Partitioning Strategies for the Block Cimmino algorithm

List of authors:

M. Zenadi [*]

T. Drummond [†] I. S. Duff [‡] R. Guivarch [*] D. Ruiz [*]

We consider the Block Cimmino projection method for the solution of the linear system of equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a nonsingular large sparse unsymmetric matrix of order n. The blocks are obtained by partitioning the system into p strips of rows, with $p \leq n$, as in:

$$\begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_p \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_p \end{pmatrix}. \tag{1}$$

The Block Cimmino method projects the current iterate simultaneously onto the manifolds corresponding to the strips, and takes a convex combination of all the resulting vectors. Our aim is to focus on preprocessing and partitioning strategies of the original system to bring it to the form (1).

If the matrix $\mathbf{A}$ is ill conditioned then there are some linear combinations of rows that are almost equal to zero and, as mentioned by [2], these linear combinations may occur inside the blocks or across the blocks after row partitionings of the form (1). Assuming that the projections in the Block Cimmino algorithm are computed exactly on the subspaces, then the rate of convergence of the Block Cimmino algorithm depends only on the conditioning across the blocks. If we consider additionally Conjugate Gradient acceleration of the Block Cimmino method, as in [1, 2], the convergence behaviour of the resulting method is directly linked to the spectrum of the $n \times n$ matrix $\mathbf{E}_{RJ}$ formed as the sum of the previously mentioned projections, and given by

$$\mathbf{E}_{RJ} = \sum_{i=1}^{p} \mathbf{A}_i{}^T \left(\mathbf{A}_i \mathbf{A}_i{}^T\right)^{-1} \mathbf{A}_i \ , \tag{2}$$

assuming, for simplicity, that the block-rows $\mathbf{A}_i$ have full row rank. An efficient implementation of Block Cimmino requires a combination of a robust method for computing the projections and a partitioning strategy that minimizes the ill-conditioning across the blocks.

If the block rows $\mathbf{A}_i$ are nearly mutually orthogonal, i.e. $\mathbf{AA}^T$ is strongly block-diagonally dominant, we can expect that the method will converge very quickly, if the projections are computed accurately. Conversely, the structure of $\mathbf{AA}^T$ tells us about the orthogonality of the subspaces represented by block partitions of $\mathbf{A}$. In all our experiments we first normalize the matrix $\mathbf{AA}^T$ using `MC77` [5] so that the 2-norms of the rows and columns are close to one. Then the entries in $\mathbf{AA}^T$ correspond to the cosine of the principal angle between every pair of rows, and we may expect that if such a cosine is relatively small the corresponding pair of rows are

[*]Université de Toulouse, INPT(ENSEEIHT)-IRIT, France
[†]Lawrence Berkeley National Laboratory
[‡]CERFACS and Rutherford Appleton Laboratory

almost orthogonal. That is, if the block (i,j)th entry of the scaled $\mathbf{AA}^T$ is close to zero then the subspaces corresponding to the blocks $\mathbf{A}_i$ and $\mathbf{A}_j$ can be considered to be orthogonal. If $\mathbf{AA}^T$ is block tridiagonal, the blocks of $\mathbf{A}$ are such that the even numbered blocks are orthogonal as are also the odd-numbered blocks. Thus if we solve the projected subproblems accurately (using say a direct method) then we also solve the subproblems corresponding to the odd and even numbered blocks accurately. The partition of $\mathbf{A}$ is thus of the form $[\mathbf{A}] = \left[\begin{array}{cc} \mathbf{B}_1{}^T & \mathbf{B}_2{}^T \end{array}\right]^T$, where $\mathbf{B}_1 = \left\{ \bigcup_i \mathbf{A}_i \;/\; i \text{ odd} \right\}$ and $\mathbf{B}_2 = \left\{ \bigcup_i \mathbf{A}_i \;/\; i \text{ even} \right\}$. Such a partitioning is called a two-block partitioning. We denote by m_1 and m_2 the number of rows in $\mathbf{B}_1$ and $\mathbf{B}_2$ respectively. It was shown by [4] that, with such a two-block partitioning, the spectrum of matrix $\mathbf{E}_{RJ}$ is

$$\begin{array}{llll} \lambda_k & = 1 + \cos\Psi_k & k & = 1, \ldots, m_2 \\ \lambda_k & = 1 - \cos\Psi_{k-m_2} & k & = m_2+1, \ldots, 2m_2 \\ \lambda_k & = 1 & k & = 2m_2+1, \ldots, n \end{array}$$

where $\{\Psi_k\}_1^{m_2}$ are the principal angles between $\mathcal{R}(\mathbf{B}_1{}^T)$ and $\mathcal{R}(\mathbf{B}_2{}^T)$.

The main idea behind the so called two-block partitioning strategy is to exploit structural orthogonality between the subspaces $\mathcal{R}(\mathbf{A}_i{}^T)$ defined by the partitioning (1). This structural orthogonality can be analysed on the basis of the sparsity pattern of the normal equations matrix $\mathbf{AA}^T$. The idea is to find a permutation to transform the normal equations matrix $\mathbf{AA}^T$ into block tridiagonal form. That is we determine a permutation matrix $\mathbf{P}$ such that $\mathbf{B} = \mathbf{PAA}^T\mathbf{P}^T$ is in block tridiagonal form using an implementation of algorithm of [3]. We then use this permutation to solve the row-wise permuted system of equations

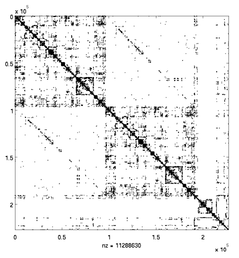
$$\mathbf{PAx} = \mathbf{Pb} \tag{3}$$

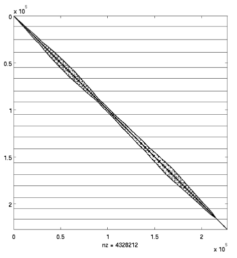
using the block Cimmino algorithm.

The main problem with this first preprocessing strategy for general sparse matrices, is that the block tridiagonal structure has diagonal blocks with very differing sizes leading to a partitioning with a bad degree of parallelism because of unbalanced tasks in the block-row projections.

In our second preprocessing strategy, we take into account the numerical values in the matrix $\mathbf{AA}^T$, and do not enforce strict orthogonality between every second block of rows. We drop small entries of $\mathbf{AA}^T$ and by using the relationship to principal angles can relax the orthogonality condition in a controlled way in an attempt to obtain much better parallelism compared to the previous strategy. We thus only keep nonzero entries above a given tolerance and permute the resulting filtered matrix into block tridiagonal form using the Cuthill-McKee algorithm and use the row partitioning defined by this block tridiagonal structure to solve (3).

Of course, since numerical values have been dropped from the normal equations matrix, the resulting partition will not provide two subsets of structurally orthogonal blocks of rows, but, if the values dropped are sufficiently small, we may expect that the numerical properties of the resulting iteration matrix will be relatively close to that of the "*strict*" two-block partitioning case. Additionally, since the filtered matrix has less entries than the original normal equations matrix, the resulting block tridiagonal permuted matrix will normally have a smaller bandwidth than **B** and this may help to define a partitioning on **A** with more blocks, better balanced projections, and a higher degree of parallelism.





To illustrate this preprocessing strategy, we show preliminary results obtained with the *bmw3_2* matrix (of size $N = 227362$) from the Tim Davis collection. The original matrix, before preprocessing, has the symmetric pattern shown in the left subplot above, in which there is no evident structural orthogonality between sets of rows.

We perform some runs on a 24 (Intel i7) cores parallel machine, with a generic partitioning into 24 equal sized blocks. The Block Cimmino method needs then 915 iterations to reach a scaled residual value of the order 10^{-7}. It requires 5.46 sec for the analysis, 2.41 sec for the factorization of the augmented systems and 63.25 sec for the iterations.

Following the second preprocessing strategy, we filter $\mathbf{AA}^T$ matrix under the value 0.1, which corresponds to the cosine of an angle of 84°, and we obtain the pattern in the right subplot after Cuthill-McKee and a reduction of nonzero entries in $\mathbf{AA}^T$ of 88%. With this block tridiagonal structure, we can define 18 almost balanced partitions, the biggest one of size 16129 rows and the smallest of 11062 rows. The Block Cimmino method needs then 296 iterations to converge to the same level, with 7.19 sec for the analysis, 3.29 sec for the factorization, 24.62 sec for the iterations, and with 2.48 sec for the extra work in the preprocessing.

We intend to develop and compare the benefits of such strategies on parallel architectures, different test examples and also with variants where we may drop entries in **A** in combination with entries in $\mathbf{AA}^T$

References

[1] M. Arioli, I. S. Duff, J. Noailles, and D. Ruiz, (1992), *A Block Projection Method For Sparse Matrices*, SIAM J. Sci. Stat. Comput., 47–70.

[2] R. Bramley and A. Sameh, (1990), *Row projection methods for large nonsymmetric linear systems*, Tech. Rep. 957, Center for Supercomputing Research and Development, University of Illinois, Urbana, IL. Also, SISSC, Vol. **13**, January 1992.

[3] E. Cuthill and J. McKee, (1969), *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings 24th National Conference of the Association for Computing Machinery, New Jersey, Brandon Press, 157–172.

[4] T. Elfving, (1980), *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math., **35**, 1–12.

[5] D. Ruiz, (2001), *A scaling algorithm to equilibrate both row and column norms in matrices*, Tech. Rep. RAL-TR-2001-034, Rutherford Appleton Laboratory.