Technique for Accelerating the Block-Cimmino Method

Mario Arioli, Iain Duff, Daniel Ruiz, and Miloud Sadkane

June 11, 2013

From

Proceeding of the Fifth SIAM Conference on
"Parallel Processing for Scientific Computing"
CHAPTER 15

Techniques for Accelerating the Block Cimmino Method

Mario Arioli$^{1,2}$
Iain Duff$^{3,4}$
Daniel Ruiz$^1$
Miloud Sadkane$^1$

Abstract. We consider acceleration techniques for the Block Cimmino iterative method for solving general sparse systems. For iteration matrices which are not too ill conditioned, the conjugate gradient algorithm is a good method for accelerating the convergence. For ill conditioned problems, the classical conjugate gradient algorithm does poorly because of clusters of eigenvalues at the ends of the spectrum of the iteration matrix. We therefore try variants of the Block Lanczos method, including the Block Conjugate Gradient method. On some test examples, these techniques are convergent whereas the conjugate gradient method is not.

1 Introduction

We consider a block projection method for the solution of the linear equations

$$\mathbf{Ax} = \mathbf{b}$$

(1.1)

where $\mathbf{A}$ is a large nonsingular sparse unsymmetric matrix of order $n$.

The blocks are obtained by partitioning $\mathbf{A}$ into strips of rows. The Block Cimmino method (Arioli, Duff, Noailles, and Ruiz 1990) projects the current iterate simultaneously onto the manifolds corresponding to the strips, and takes a convex combination of all the resulting vectors. This basic solution scheme is introduced in Section 2.

In Section 3, we introduce a particular test problem and discuss its characteristics. This test problem is used in Section 4 when testing the stability and the convergence of Block Conjugate Gradient and Block Lanczos techniques for accelerating the Block Cimmino method.

2 Block iterative methods

The blocks are obtained by partitioning the system (1.1) as

$$\begin{pmatrix} A^1 \\ \vdots \\ A^p \end{pmatrix} \mathbf{x} = \begin{pmatrix} b^1 \\ \vdots \\ b^p \end{pmatrix}$$

(2.1)

where $1 \leq p \leq n$.

If we define the Moore-Penrose pseudo inverse of $A^i$ by

$$A^{i-} = A^{iT}(A^{iT}A^{i})^{-1}$$

$^1$CERFACS, 42 av. G. Coriolis, 31057 Toulouse Cedex, France.
$^2$Istituto di Elaborazione dell’Informazione, CNR, via S. Maria 46, 56100 Pisa, Italy.
$^3$Rutherford Appleton Laboratory, OXON OX11 0QX, England.
and the projector onto the range of $A^T$ by

$$P_{RA^T} = A^T A^T,$$

we can describe the block Cimmino algorithm in the following way:

**Algorithm 2.1 (Block Cimmino Method)**

Choose $x^{(0)}$, set $k = 0$
repeat until convergence
  begin
    do in parallel $i = 1, \ldots, p$
      $$\delta^{(k)}_i = A^{(i)} b_i - P_{RA^T} x^{(k)}_i$$
      $$= A^{(i)} (b_i - A^T x^{(k)}_i)$$
    end parallel
    $$x^{(k+1)} = x^{(k)} + \omega \sum_{i=1}^{p} \delta^{(k)}_i$$
  set $k = k+1$
end

This gives rise to a general purpose iterative solver, which presents a natural degree of parallelism. If we take $p = n$, then Algorithm 2.1 becomes the algorithm of Cimmino (Sloboda 1988). In the following, we will denote by $H$ the matrix $(I - Q)$, where $Q$ is the iteration matrix of the block Cimmino algorithm.

In this paper we do not study the characteristics and implementation of these methods, but rather we examine the acceleration process. This is crucial because of the slow convergence generally achieved by Algorithm 2.1 alone. The Conjugate Gradient acceleration of the Block Cimmino method is not influenced by the value of the relaxation parameter $\omega$ so we will assume thenceforth that $\omega$ is equal to 1. Details about the convergence and the parallelism of the method can be found in Arioli, Duff, Noailles, and Ruiz (1990).

We use partitionings (2.1) where $A_i$ is structurally orthogonal to $A_i^{(2)} (i = 1, 2, \ldots, p - 2)$, giving the equivalent of a partitioning in two blocks. For such a partitioning, Elfving (1980) shows that the eigenvalues of the iteration matrix of the Block Cimmino method are the cosines of the principal angles between the two manifolds defined by these two blocks. In this case, the spectrum of the iteration matrix lies strictly between $-1$ and $1$, is symmetric around $0$, and has many eigenvalues clustered around $0$. For iteration matrices which are not too ill conditioned, the conjugate gradient algorithm is a good method for accelerating the convergence of the iterative scheme, and generally achieves fast convergence.

For ill conditioned problems with poor convergence, we use preconditioning techniques giving rise to non-orthogonal projections. This enables us to perform, implicitly and in parallel, a right-hand side preconditioning of the matrix $A$. The spectrum of the iteration matrix we obtain still has the same characteristics as those described above, but also has clusters at the ends of the spectrum. This can cause poor convergence of the classical Conjugate Gradient method.

## 3 A problem with convergence

In this section, we want to introduce a particular class of problems on which the classical conjugate gradient acceleration performs poorly. We will discuss the characteristics of a typical problem of this type.

This example, of relatively small size ($n = 532$), comes from oil reservoir modelling simulation, and is recorded as problem PORES 3 in the Harwell-Boeing set of sparse matrices. The matrix is unsymmetric but with a symmetric pattern. To obtain reasonably fast convergence, the rows of the original matrix are reordered to maximize the minimum entry on the diagonal, and a symmetric permutation is then computed to extract a block tridiagonal structure with small bandwidth. The resulting block tridiagonal matrix is then partitioned in a two-block partitioning and is
preconditioned using a right-hand side preconditioner designed to make the two blocks more orthogonal. We are not here concerned with the details of the preconditioning but note that even in the best situation we obtain strange behaviour in the convergence of the conjugate gradient acceleration, as shown in Figure 3.1.

![Convergence with classical conjugate gradient](image)

Figure 3.1. Behaviour of the classical conjugate gradient acceleration on PORES 3 test problem.

The normwise backward error which is used to monitor the convergence is the scaled residual $\omega_k$ defined by

$$\omega_k = \frac{\| A x^{(k)} - b \|_\infty}{\| A \|_\infty \| x^{(k)} \|_1 + \| b \|_\infty}.$$  

A small value for $\omega_k$ means that the algorithm is normwise backward stable (Oettli and Prager 1964) in the sense that the solution $x^{(k)}$ is the exact solution of a perturbed problem where the max norm of the error matrix is less than or equal to $\omega_k$.

In Figure 3.1, we see some plateaux which are separated by parts of superlinear convergence. The level of accuracy which is obtained after 220 iterations is good, since $\omega_k$ less than $10^{-13}$ corresponds to a componentwise error in the solution of less than $10^{-5}$. Note, however, that after 157 iterations only one correct digit componentwise in the solution is obtained.

Figure 3.2 shows the spectrum of the matrix $H$ (see Section 2) on this problem. We first notice the high classical condition number of order $4 \times 10^8$, as the smallest eigenvalue is $0.53 \times 10^{-8}$ and the biggest is 2 less this value. Additionally, we observe a very strong clustering of the eigenvalues around one, with only 64 eigenvalues outside the interval $1 - 0.5 \times 10^{-2}, 1 + 0.5 \times 10^{-2}$. In spite of this, we see that the conjugate gradient algorithm requires many more than 64 iterations before reaching superlinear convergence. We also notice a cluster of 13 eigenvalues around the extremes, and examining the distribution more deeply reveals that, for an interval length of $10^{-4}$, there are in fact only 4 eigenvalues clustered around each extreme. The other eigenvalues are isolated and spread between these three clusters. We believe that the clusters at the extremes of the spectrum, which corresponds to the "ill conditioned part" of the iteration matrix, is the reason for this poor convergence of the conjugate gradient algorithm.

In the following section, we will experiment on this particular test problem, and try to find an alternative to the classical conjugate gradient algorithm which does not have the same trouble.
4 Acceleration with block conjugate gradient algorithms

Block Lanczos and block conjugate gradient algorithms, as opposed to the classical Lanczos and conjugate gradient algorithms, can identify multiple eigenvalues, and therefore offer the possibility of handling clusters of eigenvalues better. O’Leary (1980) gives a nice survey and studies the convergence of these two algorithms and of related techniques.

The block conjugate gradient acceleration procedure can be applied to the preconditioned case in the following manner.

Let $s$ be the block size in the block conjugate gradient algorithm, and let $K$ be the $n \times s$ rectangular matrix corresponding to the block right hand-side. One vector column of $K$ must correspond to the transformation of the right-hand side of the original linear system by the Block Cimmino iteration

$$k = \sum_{i=1}^{s} A_{G}^{-1} A^{T} (A' G^{-1} A'^{T})^{-1} b^{i},$$

and the $s-1$ others can be generated randomly, such that the matrix $K$ has full rank $s$.

Algorithm 4.1 (Block conjugate gradient acceleration algorithm)

$X^{(0)}$ is arbitrary. $R^{(0)} = K - HX^{(0)}$,

$P^{(0)} = R^{(0)}$ and $P^{(0)} = P^{(0)} \beta_{n}^{-1}$

for $j = 0, 1, 2, ..., $ until convergence do :

$$\lambda_{j} = (P^{(0)} R^{(0)})^{-1} \beta_{j}^{T} (R^{(0)} G R^{(0)})$$

$$X^{(j+1)} = X^{(j)} + \frac{\lambda_{j}}{\beta_{j}}$$

$$R^{(j+1)} = R^{(0)} - H P^{(j)} \lambda_{j}$$

$$\alpha_{j} = \beta_{j} (R^{(0)} G R^{(0)})^{-1} (R^{(j+1)} G R^{(j+1)})$$

$$P^{(j+1)} = R^{(j+1)} + \frac{\alpha_{j}}{\beta_{j+1}}$$

and $P^{(j+1)} = P^{(j+1)} \beta_{j+1}^{-1}$

where the matrices $\beta_{j}, j = 0, 1, 2, ..., $ are square matrices of full rank $s$. 
Algorithm 4.1 is the exact generalisation of the conjugate gradient acceleration described in Hageman and Young (1981), where the use of the symmetric positive definite preconditioning matrix $G$ (Arioli, Duff, Noailles, and Ruiz 1990) corresponds to the generalization of the block Cimmino method to any ellipsoidal norm (using ellipsoidal projections instead of orthogonal ones). We stress that the use of ellipsoidal norms $G$ requires only matrix-matrix products with the matrix $G$.

The algorithm is well defined so long as the matrices $R^{(j)}$, and $P^{(j)}$ retain full rank. As the $R^{(j)}$ matrices, which correspond to the residuals $K - H X^{(j)}$, go to zero at convergence, failure or ill conditioning might occur, especially if one residual vector converges faster than the others. However, O’Leary (1980) suggests monitoring the rank deficiency by using the $\beta_j$ transformation matrices, which can be for instance the matrices orthonormalizing the columns of the $P^{(j)}$, and to reduce the block size $s$ when linear dependence is detected.

The main property of these block techniques (O’Leary 1980), is that their convergence is governed by the reduced condition number $\kappa = \lambda_i / \lambda_s$, where $\lambda_i$, $i = 1, \ldots, n$ are the eigenvalues of the matrix $H$ in increasing order and $s$ is the block size, and not by the classical condition number $\kappa = \lambda_s / \lambda_1$ as for the conjugate gradient or Lanczos algorithm. This can improve the convergence considerably especially when the eigenvalues are clustered in a similar fashion to that observed for problem PORES 3. In that case for instance, the reduced condition number can vary from $4 \times 10^8$, for a block size of 1, to a value of order $10^2$ with a block size equal to 14.

![Figure 4.1. Convergence of block conjugate gradient algorithm on problem PORES 3, with varying block sizes.](image)

We have tried also the block Lanczos acceleration technique (O’Leary 1980). Provided the starting point $X^{(0)}$ is the same, both the block conjugate gradient and the block Lanczos algorithms generate, in exact arithmetic, the same sequence of iterates $X^{(j)}$. The only difference is how the Krylov space is spanned. In the block conjugate gradient algorithm we have $K r_j(R^{(0)}) = \text{Span}\{R^{(0)}, \ldots, R^{(j-1)}\}$ where $R^{(j)} G R^{(j)} = 0$ if $j \neq k$. The block Lanczos algorithm builds instead some $V$ matrices such that, $V^{(j)} G V^{(j)} = 0$ if $j \neq k$ as for the $R^{(j)}$ matrices in the block CG algorithm, but in addition $V^{(j)} G V^{(j)} = I$ for all $j$ whereas $R^{(j)} G R^{(j)} \neq I$. Because of that, the spanning of the Krylov space will be much better conditioned in the block Lanczos than in the block conjugate gradient algorithm. In fact, for block size $s = 16$ for instance, the matrices in
the block CG algorithm became singular after iteration 7, when the normwise backward error is $10^{-9}$, whereas the block Lanczos algorithm manages to reach an accuracy of less than $10^{-11}$ at iteration 9. On the other hand, the update of the iterates $X^{(0)}$ is more costly in the block Lanczos algorithm than in the block conjugate gradient algorithm (O’Leary 1980).

Our idea is then to couple the advantages of the two algorithms, to use the two-term recurrence formulae of the block conjugate gradient algorithm, which builds a basis of the Krylov space $K_{x_j}(R^{(0)})$ and is easy to implement, and to reorthonormalize the residual matrices $R^{(0)}$ into some $\bar{R}^{(0)}$ in order to avoid instability. This would result in the following algorithm:

Algorithm 4.2 (Modified block conjugate gradient acceleration algorithm)

- $X^{(0)}$ is arbitrary, $R^{(0)} = K - H X^{(0)}$,
- $\bar{R}^{(0)} = R^{(0)} \gamma_0^{-1}$ with $\gamma_0$ such that $(\bar{R}^{(0)\top} G \bar{R}^{(0)}) = I$
- $\bar{P}^{(0)} = R^{(0)} \beta_0^{-1}$ with $\beta_0$ such that $(\bar{P}^{(0)\top} G H \bar{P}^{(0)}) = I$

for $j = 0, 1, 2, \ldots$ until convergence do:

\[ \lambda_j = \beta_j^{-1} \]

\[ \bar{R}^{(j+1)} = (\bar{R}^{(0)} - H \bar{P}^{(0)} \lambda_j) \gamma_{j+1}^{-1} \] with $\gamma_{j+1}$ such that $(\bar{R}^{(j+1)\top} G \bar{R}^{(j+1)}) = I$

\[ X^{(j+1)} = X^{(0)} + \bar{P}^{(0)} \bar{P}^{(0)\top} G R^{(0)} \]

\[ = X^{(0)} + \bar{P}^{(0)} \bar{P}^{(0)\top} G R^{(0)} \left( \prod_{i=j}^{0} \gamma_i \right) \] \hspace{1cm} (4.1)

\[ \alpha_j = \beta_j \gamma_{j+1}^{-1} \]

\[ \bar{P}^{(j+1)} = (\bar{R}^{(j+1)} + \bar{P}^{(0)} \alpha_j) \beta_{j+1}^{-1} \] with $\beta_{j+1}$ such that $(\bar{P}^{(j+1)\top} G H \bar{P}^{(j+1)}) = I$

Consequently, as for the $V$ matrices in block Lanczos, the matrices $\bar{R}$ do not now correspond to the residuals. This would induce an overhead in the update of $X^{(j+1)}$, as the computation of the residuals $R^{(0)}$ requires an additional call to the matrix by matrix product $H X^{(0)}$. However, as indicated in equations (4.1), the residuals $R^{(0)}$ are linked to the $\bar{R}^{(0)}$ matrices, and the computation of $X^{(0)}$ involves only little overhead compared to Algorithm 4.1.

Figure 4.2. Convergence of the modified block conjugate gradient algorithm on problem PORES 3, with varying block sizes.

Copyrighted material
As can be seen in Figure 4.2, Algorithm 4.2 performs much better than the unmodified algorithm. The improvements due to the block acceleration techniques can easily be seen in the convergence of the block CG algorithm with block size 8 for instance. In this case, a normwise backward error of less than $10^{-17}$ is reached within 12 iterations, which corresponds to $12 \times 8 = 96$ iterations of the classical CG algorithm, and is much less than the 188 iterations required by this one for reaching the same level of accuracy (Figure 3.1).

References


