factorization at \((k+1)\)th sample instant. Since our interest is in update \( \hat{\mathbf{u}}_{k+1} \) of the upper triangle only, therefore, to avoid un-necessary computations, algorithm 1 does not involve the updation steps of \( \hat{Q}_{k} \).

**ALGORITHM 1**: No fill-in Update of \( \hat{\mathbf{u}}_{k} \):

**Initializations**: \( \hat{\mathbf{u}}_{i} \) (given by (9))

**Update**: 
for \( i = 1, 2, \ldots, t \)

Input vector :
\[
\begin{align*}
\mathbf{a}_{k+i} &= \begin{bmatrix} \mathbf{a}_{k+i}^{(1)} & \mathbf{a}_{k+i}^{(2)} & \cdots & \mathbf{a}_{k+i}^{(p)} \end{bmatrix}^T \\
\end{align*}
\]

for \( j = 1, 2, \ldots, p \)

for \( n = 1, 2, \ldots, N_{j} \)

\[
\text{denominator} = \sqrt{\left( \mathbf{R}_{k+i-1}^{(j)}(n,n) \right)^2 + \left( \mathbf{a}_{k+i}^{(j)}(n) \right)^2}
\]

\[
\begin{align*}
\hat{c}_{n}^{(j)} &= \frac{\mathbf{R}_{k+i-1}^{(j)}(n,n)}{\text{denominator}} \\
\hat{s}_{n}^{(j)} &= \frac{\mathbf{a}_{k+i}^{(j)}(n)}{\text{denominator}}
\end{align*}
\]

for \( l = n, n+1, \ldots, N_{j} \)

\[
\begin{align*}
\mathbf{R}_{k+i}^{(j)}(n, l) &= \hat{c}_{n}^{(j)} \mathbf{R}_{k+i-1}^{(j)}(n, l) + \hat{s}_{n}^{(j)} \mathbf{a}_{k+i}^{(j)}(l) \\
\mathbf{a}_{k+i}^{(j)}(l) &= -\hat{s}_{n}^{(j)} \mathbf{R}_{k+i-1}^{(j)}(n, l) + \hat{c}_{n}^{(j)} \mathbf{a}_{k+i}^{(j)}(l)
\end{align*}
\]

End
Continuing this process for \( t \) input instants, with \( 1 < t \leq n - k \), we get an incomplete block-diagonalized upper triangular matrix \( \tilde{U}_{k,t} \), which is an approximation of the Cholesky’s factor \( R_n \) of the autocorrelation matrix of input data matrix \( A_n \). Since

- The structure of \( R_n \) is such that the elements which are far away from the main diagonal are very small as compared with the elements of the diagonal and its upper band.
- The sparsity pattern of \( \tilde{U}_{k,t} \) is such that maximum number of non-zero elements are at the diagonal.
- The diagonal of \( \tilde{U}_{k,t} \) is a fair approximation of that of \( R_n \) and

\[
\text{diag}(\tilde{U}_{k,t}) \rightarrow \text{diag}(R_n) \text{ as } t \rightarrow n - k.
\]

The block diagonal submatrix \( \tilde{U}_{k,t} \) is the required incomplete QR-factorization preconditioner (RBIFP) for the adaptive filtering problem and can reduce the eigenvalue spread of the input data.

### 4.1. Complexity Analysis of RBIFP

Complexity of our preconditioner involves complexity of \( t \) updates and the complexity of the inversion of block-diagonalized preconditioner \( \tilde{U}_{k,t} \) as explained in previous sections.

From equation (8), we have

\[
\sum_{j=1}^{p} N_j^2 \leq N^2
\]

(14)

Since an update of QRD-RLS algorithm has \( O(N^2) \) complexity, update of our preconditioner has \( \sum_{j=1}^{p} O(N_j^2) \) complexity. It follows from (14) that

\[
\sum_{j=1}^{p} O(N_j^2) \leq O(N^2)
\]

Furthermore inversion of an upper triangular matrix of order \( N \times N \) has \( O(N^2) \) complexity, and we need to find inverse of our preconditioner, while applying it in some iterative method. In our formulation of diagonal subblocks, this complexity too reduces to \( \sum_{j=1}^{p} O(N_j^2) \). Hence it is justified to say that with appropriate choice of the sizes \( N_j \) and number \( p \) of the upper triangular blocks, the cost of formation of preconditioner, as well as its inversion, can be significantly reduced. The lowest complexity, equal to \( O(N) \) is reached for \( p = N \),
whereas the highest complexity is reached for $p=1$. $\frac{N}{2} < p \leq N$ is not a suitable choice for our preconditioner because in that case $\hat{U}_{k\ell}$ does not have much effect on the spectral condition number of the autocorrelation matrix of input data. Our simulation results show significant decrease in the spectral condition number for $1 \leq p \leq \frac{N}{2}$.

### 5. Simulation Results

To check the performance of our preconditioner, we use an adaptive system identification configuration. Consider a finite impulse response (FIR) filter of order $N$. A white Gaussian input signal of variance $\sigma^2 = 1$ is passed through a colouring filter with frequency response

$$H(z) = \frac{\sqrt{1-\alpha^2}}{1-\alpha z^{-1}},$$

where $|\alpha| < 1$, $\alpha$ controls the spectral condition number of the autocorrelation matrix. $\alpha = 0$ corresponds to the case when condition number is close to 1. We choose the sizes $N_j$ of the upper triangular blocks $R_{ij}^{(j)}$; $(1 \leq j \leq p)$ such that $N_j = L$ for $j = 1, 2, \ldots, p-1$, while $1 \leq N_p < N$.

Setting $N = 50$, $r = 2N$, we take $p$ such that $1 \leq p \leq \frac{N}{2}$. Figure-2 shows the computational complexity of multiplications used in the formation of the preconditioner as a function of $p$. Considerable decrease in the computational cost is observed with the increase in the number of blocks.

Now we apply our preconditioner to the autocorrelation matrix of input signal according to equation (3). Figure-3 shows the decrease in the spectral condition number of the autocorrelation matrix by the application of RBIFP as a function of $p$ for $1 \leq p \leq \frac{N}{2}$.

We have considered three types of ill conditioned matrices corresponding to $\alpha = 0.7, 0.8, 0.9$ and have observed significant decrease in the spectral condition number. It is seen that rate of decorrelation increases with the increase in the value of correlation parameter $\alpha$. For $\alpha = 0.9$ the input is highly correlated and 97% decrease in observed corresponding to $p = 2$. 
A recursive block incomplete factorization preconditioner

Figure 2. Computational Complexity of the preconditioner.

Figure 3. Percentage of reduction in spectral condition number after application of preconditioner.

6. Conclusion

A preconditioner for the iterative solvers of the adaptive filtering problem is developed by recursively updating the block-diagonalized upper triangular QR-
factor. The preconditioner RBIFP is computationally simple and reduces the eigenvalue spread of the input signals by more than 50%. Our future work concerns with developing a RBIFP preconditioned adaptive filtering algorithm with RLS-like performance at a very low computational cost.

**Acknowledgement:** The authors would like to acknowledge University Sains Malaysia (USM), Penang, Malaysia and Lahore College for Women University (LCWU), Lahore, Pakistan for financial support.

**References**


**Received: February 6, 2013**