# A parallel sparse direct finite element solver for desktop computers

**Sergiy Fialko**[1]

[1]*Department of Physics, Mathematics and Computer Science of Tadeusz Kosciuszko Cracow University of Technology*
*Warszawska 24 St., 31-155 Kraków, Poland*
*e-mail: sfialko@poczta.onet.pl*

Abstract

The parallel finite element solver, which is based on block $\mathbf{L}\cdot\mathbf{S}\cdot\mathbf{L}^T$ factoring, where $\mathbf{S}$ is a sign diagonal, is presented. The sparse symmetric positive definite and indefinite matrices are considered. Unlike the methods presented in the libraries of high-performance procedures, including PARDISO solver, the proposed approach uses the disk storage, if the dimension of the problem exceeds the capacity of random access memory RAM. Two out of core modes OOC and OOC1 are developed. The OOC mode produces the minimal I/O operations, ensures the stable speed-up when the number of processors increases, but requires respectively larger amount of RAM. The OOC1 mode demands the smaller amount of RAM, but the number of I/O operations is significantly greater than in OOC mode and speed up is much poorer. The presented method is essentially faster than multi-frontal solver, especially on multiple processors, because it is not using redundant data transfer from one memory buffer to another.

*Keywords: sparse direct solver, finite element method, parallel computing, multithreading, high performance, desktop computers*

## 1. Introduction

Currently, large engineering problems are increasingly being solved on the desktop, rather than on workstations, clusters, and computer networks. The main features of these computers are the limited RAM and low bandwidth of memory system. Therefore, many algorithms, which are well-established on computers with distributed memory, are often less effective for desktop computers with multiple processors and shared memory. So, the development of special methods for multiprocessor desktop computers is required. This article is devoted to block looking-left decomposition of the sparse symmetrical matrix:

$$\mathbf{K} = \mathbf{L}\cdot\mathbf{S}\cdot\mathbf{L}^T, \qquad (1)$$

where $\mathbf{S}$ is a sign diagonal.

For today, the multi-frontal method is the most widespread among direct methods for solving systems of linear algebraic equations in the FEA software [2], because it carefully uses the RAM and involves the disk memory when the dimension of problem exceeds the resources of RAM [3] – [5]. On the other hand, the multi-frontal method produces a redundant coping of data from one region of memory to another. These operations are only slightly accelerated on shared-memory computers with the increase in the number of processors [6].

The solvers from high-performance libraries demonstrate a good performance and speed-up, but become helpless if the dimension of the problem exceeds the capacity of core memory.

PARDISO solver [7] from Intel Math Kernel Library (MKL) exhibits fine results. According to MKL manual, it supports the out-of-core mode (OOC), but produced tests show that it can solve only small problems in this mode and works with low productivity [6]. Therefore, we assign PARDISO to the previous group of solvers.

This paper presents a parallel finite element solver PARFES which shows a high performance on each processor and stable speed-up for large problems and involves a disk memory when a dimension of problem exceeds a capability of RAM.

Unlike the previous version presented in [6], this paper generalizes the method for indefinite matrices (1) and adds OOC1 mode, which allows one to solve large problems on computers with small memory, such as laptops.

## 2. Analysis phase

The analysis phase creates the adjacency graph (AG) for the nodes of finite element models, produces a reordering to reduce fill-ins, preparing an elimination tree and finding its super-nodes. As opposed to algebraic methods PARFES uses the nodal AG instead of AG for sparse matrix [1]. Each vertex of nodal AG corresponds to block-column in sparse matrix and each edge of AG fits a dense block in such a block-column. The width of each block-column equals to the number of degrees of freedoms in a given node. The symbolic factorization, produced after reordering, creates the nonzero structure $L$ of low triangle factor-matrix.

The initial dimensions of blocks are too small for achieving high performance and conventional super-nodal technique is applied to enlarge of dense blocks. After such a procedure the sparse matrix is divided to rectangular dense blocks and high performance matrix multiplication algorithms from Intel MKL library are applied during numerical factoring. The details are presented in [6].

## 3. Numerical factorization phase

The block left-looking algorithm is applied to produce the numerical factoring:

1. if(core mode)
   prepare block-columns $jb \in [1, N_b]$
2. do $jb = 1, N_b$
3. if(OOC $\vee$ OOC1)
   prepare block-column $jb$
4. Parallel correction of block-column $jb$:

$$\mathbf{A}_{ib,jb} = \mathbf{A}_{ib,jb} - \sum_{kb \in List[jb]} \mathbf{A}_{ib,kb} \cdot \mathbf{S}_{kb} \cdot \mathbf{A}^T{}_{jb,kb}; \quad ib \geq jb, ib \in L$$

5. Factoring of block-column $jb$:

$$\mathbf{A}_{jb,jb} = \mathbf{L}_{jb,jb} \cdot \mathbf{S}_{jb} \cdot \mathbf{L}^T{}_{jb,jb};$$

   parallel loop $ib \geq jb$, $ib \in L$:

$$\mathbf{L}_{jb,jb} \cdot \mathbf{S}_{jb} \cdot \mathbf{L}^{T}_{ib,jb} = \mathbf{A}^{T}_{ib,jb} \quad \rightarrow \quad \mathbf{L}_{ib,jb};$$

if(OOC ∨ OOC1)

write block-column *jb* to disk and free RAM for block-row *ib* = *jb*.

6. Add *jb* to *List*[*lb*], *lb* > *jb*, if block-column *jb* corrects the block-column *lb*.
7. end do.

Here $N_b$ is a number of block-columns in matrix, *jb* –block-column number, *ib* –block-row number ($ib \geq jb$, $ib \in L$), $kb \in List[jb]$, *List*[*jb*] contains the numbers of block-columns, which are located at left from block-column *jb*, and corrected it.

The current block-column *kb* is read from disk, when OOC1 mode is turned on (step 4). The core memory buffer must contain the block-column *jb* and part of current block-column *kb* ($ib \geq jb$). This leads to the lowest requirements of core memory, but to intensive disk accesses.

For OOC mode it is required to store the block-column *jb* and parts all block-columns $kb \in List[jb]$ ($ib \geq jb$) in core memory. On the other hand, the number of I/O operations with disk is a minimal.

Step 4 produces about 98% – 99% of floating point operations; therefore it must be paralleled primarily. We map the entire block-row to a single processor to avoid a synchronization procedure, which usually leads to degradation of performance of such algorithms. To ensure a load balance between processors, the weights – number of nonzero entries for each block-row – are computed. The block-rows are sorted in descending order of weights and are consistently mapped at the processors. The next block-row is assigned to processor that has a minimum amount of weights [6].

## 4. Numerical results

We compare three solvers: PARFES, PARDISO [7] and BSMFM (block substructure multi-frontal method), which is not inferior in performance to multi-frontal solver of ANSYS v11 software at least on the model problems [4] – [6]. So, it can be taken for such a comparison as a good realization of multi-frontal solver.

The design model of multi-storey block concrete building is taken from computational practice of SCAD Soft (www.scadsoft.com) and comprises 3 198 609 equations.

The four-core desktop computer with processor AMD Phenom™ II x4 995 3.2 GHz, RAM DDR3 1033 MHz 16 GB, OS Windows Vista™ Business (x64), Service Pack 2 is taken.

All solvers run in OOC mode on 64 bit platform. PARDISO has failed. The time of numerical factorization for PARFES on four processors is about 3 times less than one for BSMFM ([6], tab. 8, p. 1263). Moreover, PARFES shows a stable speed-up even for OOC mode ($S_p=T_1/T_p$ = 3.1 for $p$ = 4, where $T_1$ - computing time on single processor, $T_p$ – on $p$ processors). BSMFM has $S_p$ = 1.4 on $p$ processors ([6], fig. 11, p. 1263).

Results for OOC1 mode are presented in tab. 1. The both: PARDISO solver as well as BSMFM does not solve this problem on 32 bit platform due to insufficient core memory. So, PARFES demonstrates the possibility to solve large problems with limited core memory.

This problem has been solved on computer with two six-core processors Intel Xeon X5660 @ 2.8 GHz /3.2 GHz (12 cores), RAM DDR3 24 GB. All solvers run in a core mode. Duration on numerical factorization is presented in tab. 2.

PARFES and PARDISO show a very close results. The minimal time of numerical factoring is about 80 s on 11 threads. The minimal time for BSMFM solver is 770 s.

Table 1. Time (s) of numerical factoring, OOC1 mode, platform ia32, AMD Phenom™ II x4 995 based computer

| Method | Number of processors | | | |
|---|---|---|---|---|
| | 1 | 2 | 3 | 4 |
| PARFES | 2199 | 1525 | 1189 | 1139 |
| PARDISO | Numer. factorization phase: error = -9 | | | |
| BSMFM | Insufficient core memory | | | |

Table 2. Time (s) of numerical factoring on computer with Intel Xeon X5660 @ 2.8 GHz /3.2 GHz processor, platform ×64, core mode

| Nos of processors | PARFES | PARDISO | BSMFM |
|---|---|---|---|
| 1 | 654 | 596 | 1406 |
| 2 | 337.8 | 305.3 | 1015 |
| 3 | 232.1 | 208.6 | 869 |
| 4 | 177.9 | 163.3 | 793 |
| 5 | 145.7 | 135.7 | 786 |
| 6 | 125.6 | 116 | 777 |
| 7 | 110.6 | 100.9 | 772 |
| 8 | 100.5 | 90.83 | 807 |
| 9 | 92.5 | 83.85 | 770 |
| 10 | 86.3 | 79.6 | 796 |
| 11 | 82.1 | 77.36 | 825 |
| 12 | 87.5 | 78.45 | 839 |

PARFES is a good alternative to multi-frontal solver on multiprocessor shared-memory computers.

## References

[1] Amestoy P.R., Duff I.S., L'Excellent J.-Y., Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Meth. Appl. Mech. Eng., 184, pp. 501–520, 2000.

[2] George A., Liu J. W.-H. Computer solution of sparse positive definite systems. New Jersey : Prentice-Hall, Inc. Englewood Cliffs, 1981.

[3] Gould N. I. M., Hu Y., Scott J. A. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. Technical report RAL-TR-2005-005, Rutherford Appleton Laboratory, 2005.

[4] Fialko S. The block substructure multifrontal method for solution of large finite element equation sets. Technical Transactions, 1-NP, issue 8, 175 – 188, 2009.

[5] Fialko S. The direct methods for solution of the linear equation sets in contemporary FEA software; SCAD SOFT, Moscow, 2009, 160 p. (in Russian).

[6] Fialko S. PARFES: A method for solving finite element linear equations memory on multi-core computers. Advances in Engineering Software, 41, pp. 1256–1265, 2010.

[7] Schenk O., Gartner K. Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems. Parallel Computing 28, 187–197, 2002.