A sparse direct multifrontal solver in SCAD software

Sergiy Yu. Fialko, Edward Z.Kriksunov and Viktor S.Karpilovskyy
Software company SCAD Soft
13, Chokolovsky bld., room 508
Kiev, 252680 GSP, Ukraine
e-mail: fialko@erriu.ukrtel.net

Abstract

A sparse direct multi-frontal method (MFM) for solving large-scale finite element linear algebraic equations is presented. Both the minimum degree algorithm (MDA) and the nested dissection method (NDM) are applied to obtain a proper ordering of equations for reduction of fill-ins during the factorization. An automatic selection of a more efficient reordering method is based on a fast symbolic factorization. This method allows to essentially reduce the computing time comparing to the prevailing skyline solver based on a reverse Cuthill-McKee algorithm (RCM). The efficiency of the proposed approach is illustrated by numerous large-scale finite element models of real buildings. This method is implemented in the SCAD commercial software (http://www.scadgroup.com/eng/).

Keywords: sparse, multi-frontal, ordering, frontal tree

1. Introduction

Sparse direct methods [4] make a powerful tool for solution of large-scale finite element problems, especially when ill-conditioned problems need to be solved. In such case iterative methods show a slow convergence. An efficient direct method based on sparse reordering MDA (minimum degrees algorithm) or NDM (nested dissection method) approaches and the multi-frontal technique is presented here. The principal effort of the authors is aimed at a reduction of fillings in the course of the Gauss elimination procedure [4]. The attention is focused on the proposed solver implementations with commonly popular PCs to extend the capabilities of analysing real large-scale engineering problems and to reduce the cost of the finite element analysis.

A properly chosen reordering method ensures the reduction of fillings during Gauss elimination or Choletsky factorization. The more fillings are reduced, the less the computational effort. The reverse Cuthill-McKee algorithm (RCM) is a prevailing reordering method which has been implemented in commercial finite element software until recently. The development of fast problem-oriented graphic pre-processors and automatic mesh generators causes the dimensions of finite element (FE) models to grow. For example, the usual size of SCAD client FE problems is about 90 000 - 300 000 degrees of freedom for today. Such large-scale problems require the implementation of advanced solution techniques because skyline solvers are still too much time-consuming.

An alternative approach is to use sparse direct solvers which appear to be more efficient even than the profile reduction techniques based on Sloan or spectral reordering methods [3].

The multi-frontal solution technique [1],[2],[3],[5] proves to be convenient for implementing in commercial and research FE software.

2. Sparse multifrontal method MFM

The MFM method is based on a combination of advantages of sparse ordering methods – the minimum degree algorithm MDA and the nested dissection method NDM [4] with the frontal [9] and multi-frontal techniques.

Key features of the proposed method follow:

- The solution of a FEM equation system consists of node-by-node elimination of equations referred to a particular node (a nodal equation set). So, the elimination process includes a number of steps equal to the number of nodes in the finite element model. Constrained degrees of freedom do not contribute their corresponding equations to the nodal equation set.
- The term “elimination of node” means the elimination of a nodal equation set.
- As opposed to the element reordering in conventional frontal or multi-frontal solution techniques, the proposed method uses a nodal reordering. It allows us to apply well-known reordering algorithms, like the minimum degrees algorithm or nested dissection method [4].
- A fast symbolic factorization [4] is applied to choose a proper reordering method for a problem: MDA or NDM.
- 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.
- A Process Descriptor data structure is created to establish the sequence of FE assembling according to the specified node reordering, sequences of fronts, lists of nodes and equations for each front.
- The set of fronts makes a frontal tree. The elimination process is a movement along the frontal tree.
- The current front is a start one if it has no predecessors (previous fronts). It is a nodal front if it has more than one predecessor, and it is a successive one if it has only one predecessor.

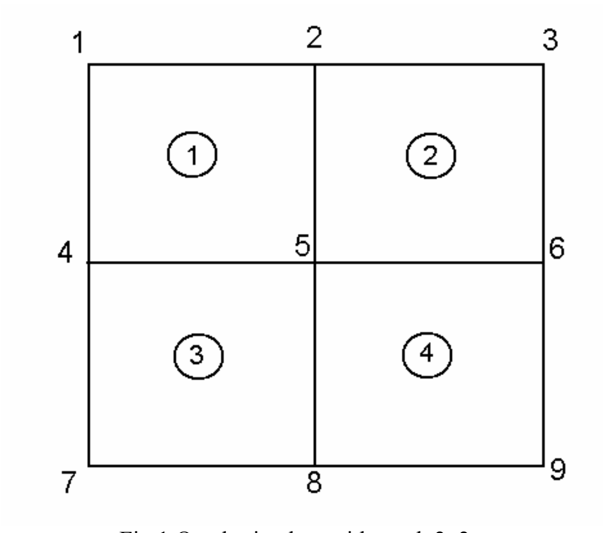

Fig.1 Quadratic plate with mesh 2x2

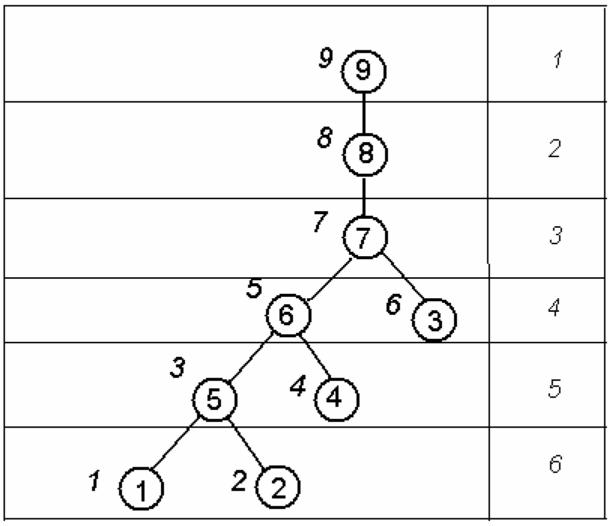

Fig. 2 Structure of levels for frontal tree

- Moreover, each front contains a pointer to a frontal matrix – a dense matrix consisting of both fully assembled equations referred to the current node being eliminated, and partially assembled equations related to other nodes of the current front. The equations correspond to eliminated unknowns and are stored on disk, and the remaining equations create an incomplete front. If the size of incomplete fronts exceeds the capacity of the core memory, those are saved to disk.
- The frontal tree is reordered to reduce the space required by the incomplete frontal matrices.
- A successive front inherits the frontal matrix of its previous incomplete front by accepting its pointer (it is a very fast operation). Then the stiffness finite element matrices corresponding to FE added at this solution step, are added to the incomplete frontal matrix, and unknowns for fully assembled equations are eliminated.
- The frontal matrix of the nodal front is assembled from the frontal matrix of the previous fronts and element matrices of finite elements which are added at this step.
- The frontal matrix of the start front is assembled only from element matrices of corresponding finite elements.

Let us consider a simple example – a square plate with a mesh 2x2 (Fig. 1) which is to illustrate the basic concepts of the proposed method.

The minimum degrees algorithm produces the following order of nodes to be eliminated: 1,3,7,9,2,6,8,4,5 .

Then, we define a sequence of the finite element assembling. The node is fully assembled if all finite elements that contain are assembled. The nodal equation set for a fully assembled node is ready to be eliminated because the continuation of the assembly does not change these equations. Table 1 is filled to obtain the sequence of the finite element assembly according to the specified nodal reordering. Each finite element can be assembled only once. Therefore, each assembled finite element is greyed to avoid multiple assembling. So, the greyed element numbers present the finite element assembling sequence where we move from the top to the bottom of table. It means that the global stiffness matrix is assembled in the following order:

$$\mathbf{K} = \sum_{e=1}^{N_e} \mathbf{K}_{NewNo[e]} \tag{1}$$

Table 1: Sequence of the finite element assembling

Number of eliminated node	List of FE required to assemble a nodal equation set
1	1
3	2
7	3
9	4
2	1,2
6	2,4
8	3,4
4	1,3
5	1,2,3,4

Table 2. Process Descriptor data structure

Elimination step	Eliminated node	List of frontal nodes	List of previous fronts	List of assembling FE
1	1	1,2,4,5	-----	1
2	3	3,6,5,2	-----	2
3	7	7,4,5,8	-----	3
4	9	9,8,5,6	-----	4
5	2	2,4,5,6	1,2	----
6	6	6,8,5,4	5,4	----
7	8	8,5,4	6,3	----
8	4	4,5	7	----
9	5	5	8	----

where $\mathbf{K}_{NewNo[e]}$ is an element's stiffness matrix and $NewNo[e],\ e=1,2,...,N_e$ is a permutation vector that defines the required order of finite element assembling, N_e is the number of finite elements.

The Process Descriptor data structure contains general information about the assembling/elimination process and is presented in Table 2. The elimination of node #1 requires the finite element #1 to be assembled (see Table 1). Finite element #1 imports nodes #1,2,3,4. Node #1 is fully assembled and all nodal unknowns are eliminated. The transformed rows of the frontal matrix corresponding to the nodal matrix of node #1, make a part of the upper matrix factor and are saved to disk. The equation sets for nodes #2,4,5 constitute the incomplete front.

The elimination of nodes #3,7,9 is performed similarly because their corresponding fronts 2,3,4 are start ones and do not require the assembling of previous fronts.

The elimination of node #2 requires the assembling of incomplete fronts 1,2 which are taken from core (if the incomplete front is located in the core memory) or from disk (if the incomplete front has been saved to disk). The nodes eliminated at the previous steps are not included in the frontal node list of the current front, because corresponding unknowns are eliminated already and appropriate parts of the frontal matrices are saved to disk. As soon as the incomplete front is taken to the assembly, the corresponding part of the core memory gets freed. Thus the size of the core memory remains relatively small even for large-scale problems. The current solver version provides a virtualisation if there is not enough core memory. The elimination of nodes #6,8,4,5 is performed exactly as that of node #2.

The structure of levels of the frontal tree (Fig. 3) is based on the Process Descriptor data structure and is created in the following way: we take the last front #9 (elimination step 9) and place it at the top (level 1) of the structure of levels. The front #8 is a predecessor for front #9 (see Table 2), so we place it at level 2. Front #7 is a predecessor for front #8 - place it at the level 3. Fronts #9,8 are successive ones. Front #7 is a nodal one, because it has two predecessors - fronts #6, 3 which form level 4. Front #3 is a start one, but front #6 is nodal. And so on, until all fronts are exhausted.

The assembling/eliminating process can be presented as a movement along the structure of levels of the frontal tree from its bottom to its top. The unknowns of a front from level k can be eliminated only after the elimination of all unknowns of previous fronts from level $k+1$ which are predecessors of the front from level k.

The reordering of fronts is performed to reduce the space occupied by incomplete fronts. The figures in circles denote the original front numbers and the figures placed to the right are the reordered front numbers.

The objective of this paper is to present an efficient solution method for usual (sequential) computations that involves a reduction of fillings due to the use of a proper reordering method. However, this algorithm also enables splitting the Gauss elimination procedure because of parallel branches in the frontal tree, which can be useful for parallel computing.

The Gauss elimination process is performed step-by-step according to the sequences of reordered nodes. Each front is considered to be an object containing a frontal matrix. The memory allocation and assembling of frontal matrix are performed for start and nodal fronts.

The list of local-global equation numbers *List(local_eqn_number) = global_eqn_number* establishes a connection between the equation numbers in the frontal matrix (local equation numbers) with equation numbers in the global stiffness matrix $\mathbf{K}$ and is based on the list of frontal nodes (see Process Descriptor data structure - Table 2). Constrained degrees of freedom do not contribute any equations to the frontal matrix.

Next, unknowns for fully assembled equations corresponding to an eliminated node at the current step (see Table 2) are eliminated, and the remaining part of the frontal matrix is still an incomplete front. The transformed rows of the frontal matrix corresponding to eliminated unknowns, are saved to disk. The incomplete fronts are saved to disk if the capacity of the core memory is exceeded.

If a successive front is met, the memory is not allocated because such front accepts a pointer to the frontal matrix from its predecessor, the previous front. Corresponding finite element matrices (see Table 2) are added if necessary. Then the elimination of unknowns is performed for the node that must be eliminated.

The Gauss elimination procedure is performed in the dense frontal matrix. Fully assembled equations may be located in any arbitrary rows. It introduces additional complications to the Gauss elimination algorithm. The following example – a square symmetric matrix 4x4 – demonstrates this. The initial matrix is shown below:

$$\begin{Bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{Bmatrix} \tag{2}$$

Let the equation 2 be fully assembled and eliminated. The second row is placed in a buffer for the factored part of the matrix and will be saved to disk after the current elimination step. The diagonal element a_{22} is a pivot entry. The matrix is divided into three sectors: A, B, C :

$$\begin{array}{ccc} & & a_{21} \quad a_{22} \quad a_{23} \quad a_{24} \\ ------ & & ----------- \\ \begin{Bmatrix} A & \# & B \\ \# & \# & \# \\ & \# & C \end{Bmatrix} & = & \begin{Bmatrix} a_{11} & \# & a_{13} & a_{14} \\ \# & \# & \# & \# \\ & \# & a_{33} & a_{34} \\ & \# & & a_{44} \end{Bmatrix} \end{array} \tag{3}$$

where $a_{11} \in A$, $a_{13}, a_{14} \in B$ and $\begin{Bmatrix} a_{33} & a_{34} \\ & a_{44} \end{Bmatrix} \in C$. Here only the top triangle part of the matrix is presented because it's symmetric. The row located in the buffer is shown under the matrix.

The elimination in sector A is performed in a regular way:

$$\hat{a}_{11} = a_{11} - a_{12} / a_{22} \cdot a_{21} \tag{4}$$

The resulting value $\hat{a}_{11}$ is placed in position of a_{11}.

The elimination in sector B occurs as follows:

$$\begin{aligned}\hat{a}_{13} &= a_{13} - a_{12}/a_{22} \cdot a_{23} \\ \hat{a}_{14} &= a_{14} - a_{12}/a_{22} \cdot a_{24}\end{aligned} \tag{5}$$

The resulting values $\hat{a}_{13}, \hat{a}_{14}$ are shifted by one position to the left.

In the sector C:

$$\begin{aligned}\hat{a}_{33} &= a_{33} - a_{32}/a_{22} \cdot a_{23} \\ \hat{a}_{34} &= a_{34} - a_{32}/a_{22} \cdot a_{24} \\ \hat{a}_{44} &= a_{44} - a_{42}/a_{22} \cdot a_{24}\end{aligned} \tag{6}$$

The resulting values $\hat{a}_{33}, \hat{a}_{34}, \hat{a}_{44}$ are shifted by one position to the left and one position to the top. The final results are:

$$\begin{array}{cccc} a_{21} & a_{22} & a_{23} & a_{24} \\ \hline \end{array}$$
$$\left\{\begin{array}{cccc} \hat{a}_{11} & \hat{a}_{13} & \hat{a}_{14} & \# \\ & \hat{a}_{33} & \hat{a}_{34} & \# \\ & & \hat{a}_{44} & \# \\ & & & \# \end{array}\right\} \tag{7}$$

The top row – a part of the factored matrix - should be saved to disk, and the remaining part of the matrix contains an incomplete front, so it is kept in the core memory as long as the next needed steps are assembled.

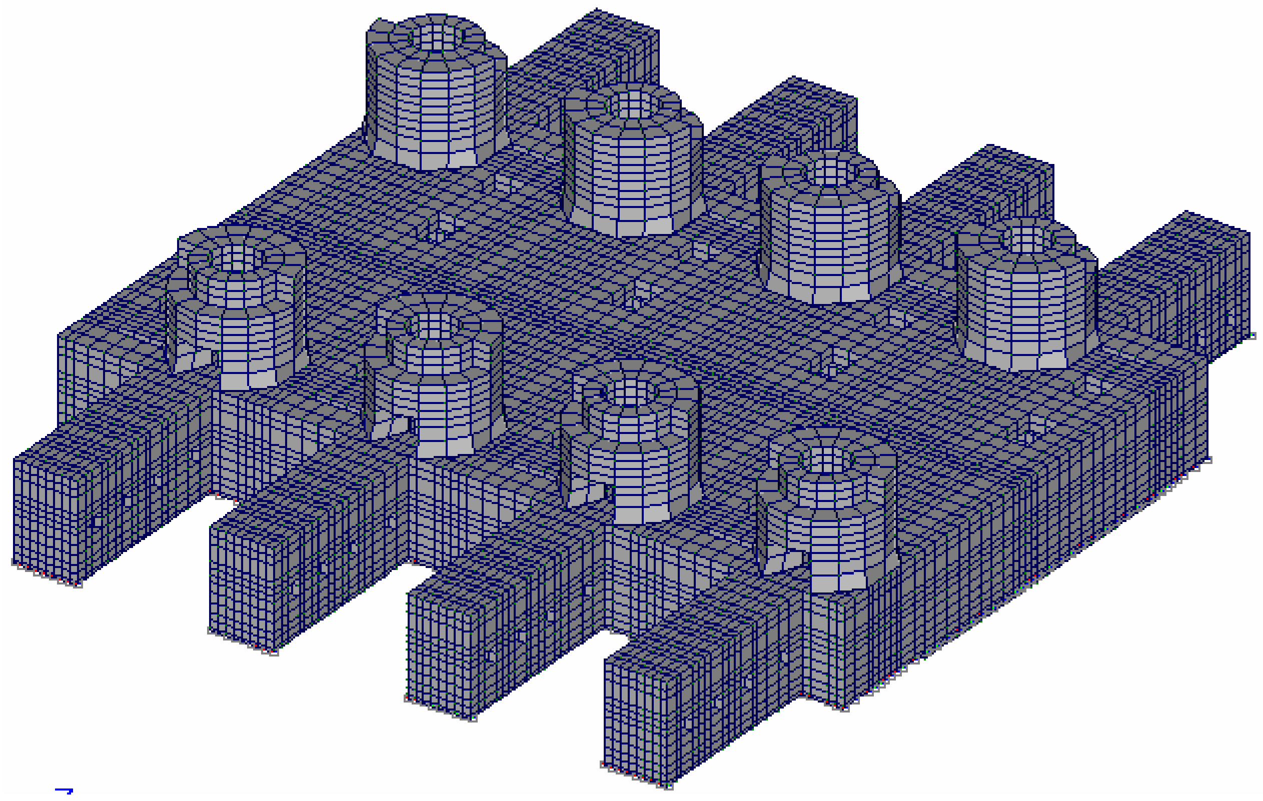

Fig. 3 A reactor unit of a nuclear power plant

3. Numerical results

A few typical examples of SCAD usages are presented to illustrate the capabilities of the method.

3.1. A reactor unit of a nuclear power plant

The finite element model contains 53 946 nodes, 40 264 spatial finite elements and 160 929 equations (Fig.3). A comparison of the performance of different solvers is presented in Table 3.

Designations: skyline is a direct solver with the profile storage approach [4] and the reverse Cuthill-McKee (RCM) reordering method; MFM (MDA) is a multi-frontal solver with minimum degrees algorithm [4] reordering, and AMIS is a fast aggregation multilevel iterative solver [6], [7], [8]. It is a preconditioned conjugate gradient method with aggregation multilevel preconditioning. The tolerance $tol = 1.0e-04$ has been accepted for the AMIS iterative solver. It means that the solution error is

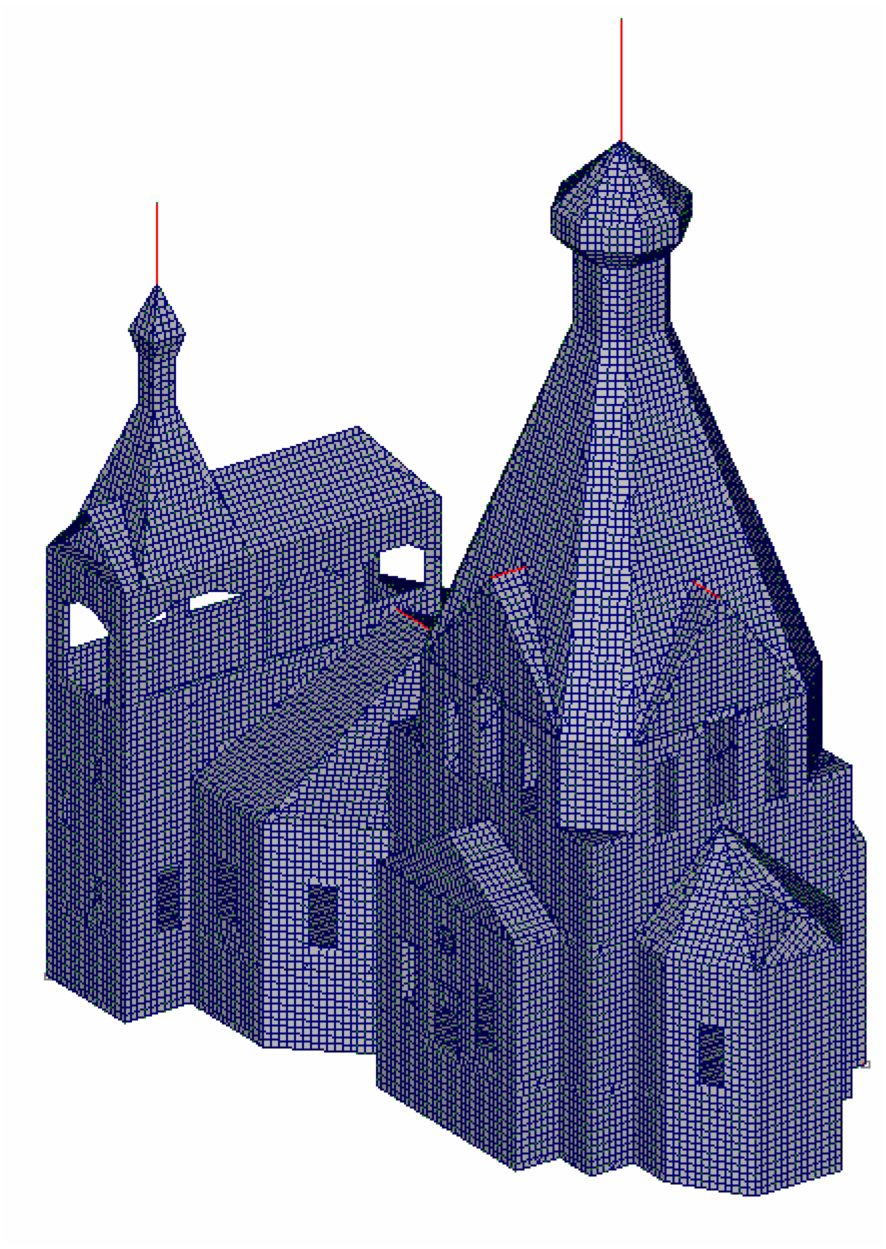

Fig.4 A finite element model of a church

Table 3: Comparison of efficiency of different solvers (the reactor block)

Solver	Computation time (Computer: Pentium III, Intel 1000 MHz processor, 512 MB RAM)	Size of nonzero entries in the factored stiffness matrix (only for direct solvers), MB
Skyline	9 h 30 min 04 s	3 552
MFM (MDA)	1 h 33 min 03 s	955
AMIS	20 min 29 s	---------

$\|\mathbf{r}\| = \|\mathbf{b} - \mathbf{Kx}\| / \|\mathbf{b}\| \leq tol$ where $\mathbf{x}, \mathbf{b}$ is the solution vector and the right side vector (r.h.s.v.) respectively. This problem contains 3 r. s. v. The solution time of the direct methods depends on the number of r. s. v. very little (at least for a small number of r. s. v.). On the other hand, the iterative solver runs every time from the beginning of each load case. Therefore the solution time of the iterative solver depends on the number of r .s. v. essentially.

The proposed multi-frontal method reduces the computation time about 6 times comparing to the conventional skyline solver. The AMIS solver seems preferable for this model that contains spatial finite elements.

3.2. *Finite element model of a church*

The model contains 52 752 nodes, 55 602 finite elements and 316 509 equations (Fig. 4). A single r .s. v. is considered. A comparison of the performance of different solvers is presented in Table 4.

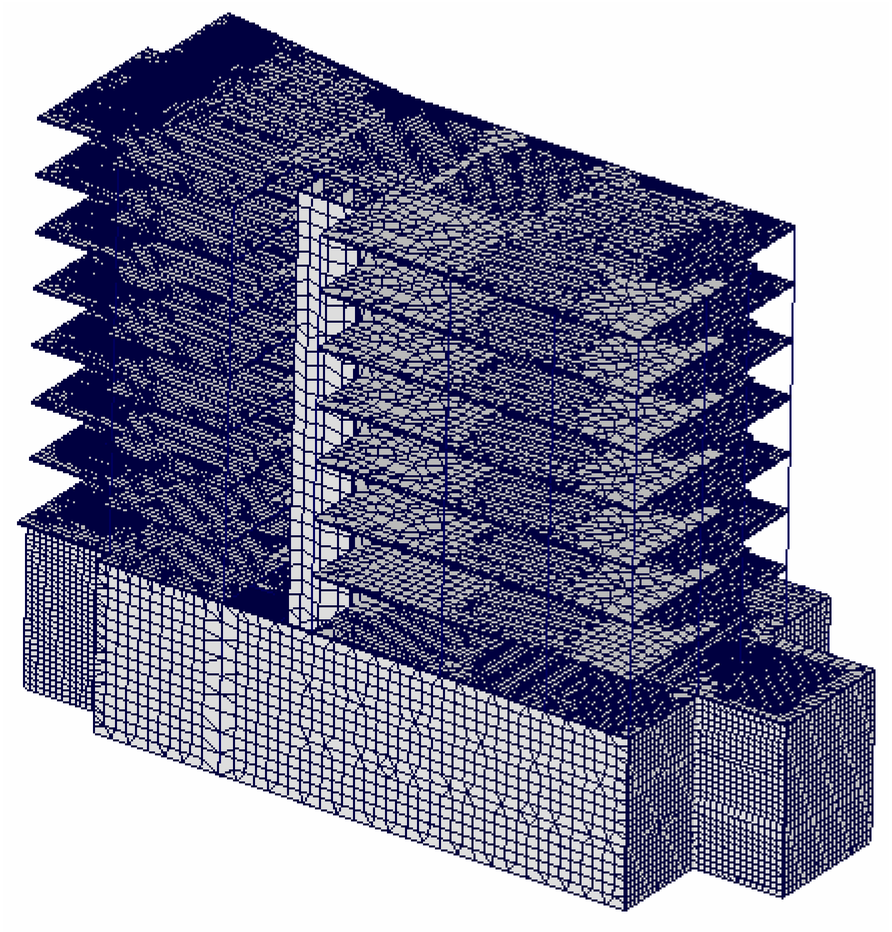

Fig.5 A finite element model of a multi-storey building

Table 4: Comparison of efficiency of different solvers (a finite element model of a church)

Solver	Computation time (Computer: Pentium III, Intel 1000 MHz processor, 512 MB RAM)	Size of nonzero entries in the factored stiffness matrix (only for direct solvers), MB
Skyline	>> 25 h	8 896
MFM (MDA)	43 min 45 s	720
AMIS	59 min 14 s	--------

The tolerance $tol = 1.0e-04$ has been accepted for the AMIS iterative solver. When the runtime of the skyline solver with RCM reordering method exceeded 25 hours, still less than one third of the stiffness matrix had been factored, so the job was cancelled.

The multi-frontal solver has proved to be most efficient of the methods considered. The aggregation multilevel iterative solver requires a bit more time than the multi-frontal solver.

3.3. Finite element model of a multi-storey building

The model contains 91 089 nodes, 96 656 finite elements and 544 410 equations. Four r. s. v. are considered. A comparison of the performance of different solvers is presented in Table 5.

Table 5: Comparison of efficiency of different solvers (a multi-storey building)

Solver	Computation time (Computer: Pentium III, Intel 600 MHz processor, 512 MB RAM)	Size of nonzero entries in the factored stiffness matrix (only for direct solvers), MB
Skyline	23 h 30 min	5 830
MFM (MDA)	35 min 22 s	763
AMIS	1 h 02 min 30 s	--------

The tolerance $tol = 1.0e-04$ has been accepted for the AMIS iterative solver. The runtime of the skyline solver with the RCM reordering method is about 24 hours.

The multi-frontal solver has proved to be most efficient of the methods considered. It reduces the computation time about 40 times comparing to the skyline solver. The aggregation multilevel iterative solver requires a bit more time than the multi-frontal solver. It should be noted that the extraction of one r. s. v. plus the preparation of preconditioning takes about 20 min.

The advantage of the multi-frontal solver for this problem is due to the fact its solution time depends little on the number of r. s. v. while the iterative solver starts iterations from the beginning for each r. s. v.

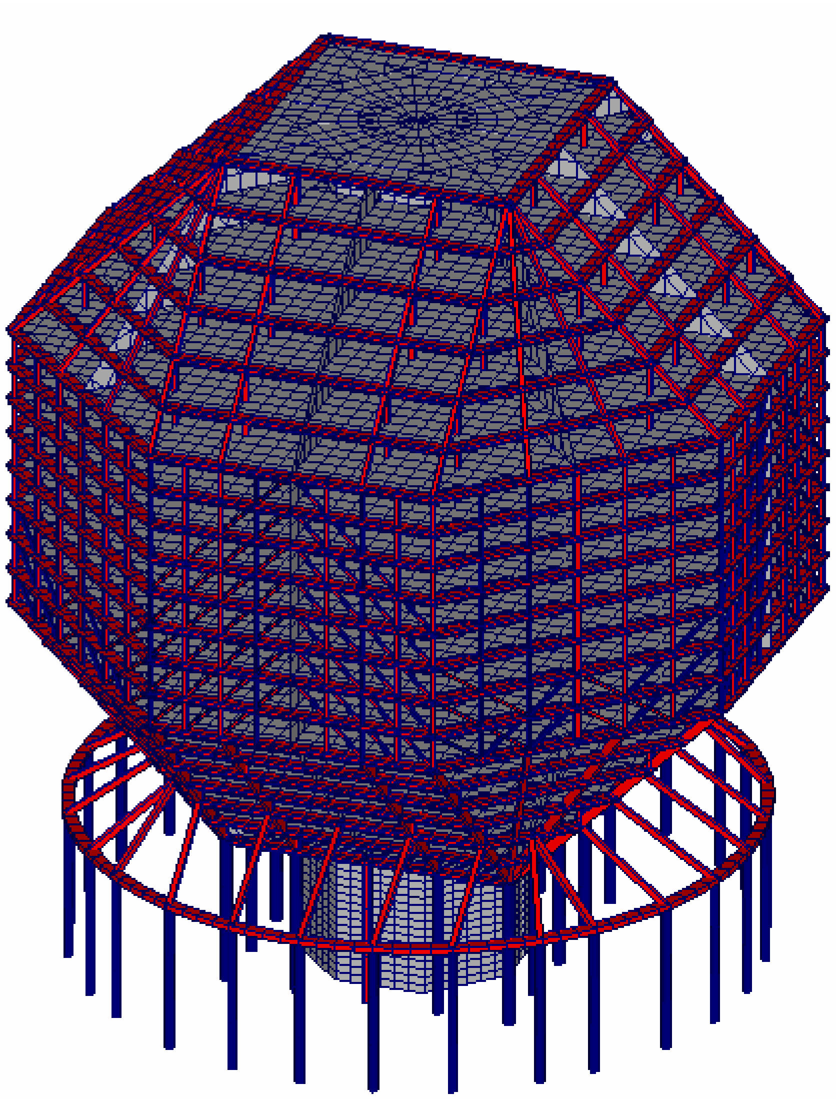

Fig.6 A finite element model of Byelorussia National Library in Minsk

3.4. Finite element model of a multi-storey building of Byelorussia National Library in Minsk

The model contains 34945 nodes, 46186 finite elements and 207 978 equations. A comparison of the performance of different solvers with it is presented in Table 6.

The tolerance $tol = 1.0e - 04$ has been accepted for the AMIS iterative solver.

The multi-frontal solver reduces the computation time 30 times comparing to the skyline solver. The aggregation multilevel iterative solver AMIS (4 r. s. v.) requires only 37 min to produce the solution with a practical accuracy.

Table 5: Comparison of efficiency for different solvers (Byelorussia National Library in Minsk)

Solver	Computation time (Computer: Pentium III, Intel 1000 MHz processor, 512 MB RAM)	Size of nonzero entries in the factored stiffness matrix (only for direct solvers), MB
Skyline	48 h 12 min 36 s	10 514
MFM (MDA)	1 h 35 min 32 s	806
AMIS	36 min 59 s	----------

4. Conclusion

The proposed multi-frontal solver implemented in the SCAD software has proved to be a robust and fast method for solving a variety of types of structural mechanics problems from actual civil engineering practice. For many large-scale problems it is competitive even with the aggregation multilevel iterative solver [6], [7], [8], especially when there are extensive right sides or ill-conditioned matrices.

References

[1]. Duff I.S., Reid J.K., The multifrontal solution of indefinite sparse symmetric linear equations, *ACM Trans. Math. Software*, 9, pp. 302-325, 1973.

[2]. Duff, I.S., Parallel implementation of multifrontal scheme, *Parallel Comput.*, 3, pp. 193-204, 1986.

[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, pp. 2555-2568, 1989.

[4]. George, A., Liu, J., *Computer solution of large sparse positive definite systems*. Prentice-Hall, Inc., 1981.

[5]. Gend, P., Oden, J.T., R.A van der Geijn., A parallel multifrontal algorithm and its implementation, *Comput. Methods Appl. Mech. Engrg.*, 149, pp. 289-301, 1997.

[6]. Fialko, S.Yu., The high-performance aggregation element-by-element iterative solver for the large-scale complex shell structural problems, *Archives of Civil Eng*., XLV, 2, pp. 193-207, 1999.

[7]. Fialko, S., High-performance aggregation element-by-element Ritz-gradient method for structure dynamic response analysis, *CAMES*, 7, pp. 537-550, 2000.

[8]. Fialko, S.Yu., High-performance iterative and sparse direct solvers in Robot software for static and dynamic analysis of large-scale structures. *Proceedings of the second European conference on computational mechanics*, Poland, June 26-29, 2001, 18 p.

[9]. Irons, B., A frontal solution program for finite element analysis, *Int. J. Numer. Meth. Engrg*., 2, pp. 5-32, 1970.