XIX International Conference on Water Resources
CMWR 2012
University of Illinois at Urbana-Champaign
June 17-22,2012

BLOCK FSAI PERFORMANCE WITH GRAPH PARTITIONING IN LARGE SIZE SUBSURFACE PROBLEMS

Massimiliano Ferronato*, Nicola Castelletto* and Carlo Janna*

*University of Padova
Department of Civil, Environmental and Architectural Engineering
via F. Marzolo 9, 35131 Padova, Italy
e-mail: massimiliano.ferronato@unipd.it, web page: http://www.dmsa.unipd.it/~ferronat

Key words: Graph partitioning, Preconditioning, Iterative methods, Parallel computations

Summary. Adaptive Block FSAI (ABF) is a novel and promising preconditioner for the efficient parallel solution of linear systems and eigenproblems arising from subsurface applications. However, one of its main drawbacks stems from the reduced scalability, as the iteration count to converge tends to grow increasing the number of processors. Graph partitioning techniques can help improve both the preconditioner performance and scalability. Different algorithms are experimented with in a test problem arising from a groundwater flow application. The results show that coupling graph partitioning with ABF appears to be an important factor to increase significantly the preconditioner efficiency, allowing for its effective use also on massively parallel simulations.

1 INTRODUCTION

The solution to large size linear systems is a central task in most numerical subsurface simulations. The continuous improvement of parallel computers can afford the opportunity to implement numerical models with up to some million unknowns, provided that an efficient solver is available. While iterative methods can be easily adapted to parallel computations, most traditional preconditioners based on incomplete LU factorizations (ILU) cannot. This is why a great interest in the research and development of novel parallel preconditioners can be recently found, e.g. references[1−4].

A novel parallel preconditioner of symmetric positive definite (SPD) problems is the Adaptive Block Factorized Sparse Approximate Inverse (ABF) algorithm[5,6]. ABF is a factored operator which aims at minimizing the distance between the preconditioned matrix and an arbitrary block diagonal matrix in the sense of the minimal Frobenius norm. Then, another block diagonal preconditioner can be applied, an "outer" preconditioner so to say, e.g. a block diagonal Incomplete Cholesky (IC) decomposition or even a direct solver for each block. In combination with IC, ABF has proved to be quite a robust and efficient tool for the parallel solution of a great variety of engineering applications[6]. One of the key features contributing to the successful perfomance of ABF is the adaptive algorithm which allows for the dynamic construction of a nearly optimal sparsity pattern.

However, a drawback penalizing ABF-IC is the increasing iteration count to converge as the number of processors grows[6]. In its native formulation, ABF gives rise to equally sized diagonal blocks operating on a uniform stripe partition of the system matrix. A possible improvement could be obtained by removing this assumption, thus letting the matrix partition to be non uniform. Although this can generate an uneven computational load among the processors, a smart partitioning could help get a better performance. The issue of partitioning a matrix is not new. Investigations on ordering techniques began in the late 60's with the aim at reducing the fill-in produced by Gaussian elimination. Afterwards, they were also used in the context of iterative methods to improve the performance of ILU preconditioners. With the coming of parallel computers, the problem of properly mapping the system unknowns to different processors arosed, so as to ensure a good load balance between the assigned tasks and a reduced communication overhead. One of the first and most common ordering methods is the Cuthill-McKee algorithm (CMK)[7] with its reverse version (RCM)[8]. Recently, more complex multilevel partitioning schemes have been developed. Available software packages collecting a variety of partitioning algorithms are for example SCOTCH[9] and METIS[10].

The aim of this communication is to investigate the effect of coupling graph partitioning techniques with ABF-IC. The paper is organized as follows. The ABF preconditioner is first reviewed, showing how the adaptive generation of the sparsity pattern is related to the ordering of the unknowns in the system matrix. Then, partitioning strategies from the SCOTCH and METIS packages are introduced and investigated in detail when used with ABF-IC in a representative groundwater flow problem. Finally, a few concluding remarks close the communication.

2 THE ABF PRECONDITIONER

Let A be an SPD matrix of size n with $\mathcal{S}_L$ and $\mathcal{S}_{BD}$ a sparse lower triangular and a dense block diagonal non-zero $n \times n$ pattern, respectively. Denote as n_b the number of diagonal blocks and m_{i_b} the size of the i_b-th block, and let D be an arbitrary full-rank matrix with non-zero pattern $\mathcal{S}_{BD}$. The Block FSAI preconditioner of A is defined as the product $F^T F$, where F is a lower block triangular factor with non-zero pattern $\mathcal{S}_{BL} = \mathcal{S}_{BD} \cup \mathcal{S}_L$ minimizing the Frobenius norm $\|D - FL\|_F$, with L the exact lower Cholesky factor of A. Differentiating with respect to the unknown entries $[F]_{ij}$, $(i, j) \in \mathcal{S}_{BL}$, and setting to 0 yield a linear relationship for the i-th row $\mathbf{f}_i$ of F [5]:

$$A[\mathcal{J}_i, \mathcal{J}_i]\, \mathbf{f}_i[\mathcal{J}_i] = \mathbf{v} \qquad \mathcal{J}_i = \{l : (i, l) \in \mathcal{S}_{BL}\} \tag{1}$$

where $A[\mathcal{J}_i, \mathcal{J}_i]$ is the SPD submatrix of A containing the entries $[A]_{kj}$ such that $k, j \in \mathcal{J}_i$, $\mathbf{f}_i[\mathcal{J}_i]$ is the subvector of $\mathbf{f}_i$ containing the components $[\mathbf{f}_i]_j$ such that $j \in \mathcal{J}_i$, and $\mathbf{v}$ is the null vector, except the last m_{i_b} components which are arbitrary, being i_b the block index the row i belongs to. The system (1) has a unique solution only after m_{i_b} components of $\mathbf{f}_i[\mathcal{J}_i]$ are set. The most practical choice is to prescribe all the components of $\mathbf{f}_i[\mathcal{J}_i]$ falling within the i_b-th diagonal block to 0 with the exception of the entry corresponding to $[F]_{ii}$ which is set to 1. This implies F to be actually a unit lower triangular matrix with structure:

$$F = \begin{bmatrix} I & 0 & \cdots & 0 \\ F_{21} & I & & \vdots \\ \vdots & & \ddots & \vdots \\ F_{n_b 1} & F_{n_b 2} & \cdots & I \end{bmatrix} \tag{2}$$

The factor F can be efficiently built in parallel as all systems (1) can be solved independently. Recalling the definition above, the preconditioned matrix FAF^T should resemble DD^T, i.e. a block diagonal matrix whatever D. In other words, FAF^T has the block structure:

$$FAF^T = \begin{bmatrix} B_1 & R_{1,2} & \cdots & R_{1,n_b} \\ R_{1,2}^T & B_2 & \cdots & R_{2,n_b} \\ \vdots & & \ddots & \vdots \\ R_{1,n_b}^T & R_{2,n_b}^T & \cdots & B_{n_b} \end{bmatrix} \tag{3}$$

where $R_{i,j}$ and B_{i_b} tend to zero and to the diagonal blocks of DD^T, respectively, as the fill-in of F increases. However, as D is arbitrary, it is still not ensured that FAF^T is better than A in an iterative solution method, so it is necessary to precondition FAF^T again. Assuming that the off-diagonal blocks $R_{i,j}$ are close to 0, an effective choice could be using as "outer" preconditioner the incomplete, or whenever feasible even the exact, factorization of each B_{i_b} block. For example, good results have been obtained by using an incomplete Cholesky factorization of each diagonal block[5,6]. Whatever the outer preconditioner $J^{-T}J^{-1}$ is, the resulting preconditioned matrix reads:

$$J^{-1}FAF^TJ^{-T} = WAW^T \tag{4}$$

with the final preconditioner $M^{-1} = W^TW = F^TJ^{-T}J^{-1}F$.

One of the key factor, however, affecting the performance of M^{-1} in the solution of a linear system or an eigenproblem is the sparsity pattern $\mathcal{S}_{BL}$ selected for the computation of F, whose optimal a priori choice is not thouroughly clear. An attractive option can be selecting $\mathcal{S}_{BL}$ dynamically during the computation of F. An adaptive algorithm can be implemented starting from the theoretical optimal properties of Block FSAI. Janna and Ferronato[6] have demonstrated that under the hypothesis that the $J^{-T}J^{-1}$ contains the exact inverse of each diagonal block B_{i_b} the Kaporin conditioning number[11] κ of the preconditioned matrix WAW^T satisfies the following bound:

$$1 \leq \kappa\left(WAW^T\right) \leq C\left(\prod_{i=1}^{n}[FAF^T]_{ii}\right)^{1/n} \tag{5}$$

where C is a constant scalar independent of W. Block FSAI has the theoretical property of minimizing the upper bound of (5) for any given pattern $\mathcal{S}_{BL}$. The basic idea of the adaptive algorithm is to select the off-block diagonal non-zero entries in any row $\mathbf{f}_i$ of F so as to reduce as much as possible such an upper bound. This is feasible because each factor $[FAF^T]_{ii}$ turns out to be a quadratic form depending on $\mathbf{f}_i$ only. Denoting by $\tilde{\mathbf{f}}_i$ the subvector of $\mathbf{f}_i$ including the off-block diagonal entries only, $\tilde{A}_{i_b}$ the square submatrix of A built from the first to the m-th row/column, with m the sum of size of the first $(i_b - 1)$ diagonal blocks of $\mathcal{S}_{BD}$, and $\tilde{\mathbf{a}}_i$ the subrow of A with the first m elements of the i-th row, each factor $[FAF^T]_{ii}$ reads:

$$[FAF^T]_{ii} = \tilde{\mathbf{f}}_i^T \tilde{A}_{i_b} \tilde{\mathbf{f}}_i + 2\tilde{\mathbf{f}}_i^T \tilde{\mathbf{a}}_i + [A]_{ii} \tag{6}$$

Minimizing every $[FAF^T]_{ii}$, $i = 1, \ldots, n$, is equivalent to minimize the upper bound in (5). The adaptive pattern search for F can be therefore implemented as follows. Start from an initial guess $\mathcal{S}_F^0$ for the sparsity pattern of the off-block diagonal part of F, e.g. the empty pattern, and compute the gradient $\mathbf{g}_i$ of each quadratic form $[FAF^T]_{ii}$:

$$\mathbf{g}_i = 2\left(\tilde{A}_{i_b}\tilde{\mathbf{f}}_i + \tilde{\mathbf{a}}_i\right) \tag{7}$$

Then, add to $\mathcal{S}_F^0$ the position j into each row i corresponding to the largest component of $\mathbf{g}_i$ computed in (7), so as to obtain an augmented pattern $\mathcal{S}_F^1$. After computing the new factor F over $\mathcal{S}_F^1$, the procedure above can be iterated in order to build $\mathcal{S}_F^2$, $\mathcal{S}_F^3$, and so on. The search into each row can be stopped when either a maximum number $k_{\max}$ of entries are added to the initial pattern or the relative variation of $[FAF^T]_{ii}$ in two consecutive steps is smaller than a prescribed tolerance ϵ_F. The Block FSAI computed with the adaptive algorithm described above is denoted as the ABF preconditioner which has proved very effective even if the outer preconditioner is not the exact inverse of the diagonal blocks B_{i_b}. For more details, see reference[6].

If the adaptive search starts from an empty $\mathcal{S}_F^0$ pattern, at the first step $\mathbf{g}_i$ is equal to $\tilde{\mathbf{a}}_i$ (see equation (7)), so that the initial positions selected into $\mathcal{S}_F^0$ are nothing but those corresponding to the largest off-block diagonal entry of A into each row. However, if $\tilde{\mathbf{a}}_i$ is the null vector, i.e. there is no connection between the i-th row and the previous blocks, $\mathbf{g}_i$ is also null, and no entry is added in the i-th row of F. In other words, if the i-th row of A has non-zero terms in its diagonal block only, the i-th row of F is empty except $[F]_{ii} = 1$. This means that ordering the unknowns so that the non-zero off-block diagonal terms of A are concentrated in a few rows only can be an efficient way to reduce and optimally distribute the fill-in of F, hence its computation and application cost, i.e. ultimately to improve the overall ABF performance.

3 NUMERICAL RESULTS

The performance of ABF coupled to graph partitioning techniques taken from both SCOTCH and METIS libraries is investigated in the solution of SPD linear systems arising from a challenging real groundwater flow problem. The Preconditioned Conjugate

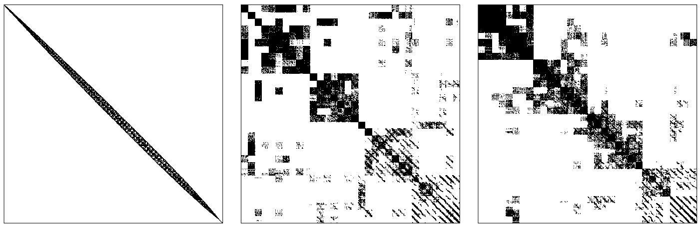

Figure 1: Sparsity pattern generated by RCM (left), SCOTCH (middle) and METIS (right)

Test case	# iter.	T_p	T_s	T_t	μ_F	μ_J	μ_W
C1	173	3.79	1.63	5.42	0.992	1.250	2.242
C2	164	0.69	1.12	1.80	0.154	1.215	1.368
C3	160	0.68	1.08	1.77	0.150	1.214	1.364

Table 1: ABF-IC performance with $n_p = n_b = 32$. The selected user-specified parameters are: $k_{\max} = 25$, $\epsilon_F = 10^{-8}$, $\rho_B = 0$, and $\rho_J = 10$

Gradient (PCG) solver is used, with the convergence achieved when the relative residual is smaller than 10^{-10}. A block diagonal IC decomposition is used as outer preconditioner, giving rise to the ABF-IC approach. The density of a matrix K is defined as the ratio between the number of non-zeroes of K and A. The computational perfomance is evaluated in terms of the number of iterations, the wall clock time in seconds T_p and T_s for the preconditioner computation and the PCG to converge, respectively, with the total time $T_t = T_p + T_s$. All the simulations have been carried out on the IBM SP6/5376 cluster at the CINECA Centre for High Performance Computing, equipped with IBM Power6 processors at 4.7 GHz with 168 nodes, 5376 computing cores, and 21 Tbyte of internal network RAM.

Three test cases are considered, namely C1, C2, and C3. In the test case C1 a simple RCM preliminary ordering is applied on A, as already done in reference[6]. This is to be considered as the "reference" case with respect to which the advances provided by graph partitioning are to be measured. In the other test cases the effects of the partitioning produced by SCOTCH (C2) and METIS (C3) are considered. The matrix selected as test problem is denoted as STOCF-1465 and arises from the groundwater flow simulation of the multi-aquifer system underlying the Venice Lagoon using a stochastic distribution of the hydraulic conductivity tensor[12]. STOCF-1465 is publicly available at the University of Florida Sparse Matrix Collection and totals 1,465,137 unknowns and 21,005,389 non-zeroes.

At first, the analysis is performed fixing the number of processors to $n_p = 32$. As already emphasized in references[5,6], the most effective option is to set the number of blocks n_b, i.e. the subdomains each matrix is to be partitioned into, equal to n_p. The other user-specified parameters required for the ABF setting are:

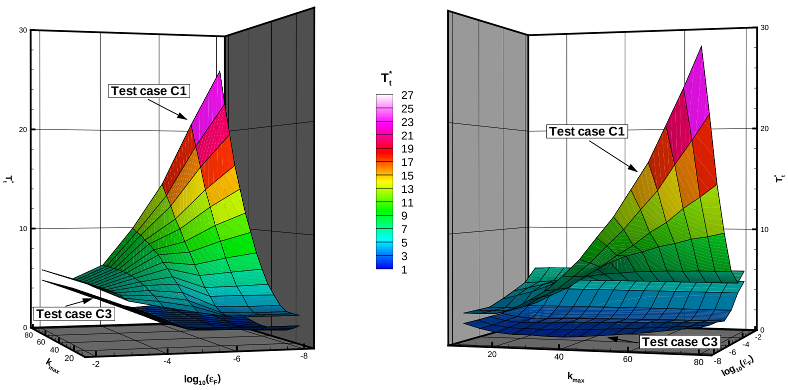

Figure 2: Normal time T_t^* plotted vs. $(k_{\max}, \epsilon_F)$ in test cases C1 and C3

- the maximum number of steps $k_{\max}$ and the exit tolerance ϵ_F for the adaptive pattern generation;
- the fill-in controlling parameters ρ_B and ρ_J for the computation and the IC decomposition of each B_{i_b} block, respectively.

The pairs $(k_{\max}, \epsilon_F)$ and (ρ_B, ρ_J) control the density of F and J, respectively, in the overall factor W.

The qualitative difference between the sparsity patterns generated by RCM, SCOTCH and METIS can be appreciated in Figure 1. The effect of SCOTCH and METIS partitionings is to concentrate a large number of entries into n_b diagonal blocks of A of almost the same size. Irrespective of the selected algorithm, using a partitioning technique allows for concentrating more than 90% of non-zeroes in the diagonal blocks, while with RCM this percentage is about 70% only. The impact graph partitioning has on the ABF-IC performance is provided in Table 1. The selected user-specified parameters are those providing the best performance. As expected, the effect on the density of F is quite important, with the total time required for the solution significantly reduced with respect to the reference C1 case using almost indifferently SCOTCH or METIS.

Now we want to investigate the effect of graph partitioning in the selection of the user-specified parameters controlling the density of F, i.e. the pair $(k_{\max}, \epsilon_F)$. For the sake of simplicity we consider the test case C3 only compared with C1. The number of steps $k_{\max}$ is varied between 10 and 80, while the exit tolerance ϵ_F spans 6 orders of magnitude from 10^{-8} to 10^{-2}. The outcome is shown in Figure 2 where the total time normalized with respect to the best T_t value obtained in the test case C3 is plotted as a function of $k_{\max}$ and ϵ_F. Clearly, graph partitioning not only improves the ABF-IC performance, but also increases significantly its robustness with respect to the selection of the user-specified parameters.

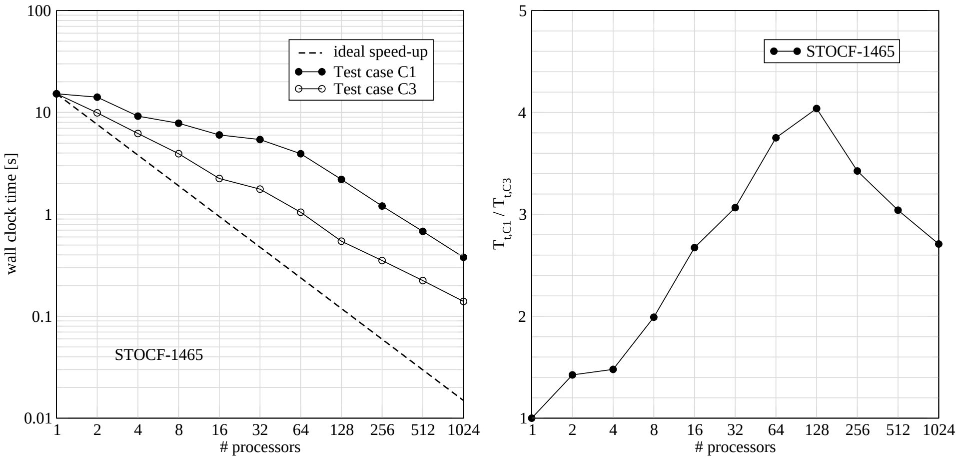

Figure 3: Wall clock time vs. the number of processors (left) and Ratio between the total wall clock time required by Test cases C3 and C1 (right)

As to the ABF-IC scalability, Figure 3 provides the overall wall clock time T_t elapsed varying $n_p = n_b$ from 1 to 1024 for the reference case C1 and the test case C3 along with the corresponding ratio. Coupling ABF-IC to a graph partitioning technique allows for a significant improvement of both the preconditioner scalability, which appears to be much closer to the ideal speed-up profile, and performance, which is up to 4 times faster. Note that 1024 is actually quite a large number of processors, especially on consideration of the matrix size, so that a worsening of the parallel degree is to be somewhat naturally expected.

4 CONCLUSIONS

Graph partitioning techniques are efficient algorithms designed to balance the computational load and limit the communication overhead among the processors in the parallel solution of linear systems or eigenproblems. In the present communications, different graph partitioning techniques have been experimented with in connection with the ABF-IC preconditioner for the solution of SPD linear systems. Graph partitioning algorithms are selected with a twofold objective: (i) reducing the preconditioner fill-in, i.e. the cost for its computation and application, and (ii) improving its scalability with the number of processors. The results obtained in a real groundwater flow problem show that coupling a graph partitioning algorithm with ABF-IC is an important factor for improving significantly both the preconditioner performance and scalability. The partitioning software packages considered in the present work, i.e. SCOTCH and METIS, prove roughly equivalent. Among them, METIS appears to be often preferable because of its great efficiency, though the partitioning cost is generally negligible with respect to that of the preconditioner and the solver.

REFERENCES

[1] J. Boyle, M. Mihajlovic and J. Scott. HSL_MI20: An efficient AMG preconditioner for finite element problems in 3D. *Int. J. Num. Meth. Engng.*, **82**, 64–98, (2010).

[2] M.F. Wheeler, T. Wildey and I. Yotov. A multiscale preconditioner for stochastic mortar mixed finite elements. *Comp. Meth. Appl. Mech. Engng.*, **200**, 1251–1262, (2011).

[3] M. Ferronato, C. Janna and G. Pini. Parallel solution to ill-conditioned FE geomechanical problems. *Int. J. Num. Anal. Meth. Geomech.*, **36**, 422–437, (2012).

[4] M. Ferronato, C. Janna and G. Pini. Shifted FSAI preconditioners for the efficient parallel solution of non-linear groundwater flow models. *Int. J. Num. Meth. Engng.*, doi: 10.1002/nme.3309, (2012). To appear.

[5] C. Janna, M. Ferronato and G. Gambolati. A Block FSAI-ILU parallel preconditioner for symmetric positive definite linear systems. *SIAM J. Sci. Comput.*, **32**, 2468–2484, (2010).

[6] C. Janna and M. Ferronato. Adaptive pattern research for Block FSAI preconditioning. *SIAM J. Sci. Comput.*, **33**, 3357–3380, (2011).

[7] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. of the 1969 24th National Conference*, 157–172, (1969).

[8] A. George. *Computer implementation of the finite element method.* Tech. Rep. STAN-CS-208, Department of Computer Science, Stanford University, Stanford, CA, (1971).

[9] F. Pellegrini. *SCOTCH, software package and libraries for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning.* Version 5.1.10, 2010. `http://http://www.labri.fr/perso/pelegrin/scotch`

[10] G. Kaypis and V. Kumar. *METIS - A software package for partitioning unstructured graphs, partitioning meshes and computing fill-reducing orderings of sparse matrices.* Version 5.0, 2011. `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`

[11] I.E. Kaporin. New convergence results and preconditioning strategies for the conjugate gradient method. *Num. Lin. Algebra Appl.*, **1**, 179–210, (1994).

[12] P. Teatini, M. Ferronato, G. Gambolati, D. Baù and M. Putti. Anthropogenic Venice uplift by seawater pumping into a heterogeneous aquifer system. *Water Resour. Res.*, **46**, W11547, doi: 10.1029/2010WR009161, (2010).