A block sparse direct multifrontal solver in SCAD software

Sergiy Yu. Fialko
Kiev National University of Construction and Architecture
Povitroflotski av., 31, 03-037 Kiev, Ukraine
e-mail: sfialko@poczta.onet.pl

Abstract

The sparse direct multifrontal solver with block factoring in frontal matrix is presented. The nodal reordering instead of widespread finite element one, possibility to work with arbitrary reordering method and block factoring procedure in dense frontal matrix are discussed. The numerous examples from computational practice of Software Company SCAD Soft (www.scadsoft.com) illustrate the high efficiency of proposed method.

Keywords: finite element method, multifrontal solver, block factoring, reordering

1. Introduction

The sparse direct solvers are widely applied in modern finite element (FE) software for analysis of many kinds of problems. When the size of finite element problem is not too big (it is about 100 000 – 1 000 000 equations for PC computers), the sparse direct solvers lead to fast solution. They possess a low sensitivity to ill-conditioning, allows one to recognize the geometrical instability of computational model and the analysis time practically does not depend on number of right-hand sides, if one is not too big. So, sparse direct solvers are a powerful tool for practical analysis of structural mechanic problems.

This paper is devoted to further development of sparse direct multifrontal solver [5], which has been applied in Robot Millennium (www.robobat.com) and SCAD (www.scadsoft.com) software. The presented here improvements cover the more efficient reordering method and block mode of $\mathbf{LSL}^{\mathrm{T}}$ factoring of dense frontal matrix instead of previously applied non-block **LU** decomposition.

2. Sparse direct multifrontal solver

The reordering methods lead to reduction of fill-in, which arises during factoring of stiffness matrix. So, efficiency of direct methods essentially depends on ability of chosen reordering strategy to reduce fill-in. The widespread fill-in reduction technique for today is a minimum degree algorithm family and nested dissection technique [1], [6], [7]. The evolution of reordering algorithms happened into following directions: improving of minimum degree method [7], integration of advantages both: the minimum degree methods and the nested dissection ones [1] and creation of high-efficient multilevel approaches for incomplete nested dissection technique [8], [9].

Second way to enlarge the performance of computations is the application of block technique to factoring procedure. Such a blocking of operations allows one to reduce the low-speed transitions of data between RAM – CPU due to efficient usage of cache memory [2].

The presented here block sparse direct multifrontal solver combines these advantages.

Key features of the proposed method are follows:

- The node-by-node elimination process instead of element-by-element one is used. That's mean an elimination of all equations, associated with current node.
- The nodal reordering procedure is applied instead of element reordering one.
- A front is a C++ class object which encapsulates all data related to a particular node of a FE model. The number of fronts is the same as the number of nodes and the number of elimination steps. Each front contains the elimination node number, the list of frontal nodes, the list of previous fronts (that is, the number of fronts which comprise the given front) and the list of assembled finite elements.
- The elimination process is possible to consider like a movement along the frontal tree.

The solution process consists of following steps.

2.1. Reordering stage

Solver takes the permutation array *Perm*, which assigns the nodal ordering, obtained due to taken reordering method to reduce the fill-ins. The reverse Cuthill-McKee algorithm [6], Sloan method [3], [10], nested dissection method (NDM) [6], minimum multiple degrees method (MMD) [7], parallel section method (PSM_MMD), improved by using of MMD method [7] for each sub-domain, factor-trees method (FQT), multilevel nested dissection method with local MMD reordering (MND_MMD) (idea of this method is taken from [1] and is very close to [8], [9]) are developed in reordering module.

Moreover, the automatic mode is developed to search the proper reordering method among MMD, MND_MMD, PSM_MMD and Sloan methods. This search is based on fast symbolic factorization algorithm [6]. The adjacency graph for finite element nodes is applied instead of equation graph to accelerate the search of proper reordering method.

So, reordering module permits one to select one from mentioned above method or assigns the automatic choice.

2.2. Estimation stage

The permutation array *Perm* sets the order of node-by-node factoring. All equations, associated with given node, are factored on the same frontal step. We set the order of finite element assembling to ensure of given nodal ordering [5]. The symbolic frontal process creates the structure of level for frontal tree and so-called data structure "Process Descriptor" (PD). Only one node (all associated with it equations) is fully assembled and is ready for factoring procedure on each frontal step. The PD data structure contains the list of nodes for each

front, list of previous fronts and list of finite elements, which must be assembled on current frontal step.

We tell the difference between successive, start and nodal fronts. The start front has no any predecessors (previous fronts). The successive front has only single predecessor, and nodal front has several predecessors.

The factorization process is performed in frontal matrix – dense matrix, consisted of fully assembled equations, i.e. equations, associated of corresponding node on given frontal step, and incomplete part. We reorder of fronts to reduce the number of incomplete fronts. The appropriate reordering algorithm is developed for it.

The estimation of required RAM and HD (hard disk) space is produced.

2.3. Factorization stage

Factorization stage consists of node-by-node procedure, which covers all nodes of finite element model. On other hand, factorization process is possible to consider as a movement along structure of levels of frontal tree.

When start or nodal fronts are met, the memory allocation routine reserves the memory space for frontal matrix. The special algorithm look over frontal steps till the next nodal front will meet. The size of frontal matrix is taken to avoid reallocation procedure due to assembling of appropriate finite elements on successive part of frontal tree.

The frontal matrix of nodal front is assembled from frontal matrices of previous incomplete fronts and matrices of finite elements, assembled on this step. As soon a frontal matrix of previous incomplete front is taken for assembling, as corresponding memory space de-allocates. It allows one to ensure the respectively small requirements to RAM.

The frontal matrix of successive front takes the pointer of frontal matrix of previous front. This allows us to avoid the waste transfer of data and allocation – de-allocation procedures.

So, each step of factorization is produced in frontal matrix, consisted of fully assembled equations and incomplete part [5].

2.4. Operations in frontal matrix

The Gauss elimination procedure is applied to partial factorization of frontal matrix in early versions of presented sparse direct multifrontal solver [5]. The main disadvantage of this approach is a low performance of Gauss elimination in frontal matrix. The block Cholesky factorization algorithm is developed instead of column-by-column Gauss elimination one and is presented in proposed paper.

The cluster of equations is associated with each node. So, the several equations are fully assembling simultaneously on each step of frontal process. It allows us to apply the block factoring approach, where size of block equals to the number of equations in cluster. Size of fully assembled block equals to the number of degrees of freedom in node. So, the size of fully assembled block usually equal 6 for 3-D finite element model, for spatial trusses and solids – 3, and so on. Such an argumentation is fairly for unconstrained nodes. If some degrees of freedom in given node is constrained, the number of fully assembled equations is less than it was presented above. It leads to slight decreasing of computational performance, but not destroy of general idea of block factorization.

First of all let us consider the situation, when fully assembled equations are located at the beginning of matrix (Fig. 1). Such an example is considered only to illustrate the principal peculiarities of block Cholesky factorization.

The partial block factorization is possible to be presented as a follows:

$$\begin{bmatrix} \mathbf{D} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{MM} \end{bmatrix} = \begin{bmatrix} \mathbf{L} & 0 \\ \tilde{\mathbf{W}} & \mathbf{M} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S}_{\mathrm{L}} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{L}^T & \tilde{\mathbf{W}}^T \\ 0 & \mathbf{I} \end{bmatrix} \tag{1}$$

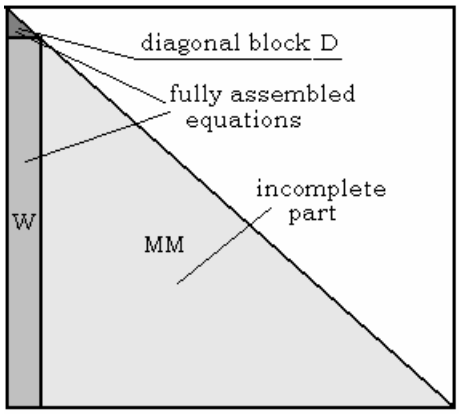

Fig. 1. Permutated frontal matrix

- Factorize diagonal block $\mathbf{D} = \mathbf{L} \cdot \mathbf{S}_{\mathrm{L}} \cdot \mathbf{L}^T$ (1), where $\mathbf{S}_{\mathrm{L}}$ is a sign diagonal. The column-by-column Cholesky algorithm is applied here. Size of diagonal block does not exceed the 6. It is a small matrix, which is possible to allocate fully in cache memory. So, the order of loops in program code does not play of important role.
- Update column $\mathbf{W}$:

$$\begin{aligned} &\mathbf{L} \cdot \mathbf{Y} = \mathbf{W}^T \Rightarrow \mathbf{Y} \\ &\mathbf{S}_{\mathrm{L}} \tilde{\mathbf{W}}^T = \mathbf{Y} \Rightarrow \tilde{\mathbf{W}}^T \end{aligned} \tag{2}$$

The number of right hand sides of (2) is 3 or 6, if 3-D problem is considered. So, the block resolution mode is applied to ensure the level of BLAS-III (Basis Linear Algebra System) [2].

- Update sub-matrix $\mathbf{MM}$:

$$\mathbf{M} = \mathbf{MM} - \tilde{\mathbf{W}} \cdot \mathbf{S}_{\mathrm{L}} \cdot \tilde{\mathbf{W}}^T \tag{3}$$

The $\mathbf{S}_{\mathrm{L}}$ is a diagonal matrix and is stored in core as a vector. So, matrix operation (3) is similar to well-known one $\mathbf{C} = \mathbf{C} - \mathbf{A} \cdot \mathbf{A}^{\mathbf{T}}$. The block matrix product is applied to ensure the level III of BLAS [2].

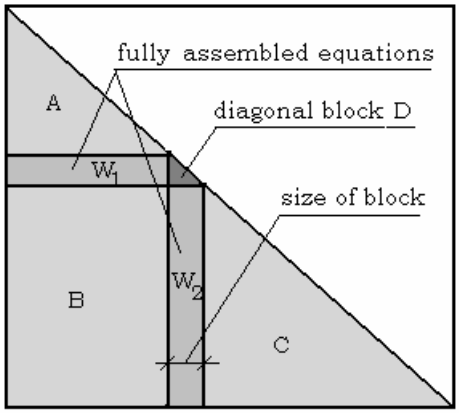

Fig. 2. Block factoring of stiffness matrix

The previous version of sparse direct multifrontal solver [5] realises the non-block mode. If the size of block is taken as 1, the presented above algorithm achieves only level II of BLAS. It is clear, that the previous version of solver [5] corresponds in the best case only to level II of BLAS. This fact explains the advantages of proposed here approach.

In reality, situation is more complicated because the cluster of fully assembled equations is placed in arbitrary part of matrix. The typical structure of frontal matrix is presented on Fig. 2. Theoretically, this case is possible to reduce to case, shown on Fig. 1, by means of permutations. However, this way leads to decreasing of computational performance due to waste operations, caused by indirect access to data.

In proposed version of solver we do not apply any permutations. The matrix is subdivided on 3 sectors: A, B, C. Fully assembled part of matrix $\mathbf{L}, \widetilde{\mathbf{W}}^T$ is stored in a special buffer, and only we produce the steps (1), (2) only in core of this buffer. After factorization of diagonal block and update of sub-diagonal one this part of matrix is a fully decomposed. When buffer is exhausted, this data are stored to secondary storage (hard disk HD), and buffer is cleared to be ready to the next usage. So, factored global finite element matrix is fully reconstituted on disk only after frontal process is finished.

The incomplete part of matrix is presented by blocks A, B, C. The update of these blocks corresponds to (3), but algorithm is more complicated. Data of sector A remain on the same positions, data of sector B are shifted up by number of fully assembled equations n (e.g. by width of block) and data of sector C are shifted up and left by n positions. So, at the end of current frontal step the frontal matrix comprises only incomplete part. Dimension of it is $N-n$, where N – dimension of frontal matrix before factoring step and n is a number of equations in fully assembling cluster (block).

Then we pass to the next frontal step.

3. Examples

3.1. Multi-storey building

The finite element model of multi-storey building (Fig. 3) comprises 195 585 nodes and 204 067 finite elements (1 171 104 equations). The computation time for different modes and ability of multi-frontal solver to reduce non-zero entries in matrix is presented in Tab. 1.

The non-block mode means, that factorization procedure in frontal matrix consists of equation-by-equation (non-block routine) factoring. In this case the performance of computations is slow, because the waste cache – core transition of data presents.

The block mode realises the presented above block factorization in frontal matrix.

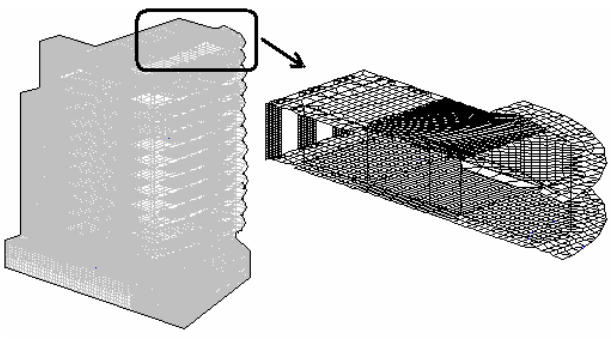

Fig. 3. Multistorey building

Column 1 presents the performance of sparse direct multifrontal solver [5], when non-block mode and QMD/NDM reordering [6] is applied. Column 2 illustrates the efficiency of improved reordering technique – MMD or multilevel nested dissection reordering [7], [8], [9]. Column 3 shows the acceleration of computations due to both: improved reordering and block mode of factoring. All computations have been produced on PC computer Pentium III (1024 MB RAM, 1.2 GHz CPU).

Table 1. Comparison of computation time and non-zero entries

	Non-block mode. QMD [6] reordering	Non-block mode. Multilevel nested dissection reordering [8]	Block mode. Multilevel nested dissection reordering [8]
Column	1	2	3
Time	5 h 44 m	3 h 29 m	1 h 08 m
Non-zero entries, MB	3 246	2 694	2 694

3.2. Ribbed cylindrical shell with discrete-point connections.

The thin cylindrical shell with circular thin-walled ribs is considered (Fig. 4). The contact between shell and rib is produced only in discrete points. Such a model simulates a point-wise welding. It is necessary to create a dense mesh to ensure a good approximation of bending moments and membrane forces in contact zone.

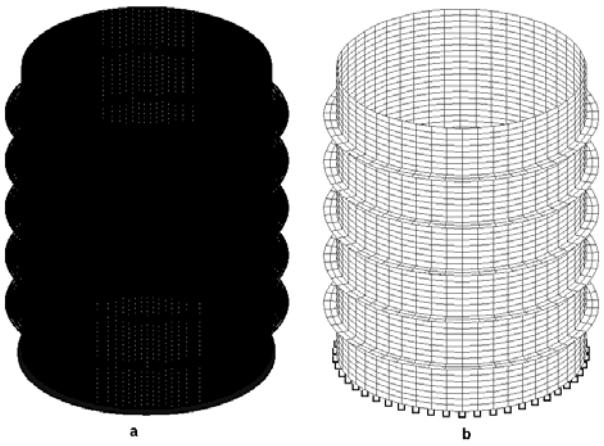

Fig. 4. Ribbed shell with discrete welded connection. a – computational model with fine mesh, b –model with a coarse mesh

The finite element model (fine mesh) comprises 304 200 nodes and 300 300 finite elements (1 819 800 equations). A coarse mesh is presented to clearly illustrate the computation model, because a too dense fine mesh does not provide a good understanding of model on figure.

All computations have been produced on PC computer Pentium III (1024 MB RAM, 1.2 GHz CPU). The results are presented in Tab. 2.

The nested dissection reordering method [6] for non-block mode is a more preferable for this problem than QMD one. The most preferable is a multilevel nested dissection reordering [8]. The block mode and proper reordering method allows us to reduce the computation time almost in 3 times comparing with old version of solver [5].

Table 2. Comparison of computation time and non-zero entries

	Non-block mode. NDM [6] reordering	Non-block mode. Multilevel nested dissection reordering [8]	Block mode. Multilevel nested dissection reordering [8]
Column	1	2	3
Time	9 h 51 m	6 h 40 m	3 h 46 m
Non-zero entries, MB	5 216	4 441	4 441

3.3. Structure – soil interaction problem.

Usually such a problem (Fig. 5) gives rise a stiffness matrix, which is very hard to be optimized to reduce fill-ins for application of direct methods [4]. The finite element model contains 104 048 nodes, 111 269 finite elements and 319 133 equations. A non-uniform mesh on the soil is denser in the building foundation area.

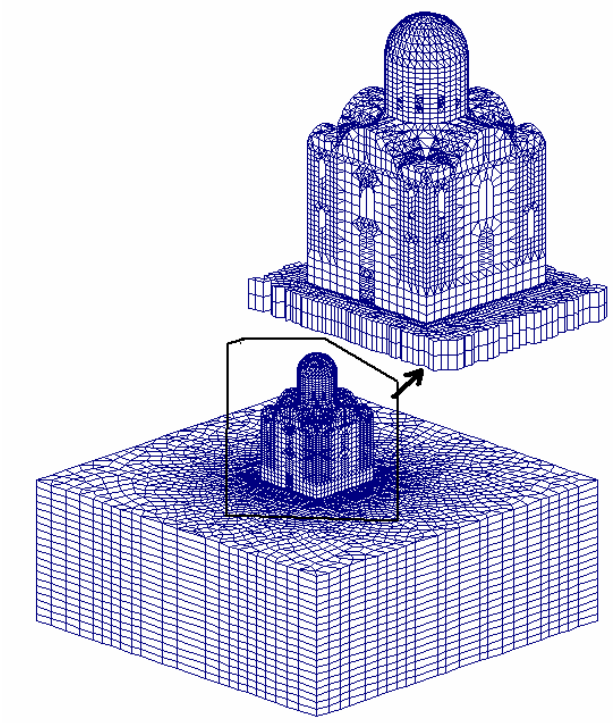

Fig. 5 Structure – soil interaction problem

The previous version of sparse direct multi-frontal solver (the nested dissection reordering method), based on non-block mode of Gauss elimination in frontal matrix, requires about 1292 MB RAM only for allocation of the maximal front (18 403 equations) which is more than the available storage on our computer (PC Pentium-III, CPU Intel-1266 MHz, 1024 MB RAM). The size of non-zero entries in stiffness matrix is 7 869 MB (Tab. 3).

The block version of multifrontal solver, using the multilevel nested dissection reordering, allows one to reduce both: the size of maximal front to 306 MB and the size of non-zero entries to 3 189 MB. It is approximately in three times less comparing with nested dissection reordering method [6]. As a result, this problem has been successfully solved on given computer. The block mode allows us to reduce the computation time almost in two times comparing with non-block Gauss elimination one.

Table 3. Comparison of computation time and non-zero entries

	Non-block mode. NDM [6] reordering	Non-block mode. Multilevel nested dissection reordering [8]	Block mode. Multilevel nested dissection reordering [8]
Column	1	2	3
Time	—	10 h 16 m	5 h 51 m
Non-zero entries, MB	7 869	3 189	3 189

4. Conclusions

The advanced reordering method together with block Cholesky factoring mode of frontal matrix essentially improves the performance of multifrontal solver, presented in [5]. It allows us to apply this method to analysis of very wide range of structural mechanics problems from practice of software company SCAD Soft (www.scadsoft.com).

References

[1] Ashcraft, C. and Liu, J. W.-H., Robust Ordering of Sparse Matrices Using Multisection, *Technical Report CS 96-01, Department of Computer Science*, York University, Ontario, Canada, 1996.

[2] Demmel J. W., *Applied Numerical Linear Algebra*, SIAM, Philadellphia, 1997, Russian edition, Moscow, Mir, 2001.

[3] Duff, I.S., Reid, J.K., Scott, J.A. The use of profile reduction algorithms with a frontal code. *Int. J. Numer. Meth. Eng, 28, 2555—2568 (1989).*

[4] Fialko, S.Yu., An aggregation multilevel iterative solver with shift acceleration for eigenvalue analysis of large-scale structures. *Proceedings of the CMM-2003 – Computer Methods in Mechanics June 3-6, 2003*, Gliwice, Poland. pp. 125 – 126., June 3-6, 2003

[5] Fialko, S.Yu., Kriksunov, E.Z. and Karpilovskyy V.S., A sparse direct multi-frontal solver in SCAD software, Proceedings *of the CMM-2003 – Computer Methods in Mechanics*, Gliwice, Poland. pp. 131 – 132., June 3-6, 2003

[6] George, A. and Liu, J. W.-H., *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[7] George, A. and Liu, J. W.-H., The Evolution of the Minimum Degree Ordering Algorithm, *SIAM Rev. 31, 1-19 (March 1989).*

[8] Karypis G., and Kumar V., A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs, *Technical Report TR 95-035, Department of Computer Science, University of Minnesota*, Minneapolis,1995.

[9] Karypis G., and Kumar V., METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, *Technical report, Department of Computer Science*, University of Minnesota, Minneapolis, 1995.

[10] Scott, J.A., On ordering elements for a frontal solver. *Commun. Numer. Meth. Engng, 15. 309—323 (1995).*