TOWARDS HIGH PERFORMANCE HYBRID DIRECT-ITERATIVE SOLVERS FOR LARGE SPARSE SYSTEMS *

PASCAL HÉNON , FRANÇOIS PELLEGRINI , PIERRE RAMET , AND JEAN ROMAN †

Key words. Parallel Incomplete LU factorization, Sparse Gaussian Elimination, Preconditioning, Direct and Iterative methods, Sparse linear systems.

Solving large sparse linear systems by iterative methods has often been quite unsatisfactory when dealing with pratical "industrial" problems (3D problems discretized with a finite element method can lead to systems in the order of millions or hundreds of millions of unknowns). The main difficulty encountered by such methods is their lack of robustness and, generally, the unpredictability and unconsistency of their performance over a wide sample of different problems; certain methods work quite well for certain types of problems but can fail completely on other problems.

Over the past few years, direct methods have made significant progress thanks to either the combinatorial analysis of the Gaussian elimination process and the parallel algorithmic of blockwise solvers optimized for modern parallel supercomputers. Its is now possible to solve practical three-dimensional problems in the order of several millions of equations in a very powerful way with the direct solvers that efficiently use the superscalar effects of modern processors and take all the benefits from the computer architectures based on networks of SMP nodes (IBM SP, DEC-Compaq, for example) [2, 8, 4, 6] However, direct methods may fail to solve very large three-dimensional problems, due to the large amount of memory needed for these cases and despite any memory optimization.

In our work, we consider an approach which, we hope, will bridge the gap between the two classes of methods. The goal is to provide a method which exploits the parallel blockwise algorithmic used in the framework of the high performance sparse direct solvers for developping robust parallel incomplete factorization based preconditioners for iterative solvers such as GMRES or Conjugate Gradient solvers. The idea is then to define an adaptive blockwise incomplete factorization [11] that is much more accurate (and numerically more robust) than the scalar incomplete factorizations commonly used to precondition iterative solvers. Such incomplete factorization can take advantage of the latest breakthroughts in sparse direct methods and particularly should be very competitive in CPU time (effective power used from processors and good scalability) while avoiding the memory limitation encountered by direct methods. By this way, we expect to be able to solve systems in the order of hundred millions of unknowns and even one billion of unknowns. Another goal is to analyse and justify the chosen parameters that can be used to define the block sparse pattern in our incomplete factorization.

The driving rationale for this study is that it is easier to incorporate incomplete factorization methods into direct solution software than it is to develop new incomplete factorizations. In that way, we use, as a starting point, the algorithms and the software components of PaStiX [6, 7] that is a parallel high performance supernodal direct solver for sparse symmetric positive definite systems and for sparse unsymmetric systems with a symmetric pattern. These different components are:

1. The *ordering* phase, which computes a symmetric permutation of the initial matrix A such that factorization process will exhibit as much concurrency as possible while incurring low fill-in. In this work, we use a tight coupling of the Nested Dissection and Approximate Minimum Degree algorithms [1, 10]. The partition of the original graph into supernodes is achieved by merging the partition of separators computed by the Nested Dissection algorithm and the supernodes amalgamated for each subgraph ordered by Halo Approximate Minimum Degree.
2. The *block symbolic factorization* phase, which determines the block data structure of the factorized matrix associated with the partition resulting from the ordering phase. This structure consists of several column-blocks, each of them containing a dense symmetric diagonal block and a set of

*This work was supported by the NSF/INRIA project *Robust parallel preconditioning methods: bridging the gap between direct and iterative solvers.*

†INRIA, Scalapplix project - LaBRI, 351 cours de la Libération 33405 Talence

dense rectangular off-diagonal blocks. This block symbolic factorization of the reordered matrix is performed in quasi-linear space and time (the linearity is proven in the case of complete Nested Dissection [3]), and block data structures are much more compact than column-compressed storage [10]. One can notice that in the direct factorization framework the symmetric diagonal blocks are considered as dense by default. That will not be the case in the incomplete factorization.

3. The *block repartitioning and scheduling* phase, which refines the previous partition by splitting large supernodes in order to exploit concurrency within dense block computations in addition to the parallelism provided by the block elimination tree, both induced by the block computations in the supernodal solver (we use a version with a local agregation of the out-going contributions for the blocks mapped on other processors). Once the initial supernodal partition is refined, the scheduling phase consists in mapping onto the processors the resulting blocks according to a 1D scheme (i.e. by column blocks) for the lower part of the tree, and according to a 2D scheme (i.e. by blocks) for the upper part of the tree. In this phase, we also compute a static optimized scheduling of the computational and communication tasks according to BLAS and communication time models calibrated on the target machine. This static scheduling will drive the parallel factorization and the backward and forward substitutions [5, 7].

Our approach consists in computing symbolically the block structure of the factors that would have been obtained with a complete factorization, and then deciding to drop off some blocks of this structure according to relevant criterions. This incomplete factorization induced by the new sparse pattern is then used in a preconditioned GMRES or Conjugate Gradient solver [11]. Our main goal at this point is to achieve a significant diminution of the memory needed to store the incomplete factors while keeping enough fill-in to make the use of BLAS3 primitives profitable. Naturally, we must keep using the ordering phase and the mapping and scheduling phase (this phase is modified to suit the incomplete factorization block computation) to ensure an efficient parallel implementation of the block preconditioner computation and the forward and backward substitutions in the iterations.

The first crucial point is then to find a good initial partition that can be used in the symbolic factorization (phase 2). It cannot be the initial supernodal partition computed by the ordering phase (phase 1) because it would be too costly to consider the diagonal blocks as dense blocks like in a complete factorization. Therefore, we use a refined partition of this supernodal partition that will then define the elementary blocks of the factors. We obtain the refined partition by splitting the column blocks issued from the supernodal partition according to a maximum blocksize parameter. This allows more possiblities to drop some blocks in the preconditioner. This blocksize parameter is an important key to find a good trade-off as described at the beginning of this paragraph. An important result from theorical analysis is that if we consider nested dissection ordering based on separations theorems [9] and if we introduce some asymptotically refined partitions, one can demonstrate that the total number of blocks computed by the block symbolic factorization and the time to compute these blocks are still linear (this result was true with the supernodal partition).

The second crucial point concerns the various criterions that are used to drop some blocks from the blockwise symbolic structure induced by the refined partition. The dropping criterions are, for example, the level-of-fill of a block in the block elimination process, the number of times a block contributes to a modification on another block or the number of times a block is modified by a contribution from another block. Remaining in the same theoretical context than in the previous paragraph, one can show that the number of operations needed to compute these criterions has a quasi-linear complexity. On the other hand, even if these steps are very cheap in time and memory compared to the numerical incomplete factorization, it could be done in parallel if necessary.

Our presentation will detail the different points presented in this abstract and will give an analysis of some first convergence results for practical large systems.

REFERENCES

[1] P. R. Amestoy, T. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. and Appl.*, 17:886–905, 1996.

[2] P. R. Amestoy, I. S. Duff, and J. Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods in Appl. Mech. Engrg.*, 184:501–520, 2000.

[3] P. Charrier and J. Roman. Algorithmique et calculs de complexité pour un solveur de type dissections emboîtées. *Numerische Mathematik*, 55:463–476, 1989.

[4] A. Gupta and V. Kumar. WSSMP: A high-performance serial and parallel symmetric sparse linear solver. In *PARA'98 Workshop on Applied Parallel Computing in Large Scale Scientific and Industrial Problems, Umea, Sweden*, June 1998.

[5] P. Hénon. *Distribution des Données et Régulation Statique des Calculs et des Communications pour la Résolution de Grands Systèmes Linéaires Creux par Méthode Directe.* PhD thesis, LaBRI, Universit Bordeaux I, Talence, France, November 2001.

[6] P. Hénon, P. Ramet, and J. Roman. PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing*, 28(2):301–321, January 2002.

[7] P. Hénon, P. Ramet, and J. Roman. Efficient algorithms for direct resolution of large sparse system on clusters of SMP nodes. In *SIAM Conference on Applied Linear Algebra, Williamsburg, Virginie, USA*, July 2003.

[8] X. S. Li and J. W. Demmel. A scalable sparse direct solver using static pivoting. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, Texas, March 22-24, 1999.

[9] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM Journal of Numerical Analysis*, 16(2):346–358, April 1979.

[10] F. Pellegrini, J. Roman, and P. Amestoy. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. *Concurrency: Practice and Experience*, 12:69–84, 2000. Preliminary version published in *Proceedings of Irregular'99*, LNCS 1586, 986–995.

[11] Y. Saad. *Iterative Methods for Sparse Linear Systems, Second Edition.* SIAM, 2003.