Metoda elementów skończonych dla symulacji zadań mechaniki konstrukcji o dużym (ponad kilka milionów równań) rozmiarze na zwykłych PC i laptopach

Sergiy Fialko

Institute of Computer Modelling, Cracow University of Technology

Kraków, Poland

sfialko@poczta.onet.pl

Preface:

- **More and more high-dimensionality problems of mechanics of solids and structures become solvable on individual desktop computers without involving expensive workstations, clusters, networking etc.**

- **This architecture requires a specific development of FEA software because methods used in distributed memory systems are often not the most efficient on desktop computers because of a restricted capacity of the core memory and a narrow bandwidth of the memory system.**

- ***The discussion will be confined to finite element solvers for problems in mechanics of solids and structures, implemented in software for individual desktop multi-core computers.***

Direct methods

$$\mathbf{Ax} = \mathbf{b} \quad \rightarrow \quad \mathbf{A} = \mathbf{L} \cdot \mathbf{D} \cdot \mathbf{L}^T$$

$$\mathbf{Ly} = \mathbf{b} \quad \rightarrow \quad \mathbf{y}$$

$$\mathbf{Dz} = \mathbf{y} \quad \rightarrow \quad \mathbf{z}$$

$$\mathbf{L}^T \mathbf{x} = \mathbf{z} \quad \rightarrow \quad \mathbf{x}$$

1. **Skyline**
2. **Frontal**
3. **Domain decomposition**
4. **Sparse direct solvers from libraries of high performance**
5. **Multi-frontal [1, 2]**
6. **PARDISO [8], PARFES [3]**

Motivation of PARFES:

- **Good scalability and high performance of PARDISO solver from Intel MKL library**

- **Poor scalability of multi-frontal methods on the shared-memory computers due to lot of data transfers from one memory area to another**

- **Solvers from well-known high performance libraries are not able to use the HD memory – only respectively small problems is possible to solve on desktop computers**

Objectives:

- Creation of the high-performance parallel finite element solver, which:

  a. Has a good speed-up in core mode

  b. Uses a disk memory when the dimension of problem exceeds of the core memory storage (virtualization property)

Outline:

- **$Kx = b \rightarrow K = L \cdot S \cdot L^T$, where $K = K^T$**

- **Decomposition of the sparse global finite element matrix on to dense rectangular matrix blocks and application of the level BLAS 3 routines from Itel MKL**

- **Parallelization scheme**

- **Virtualization**

- **Numerical results and its comparison with multi-frontal solver and PARDISO**

Sparse direct solvers for FEA software: key stages

- Reordering for reduction of fill-inns

- Subdivision of sparse matrix to dense rectangular blocks – a key moment for achievement of high performance (matrix-matrix multiplication procedure instead of vector-scalar ones)

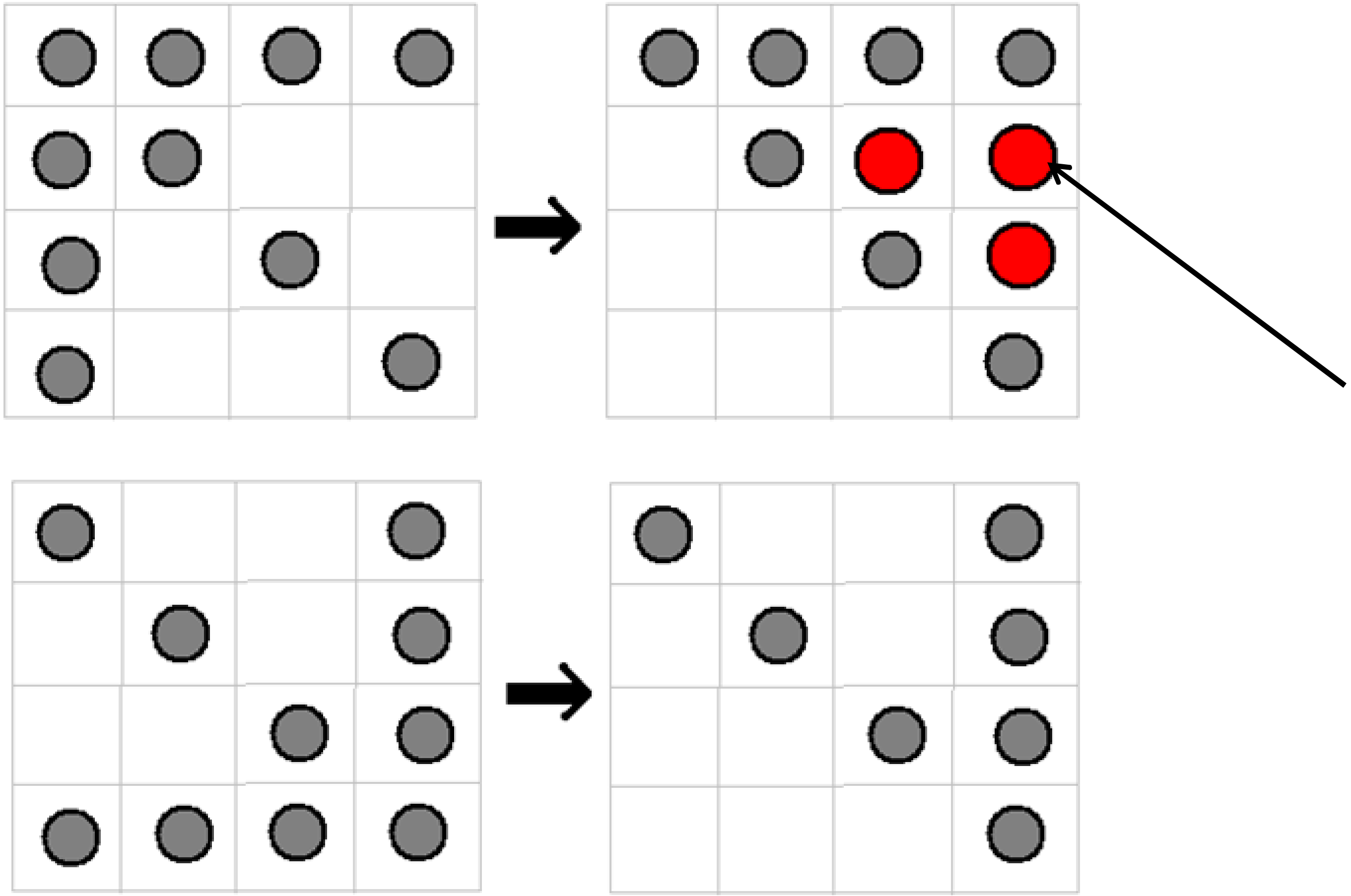
- Speed-up with increasing of processor numbers

- Virtualization when dimension of problem exceeds of core memory capacity

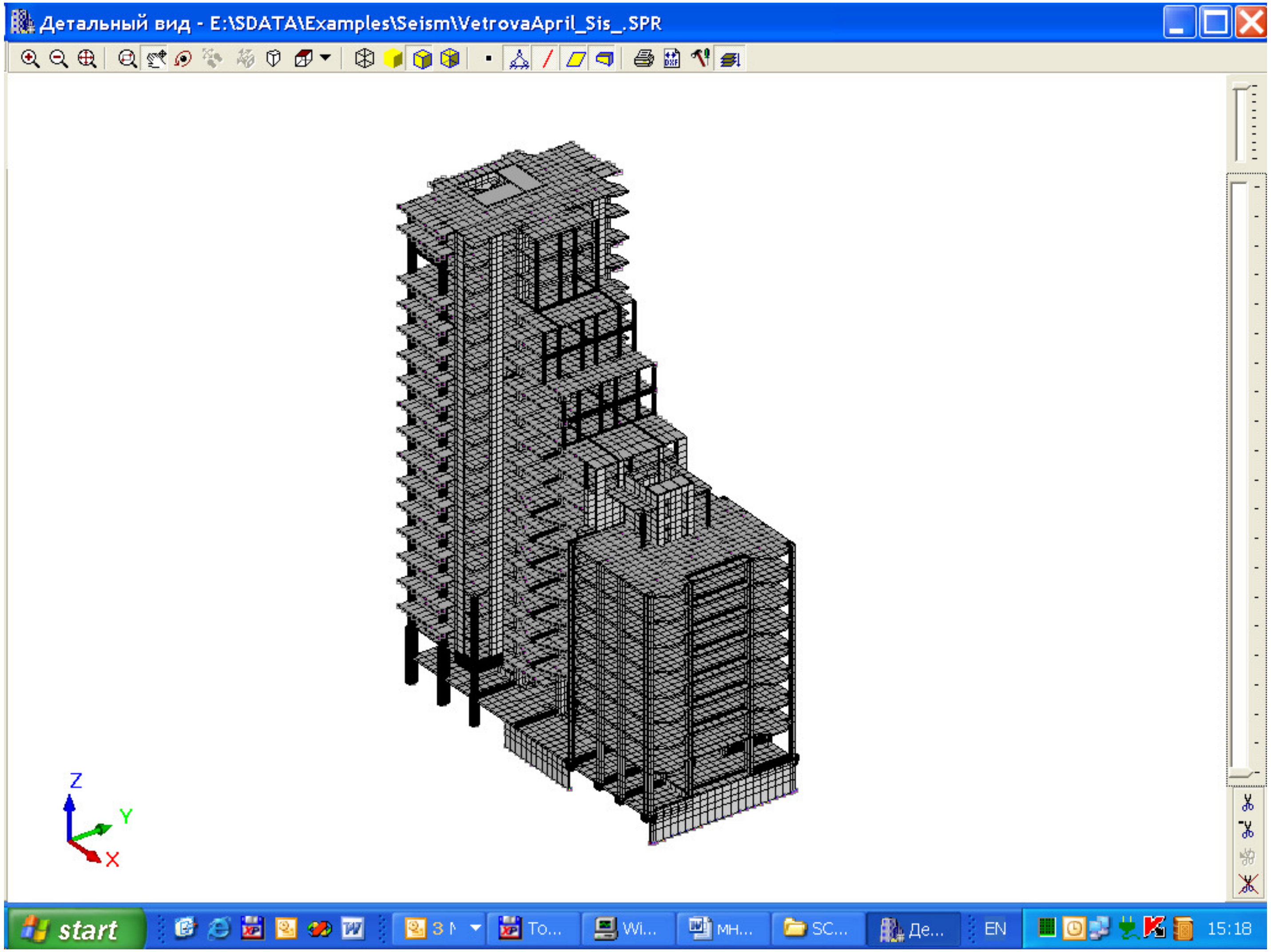
- Controlling of singularity and precision

Reordering for reduction of fill-inns

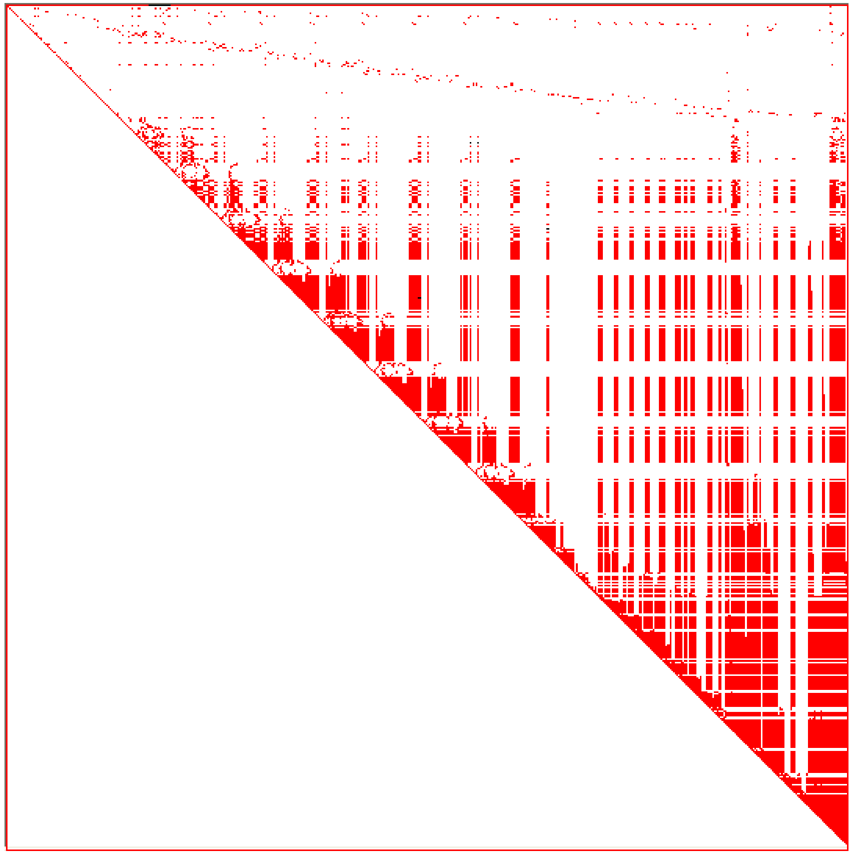
For sparse matrices the number of nonzero entries after factoring essentially depends on order of elimination of equations - reordering



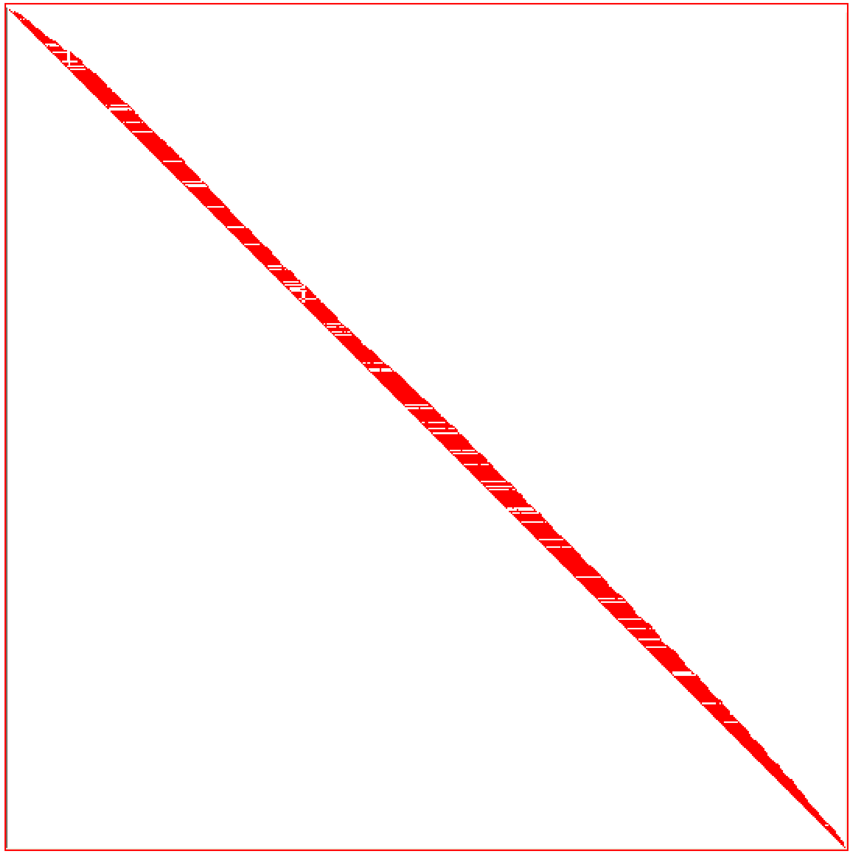
Fill-in – nonzero entry in factored matrix which arises on position of zero entry of source matrix


Детальный вид - E:\SDATA\Examples\Seism\VetrovaApril_Sis_.SPR
Z
Y
X
start
3 N
То...
Wi...
мн...
SC...
Де...
EN
15:18

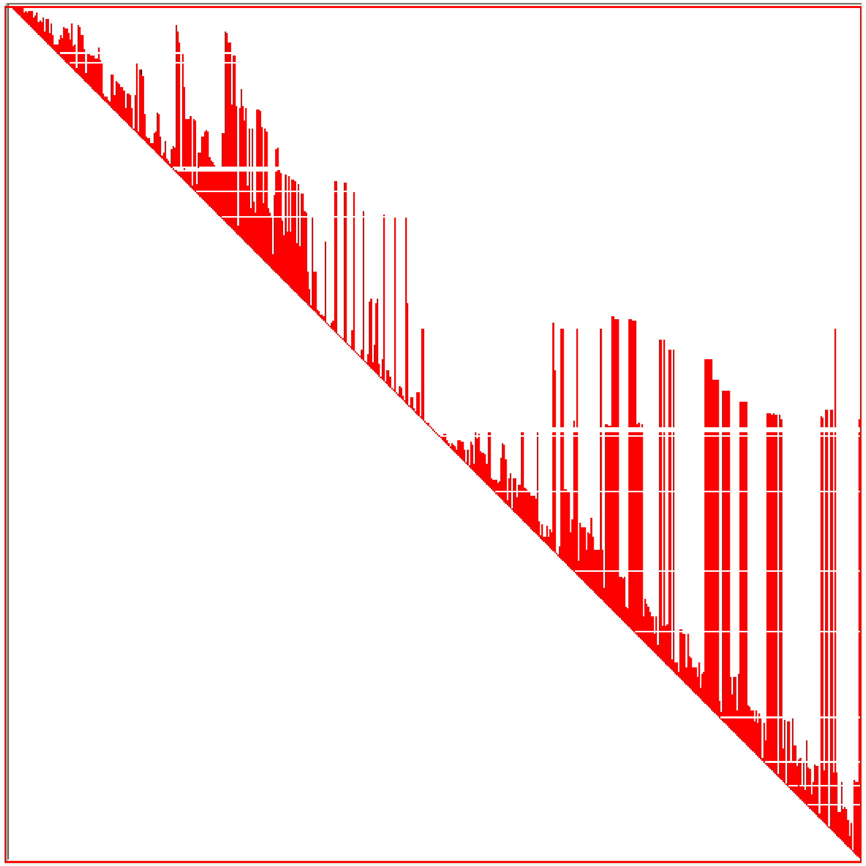
Real FE model from collection of SCAD Soft (www.scadsoft.com): 19 409 nodes, 19 456 finite elements and 115 362 equations



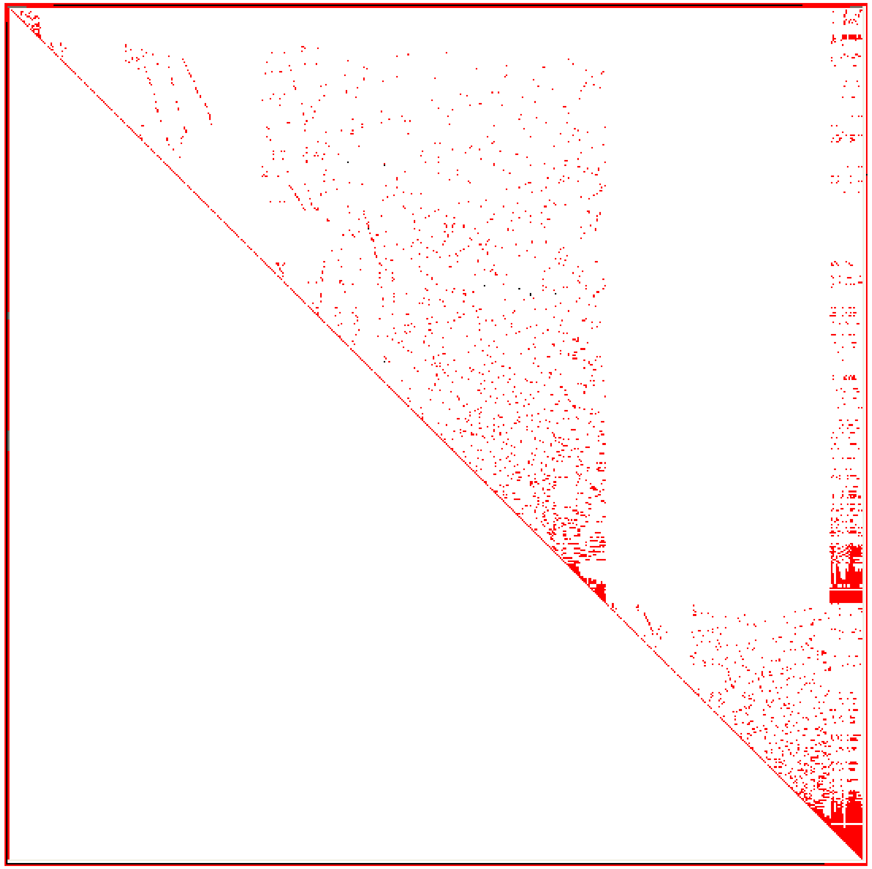
No reordering - 3 741 Mb



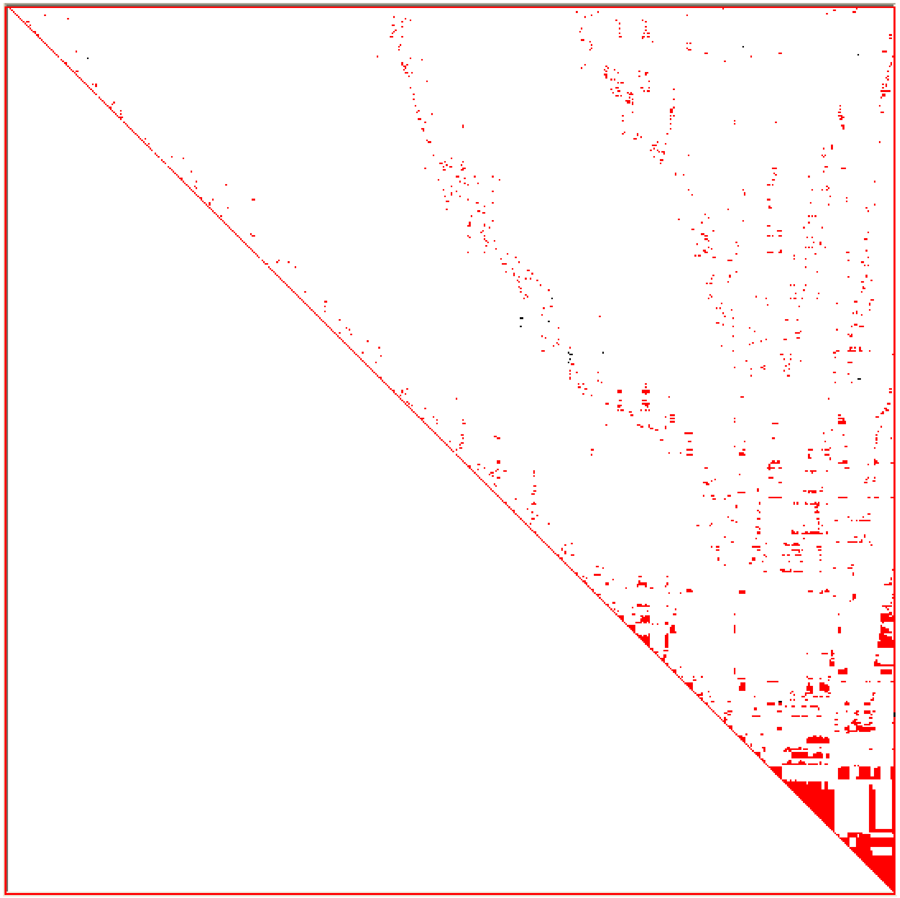
RCM - 1 618 Mb



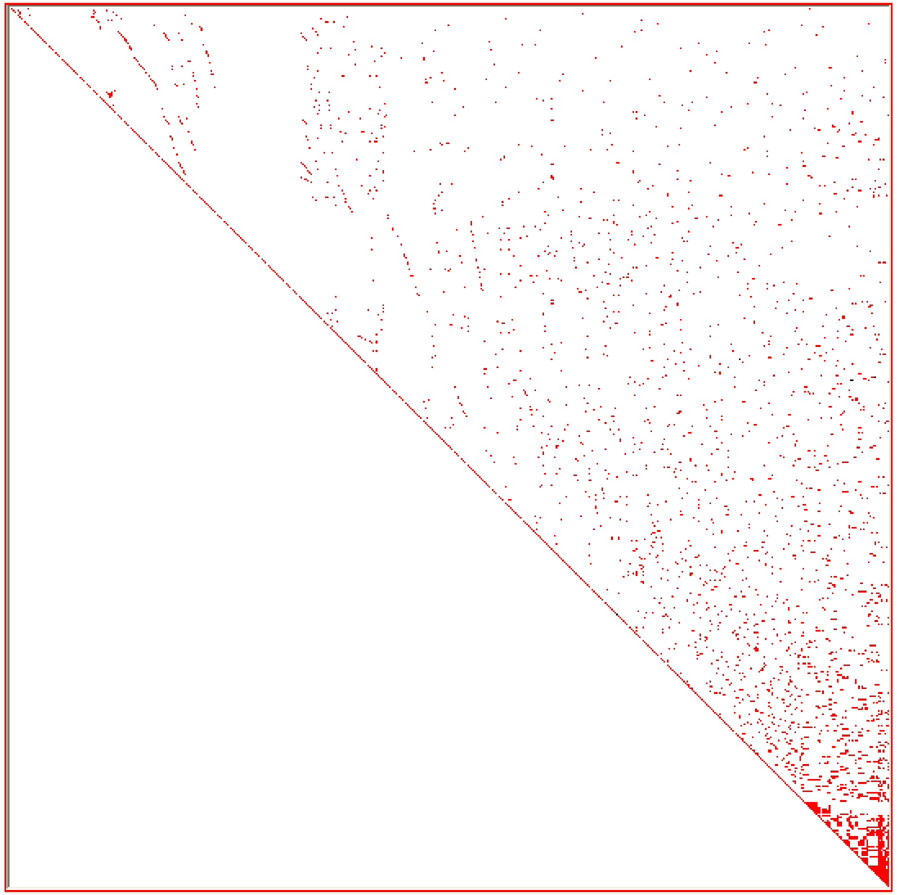
Sloan - 1 386 Mb



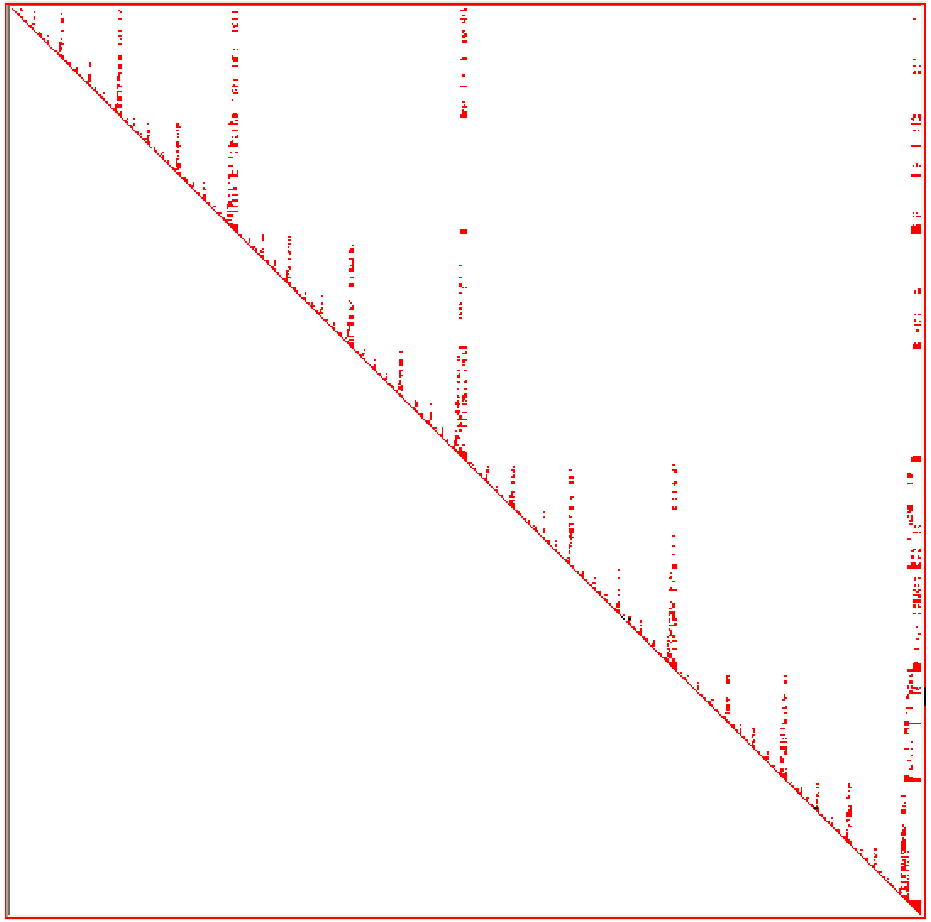
PSM+MMD - 345 Mb



ND - 644 Mb



QMD, MMD – 209 (191) Mb



Multilevel Reordering– 193 Mb

Reordering Method	Nonzero entries in factorized matrix	Size of factorized matrix, Mb
No reordering	490 366 701	3 741
RCM	212 143 113	1 618
Sloan	181 750 005	1 386
PSM+MMD	45 281 385	345
ND	84 522 753	644
QMD	27 501 777	209
MMD	25 142 373	191
Multilevel Reordering	25 341 381	193

- The computational cost of finding of optimal ordering is not less than the factoring cost of non-ordered matrix. Therefore, in practice, using heuristic algorithms.

- The result is that :

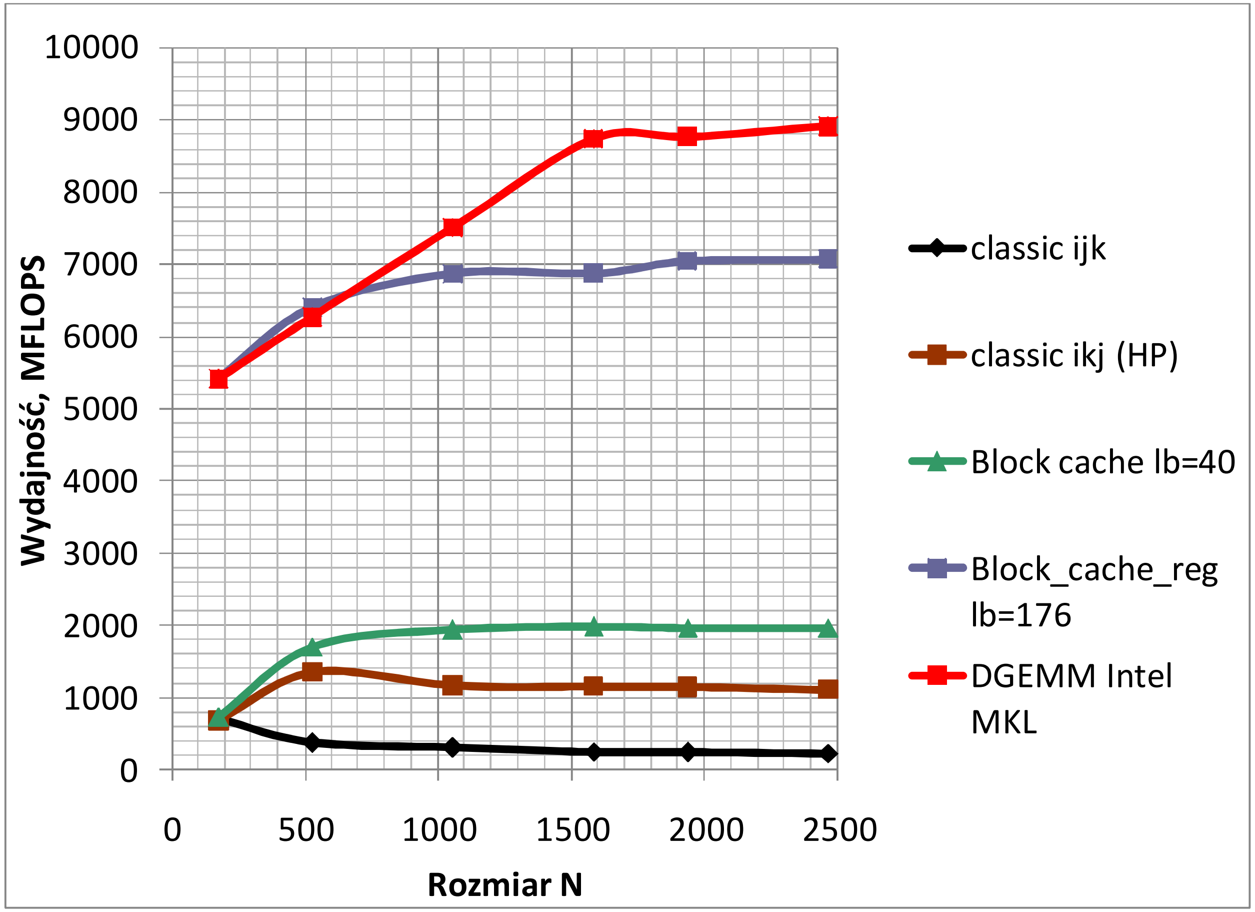
  - exists the many kinds of such algorithms

  - none of them does not lead to the optimal solution, but for better or worse approximation to it

  - for given problem is not known in advance which of the algorithms leads to the smallest number of nonzero entries

- Fast symbolic factorization algorithm, which works on adjacency graph of sparse matrix, allows one try the several algorithms during the few seconds and select the most proper one.

Subdivision of sparse matrix to dense rectangular blocks – a key moment for achievement of high performance. (the matrix-matrix multiplication procedure instead of matrix-vector or vector-scalar ones is applied)



Matrix multiplication: C = C+A·B .
Comparison of performancefor several algorithms.

Procesor – Intel® Core™2 Quad CPU Q6600 @2.40 GHz
Cache - L1: 32 KB,
L2:4096 KB
Pamięć: DDR2 800 MHz 8 GB

Classic ijk:
One ops – one transaction – processor performs mainly the empty tics.

Intel MKL:
the using of fast memory hides the slow memory system – processor runs on top of performance.

Example: a plane frame

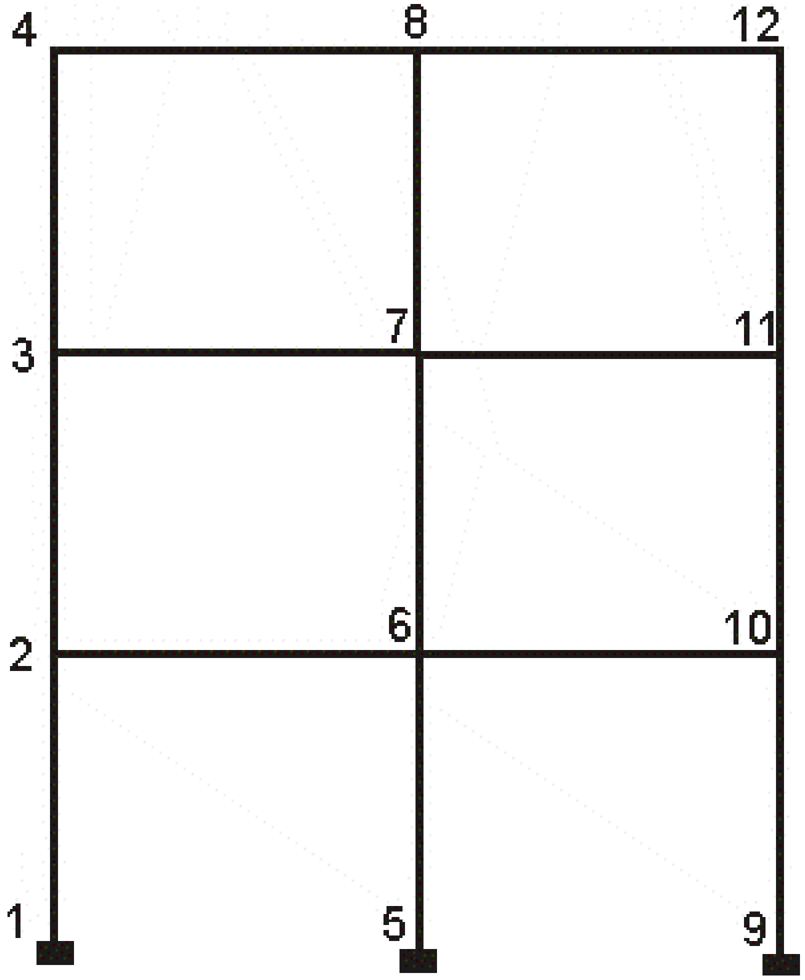


Fig. 1. The plane frame

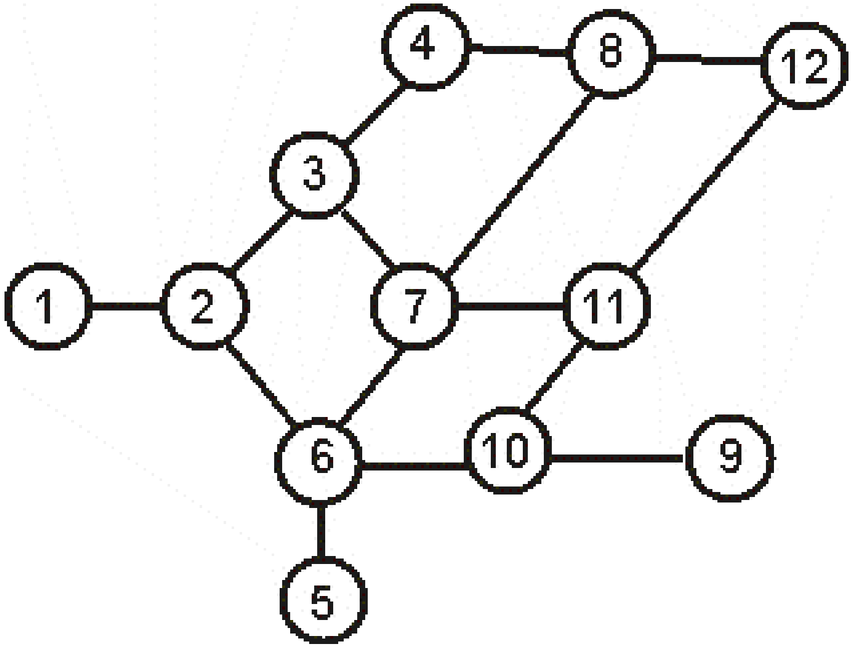


Fig. 2. The nodal adjacency graph before ordering

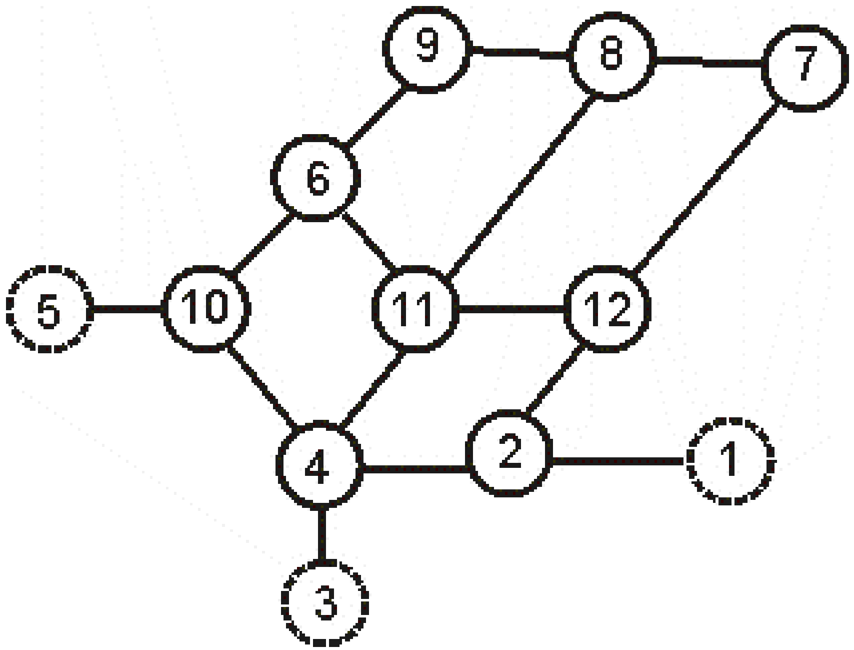


Fig. 3. The nodal adjacency graph after reordering. The supported nodes are marked by dash line

$$\begin{pmatrix}
2 &&&&&&&& \\
x & 4 &&&&&&& \\
&& 6 &&&&&& \\
&&& 7 &&&&& \\
&&& x & 8 &&&& \\
&& x && x & 9 &&& \\
& x & x &&& x & 10 && \\
& x & x && x & x & x & 11 & \\
x & x && x & x & x & x & x & 12
\end{pmatrix}$$

Fig. 4. The sparse matrix after reordering and symbolic factorization. “x” means nonzero entry in factorized matrix

- The main task is to subdivide the sparse matrix on rectangular blocks without essential increasing of non-zero entries.
- The super-nodal technique is applied for it.

Fig. 5. Elimination tree and super-nodal elimination tree

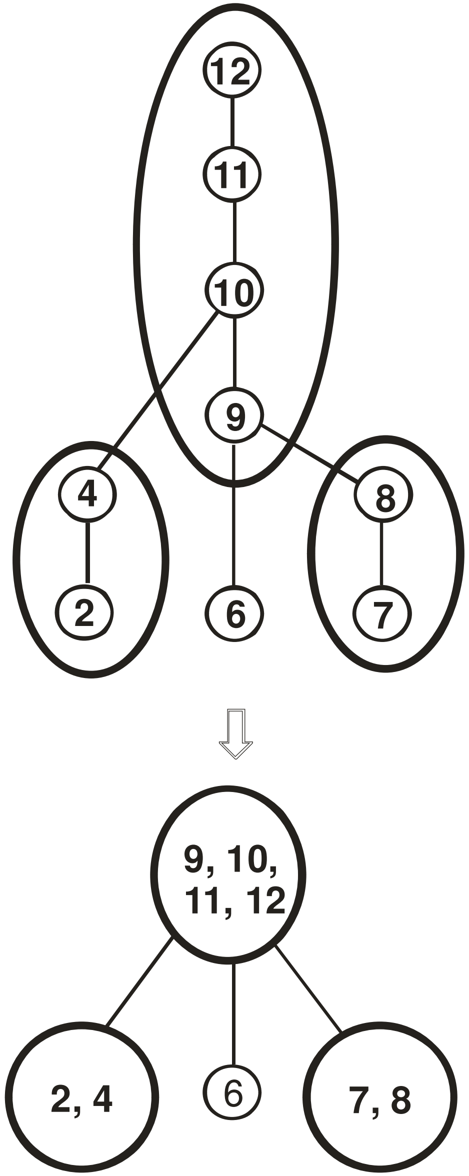
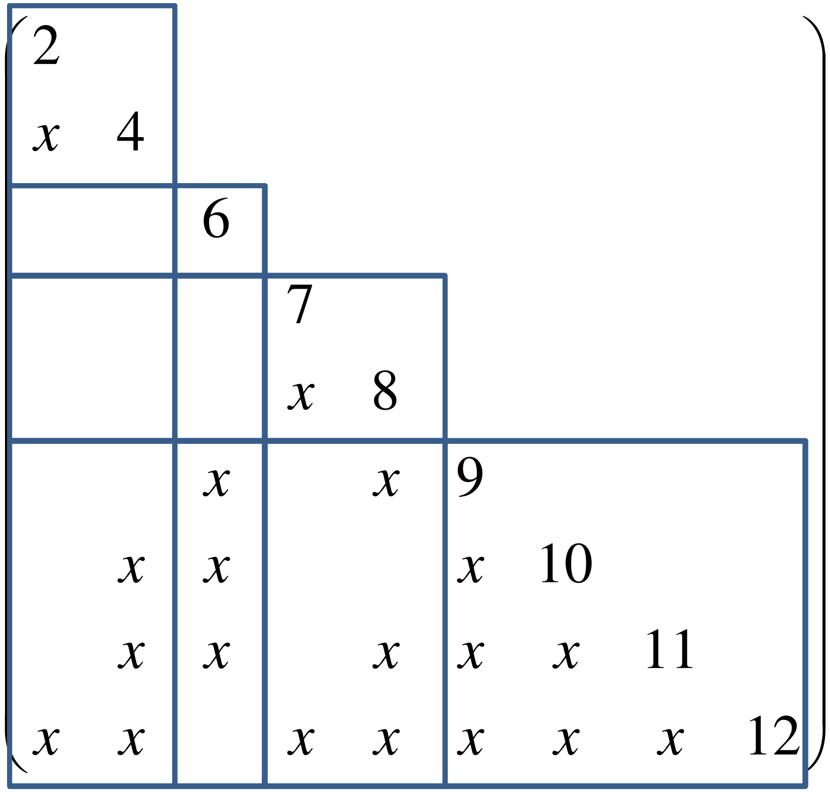
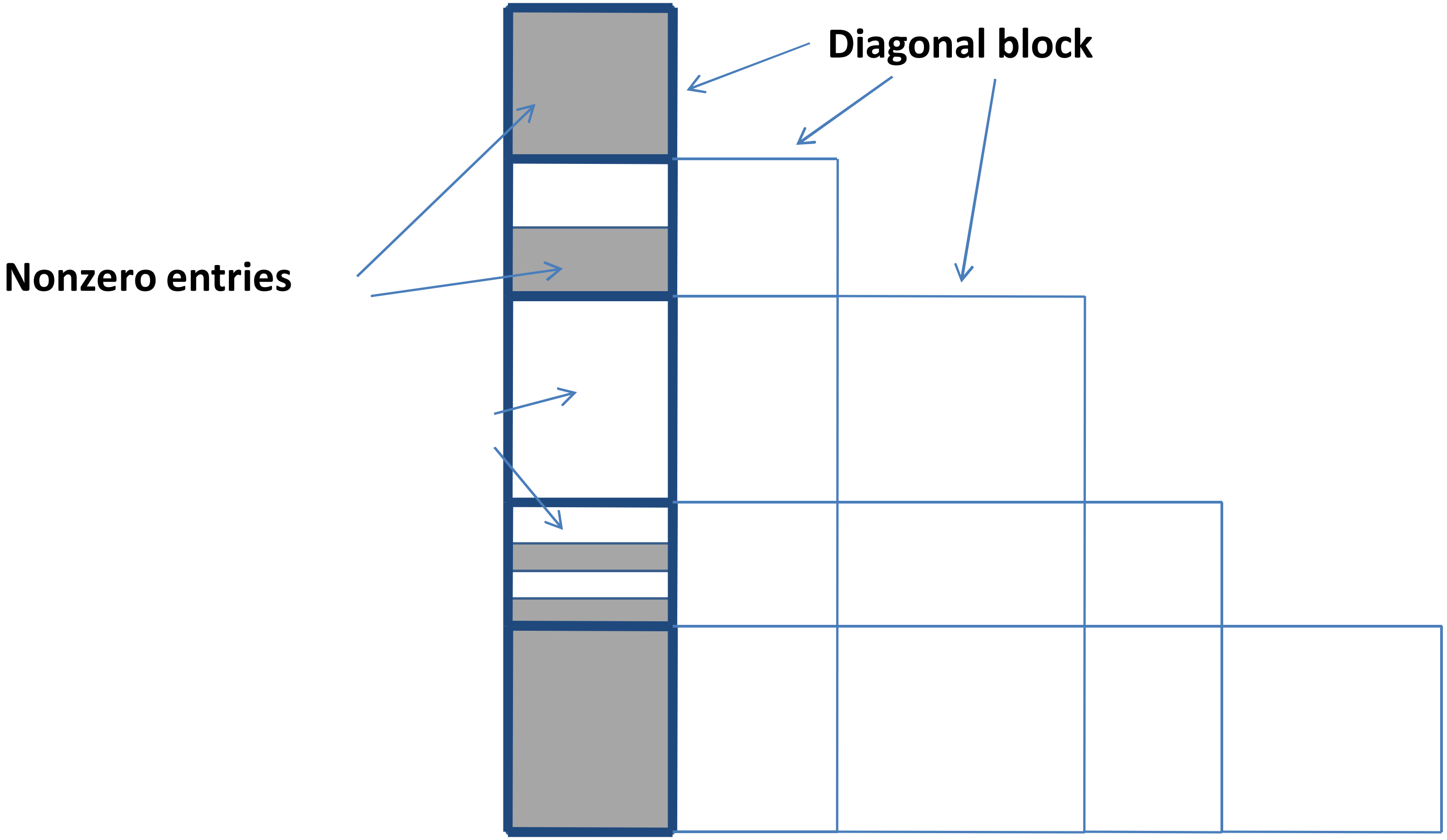


Fig. 6. The subdivision of sparse matrix on rectangular dense submatrices. Each diagonal block presents a supernode.



$$\left(\begin{array}{cc|c|cc|cccc}
2 & & & & & & & & \\
x & 4 & & & & & & & \\
\hline
 & & 6 & & & & & & \\
\hline
 & & & 7 & & & & & \\
 & & & x & 8 & & & & \\
\hline
 & & x & & x & 9 & & & \\
 & x & x & & & x & 10 & & \\
 & x & x & & x & x & x & 11 & \\
x & x & & x & x & x & x & x & 12
\end{array}\right)$$

- **Preparation of special data structures for storage of rectangular dense sub-matrices**



Typical structure of block column

Decomposition of the sparse global finite element matrix on to dense rectangular matrix blocks

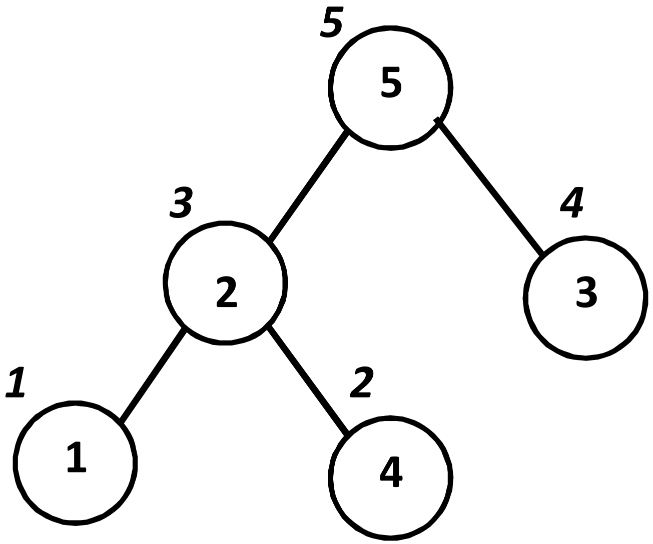
- Each node of the FE model contains the several equations which produce a dense submatrix - the natural grouping of equations occurs.

- The topological characteristics of the design model is used rather than structure of sparse global matrix.

- The reordering procedure is applied to reduce the fill-inns. The nodal adjacency graph is analyzed for it.

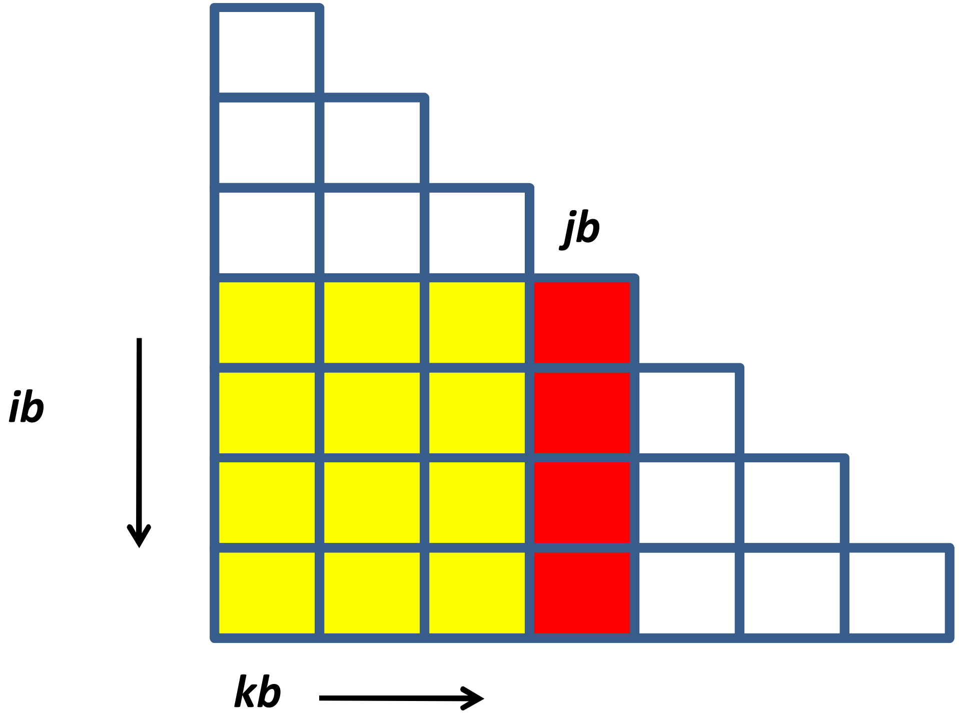
- The symbolical factorization, based on theorem by D.J.Rose, is produced to obtain a sparse specimen of factor matrix L .

- Creates an elimination tree and produces its renumbering



- Permutes the block columns in sparse matrix according with new numbering

Sparse decomposition algorithm: (looking left [3])



jb – the block column, which is factored on this step. This column is updated by columns, are located at the left.

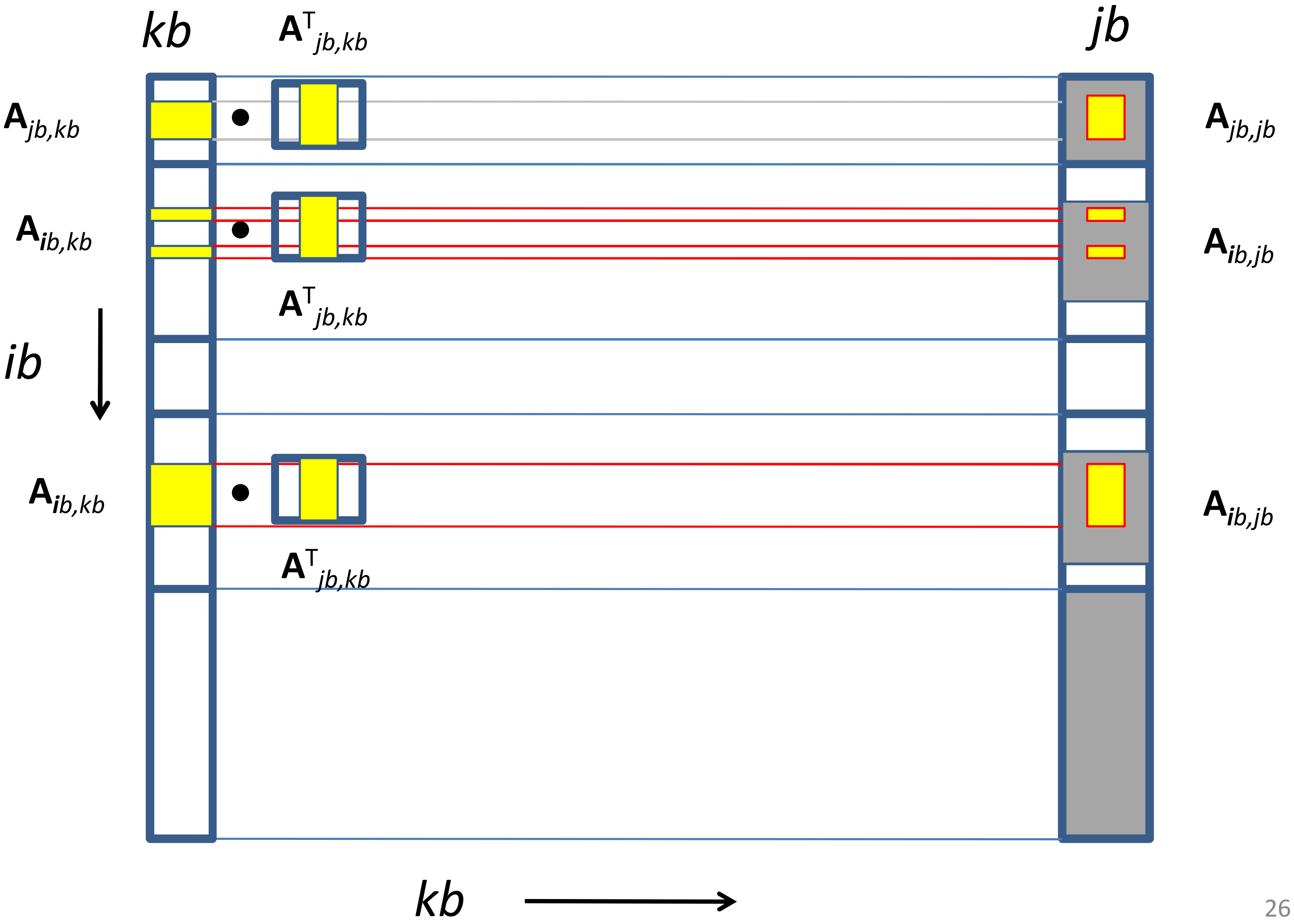
Sparse decomposition algorithm: (looking left)

```
1.  if(core mode)
        prepare block-columns $jb \in [1, N_b]$
2.  do $jb = 1, N_b$
3.      if(OOC ∨ 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 $ib \geq jb, ib \in L$:

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

Sparse decomposition algorithm: (looking left)

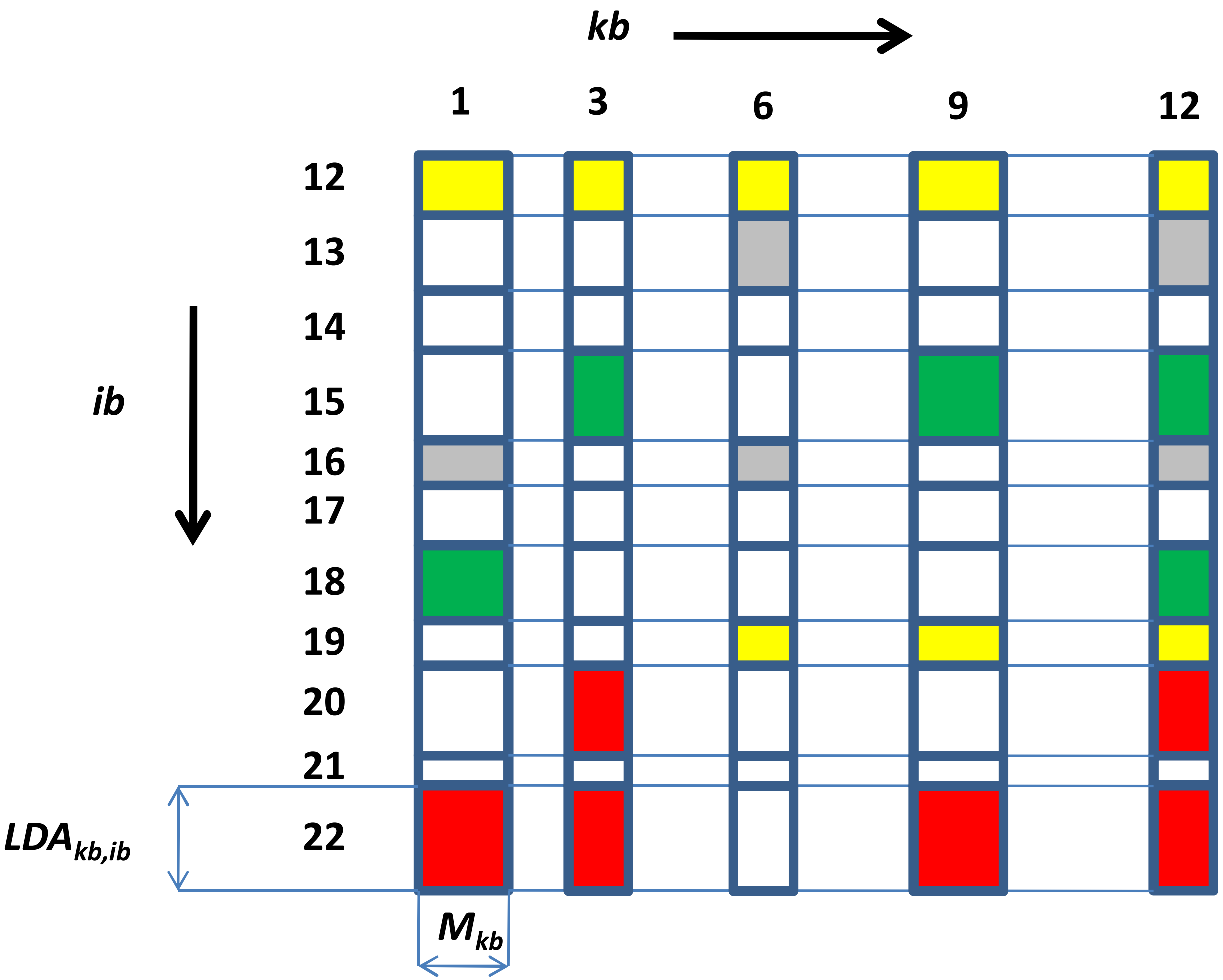
```
          if(OOC ∨ OOC1)
                write block-column jb to disk and free RAM
                for block-row ib = jb.

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

7.    end do.
```

➢ Update of block column *jb*:



➢ Mapping of blocks $A_{ib,kb}$ on to processors [3]:

$$\sum_{kb} W_{ib} = \sum_{kb} M_{kb} \cdot LDA_{kb,ib}$$



ip = 2; $\sum W_{12}$

ip = 3; $\sum W_{13}$

ip = 1; $\sum W_{15}$

ip = 3; $\sum W_{16}$

ip = 1; $\sum W_{18}$

ip = 2; $\sum W_{19}$

ip = 0; $\sum W_{20}$

ip = 0; $\sum W_{22}$

Sort: $W_{22} > W_{15} > W_{12} > W_{13} > W_{20} > W_{18} > W_{19} > W_{16}$

➢ Mapping of blocks A$_{ib,kb}$ on to processors [3] :

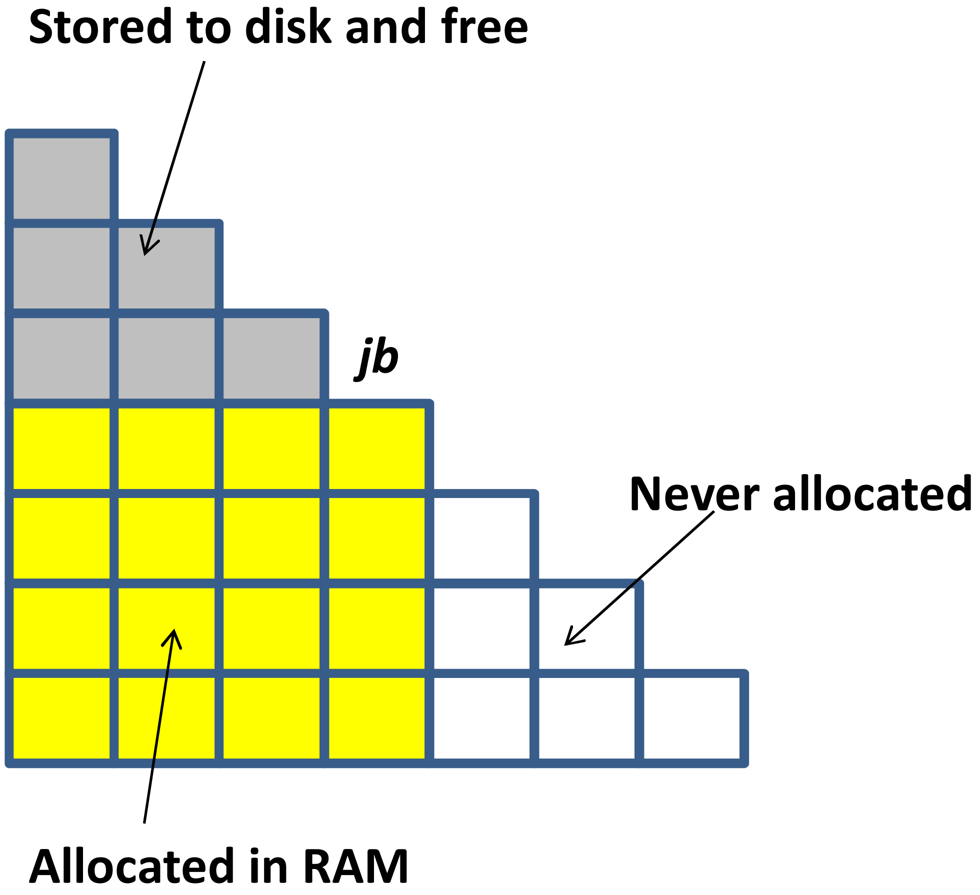
- **Defines the sum of weights:** $\sum_{kb} W_{ib} = \sum_{kb} M_{kb} \cdot LDA_{ib,kb}$
- **Descending sort of sum of weights**
- $\forall ip \in [0, 1, \ldots, ProcNumb-1] \quad sumofweights[ip] = 0$
- $\forall ib \in L_{jb}$ ***find min_ip*** **ϵ [0, 1, …, ProcNumb-1] |**
  (*sumofweights[min_ip]* is minimal)
- ***sumofweights[min_ip] +=*** $\sum_{kb} W_{ib}$ ***; thread_numb[ib] = min_ip***
- ***end loop over ib***
- $\forall kb \in Link[jb]$ ***(loop over kb)***
- $\forall ib \in L_{kb}$ ***(loop over ib)***
- ***Q[thread_numb[ib]] ←*** **(A$_{ib,kb}$; A$_{jb,kb}$; *kb*)** ***(put to queues Q[ip])***
- ***end loops over ib, kb***

➢ Parallel update of block column *jb* [3] :

```
○  # pragma omp parallel  ( ip ϵ [0, ProcNumb-1])
○      while(Q[ip] is not empty)
○          $A_{ib,kb}$;  $A_{jb,kb}$ ← Q[ip]; Q[ip] ← (Q[ip] /($A_{ib,kb}$;  $A_{jb,kb}$ ; kb))
○          $\mathbf{A}_{ib,jb} = \mathbf{A}_{ib,jb} - \mathbf{A}_{ib,kb} \cdot \mathbf{S}_{kb} \cdot \mathbf{A}^{T}{}_{jb,kb};$
○      end while
○  end of parallel region
```

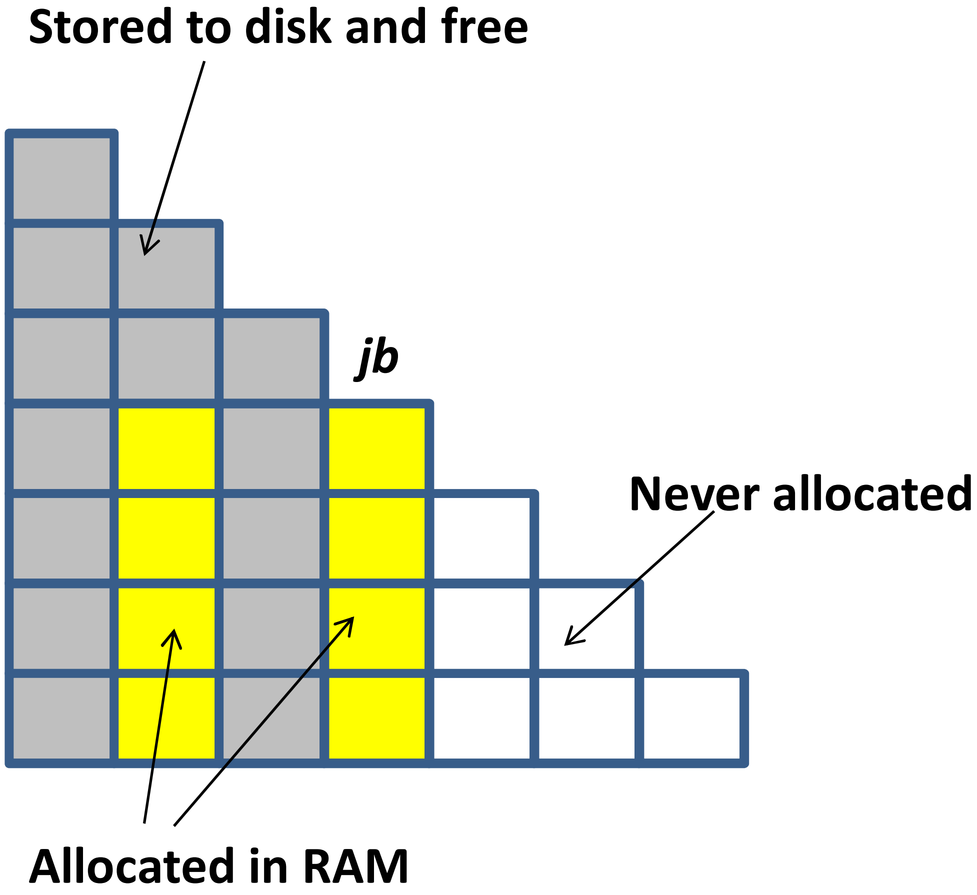
Virtualization

- OOC mode is turned on if the dimension of problem exceeds the core memory storage [3]



Virtualization

- **OOC1 mode is turned on if the dimension of problem exceeds the capability of OOC mode**



Numerical results

1. 4-core computer Intel® Core™2 Quad CPU Q6600 @2.40 GHz,
   cache L1 – 32 KB, L2 – 4096 KB,
   RAM: DDR2 800 MT/s, 8 GB core memory,
   Chipset: Intel P35/G33/G31,
   OS – Windows Vista^{TM} Business (64-bit), Service Pack 2

2. 4-core computer AMD Phenom™ II x4 995 3.2 GHz;
   L1: 4x64 KB L2: 4x512 KB L3: 6 MB;
   RAM: DDR3 1066 MT/s, 16 GB core memory,
   Chipset: AMD 790X,
   OS: Windows Vista^{TM} Business (64-bit), Service Pack 2

Numerical results

3. **Workstation DELL with two processors Intel Xeon X5660 @ 2.8 GHz /3.2 GHz (2×6 = 12 cores),**
   RAM DDR3, 24 GB core memory,
   OS – Windows 7 (64-bit)

4. **Notebook Toshiba Satellite:**
   Processor: Intel Pentium Dual CPU T3200 @ 2.00 GHz
   Cache: L1: 32 KB,   L2: 1024 KB
   RAM: DDR2 – 667 MT/s 4 GB
   Chipset: Intel GL40 rev. 07
   OS: Windows Vista^{TM} Business (64-bit), Service Pack 2

Numerical results

Table 1. Duration of numerical factorization (s) for a Cube 50x50x50 problem (397,941 equations), methods BSMFM and ANSYS v11.0 are used, a Core™2 Quad based computer

Method	Number of processors			Comments
	1	2	4	
BSMFM	827	504	365	ia32
ANSYS v11.0	1 610	882	544	ia32

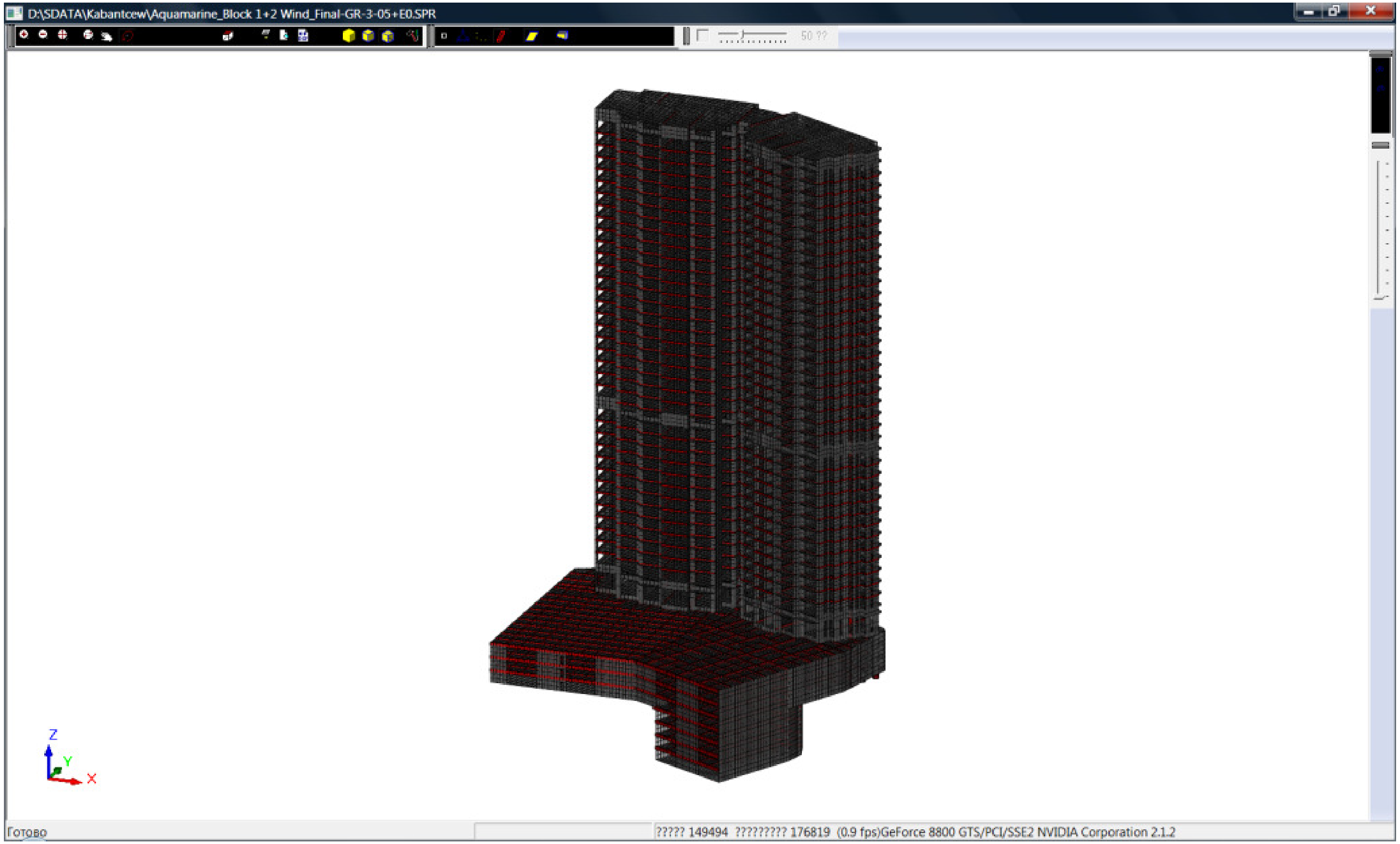
The performance of the BSMFM solver [4, 5] is at least as good as that of the multi-frontal method implemented in the well-known ANSYS software. Therefore the BSMFM method can be treated as a good implementation of the multi-frontal method which is quite usable in comparisons of this kind.

Numerical results

Multi-functional complex “Aquamarine” in Vladivostok

Numerical results

- **Real problems from computational practice of SCAD**



D:\SDATA\Kabantcew\Aquamarine_Block 1+2 Wind_Final-GR-3-05+E0.SPR

Готово

????? 149494 ????????? 176819 (0.9 fps)GeForce 8800 GTS/PCI/SSE2 NVIDIA Corporation 2.1.2

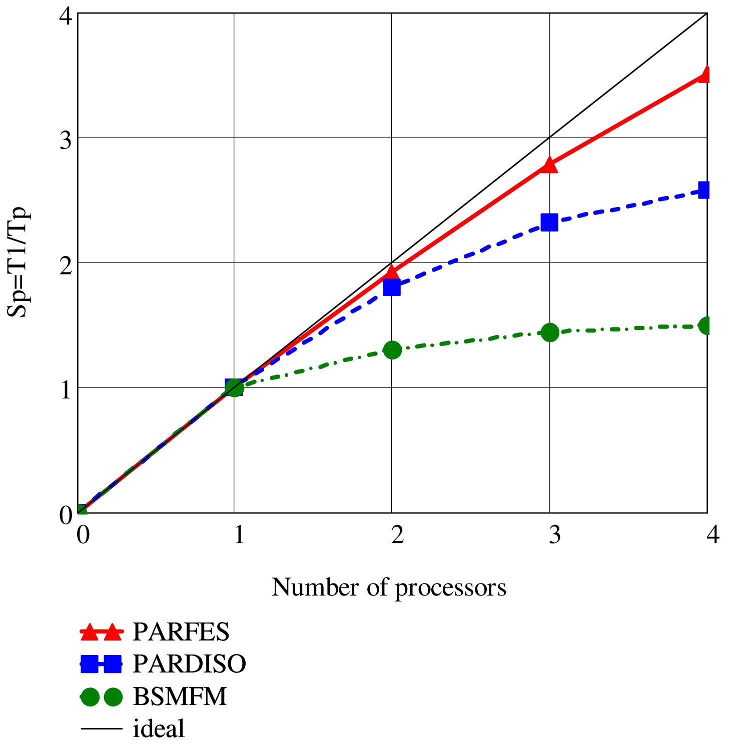
Aquamarine problem, 881 908 equations

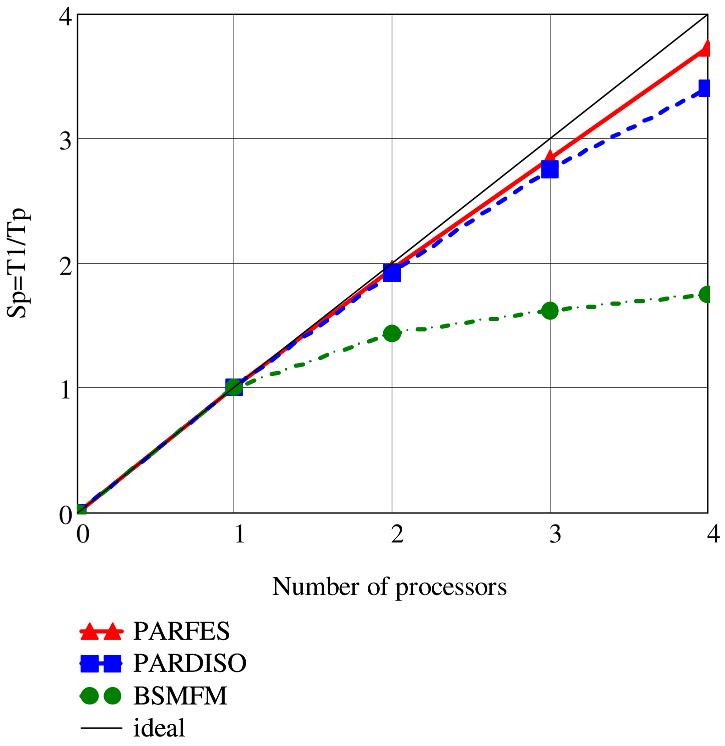
Numerical results

Table 2. Duration of numerical factorization (s) of the for the Aquamarine problem, 881 908 equations [3].

Method	NonZer (L), MB	Number of processors				Comments
		1	2	3	4	
PARFES (CM)	3 511	186	97	67	53	x64, Core™2 Quad
PARDISO(CM)	3 252	160	89	69	62	x64, Core™2 Quad
BSMFM (CM)	3 187	369	284	257	246	x64, Core™2 Quad
PARFES (CM)	3 511	139	71.9	49.5	38.6	x64, AMD Phenom™ II x4 995
PARDISO(CM)	3 187	135	70.6	49.2	39.9	x64, AMD Phenom™ II x4 995
BSMFM (CM)	3 187	291	203	180	166	x64, AMD Phenom™ II x4 995

Numerical results [3]

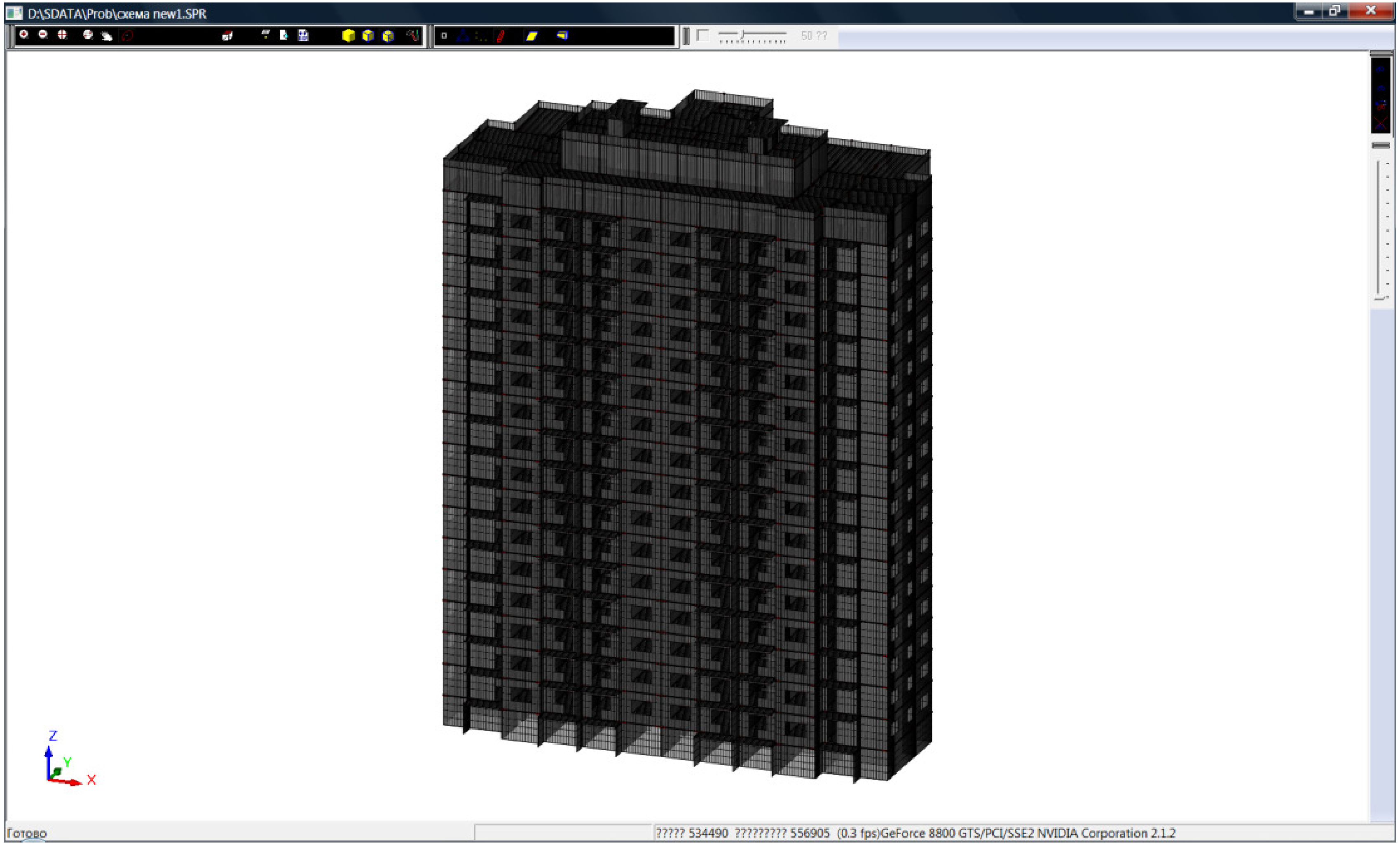




Aquamarine problem on a Core™2 Quad based computer (numerical factorization phase)

Aquamarine problem on an AMD Phenom™ II x4 995 based computer (numerical factorization phase)

Numerical results



D:\SDATA\Prob\схема new1.SPR

Готово

????? 534490 ????????? 556905 (0.3 fps)GeForce 8800 GTS/PCI/SSE2 NVIDIA Corporation 2.1.2

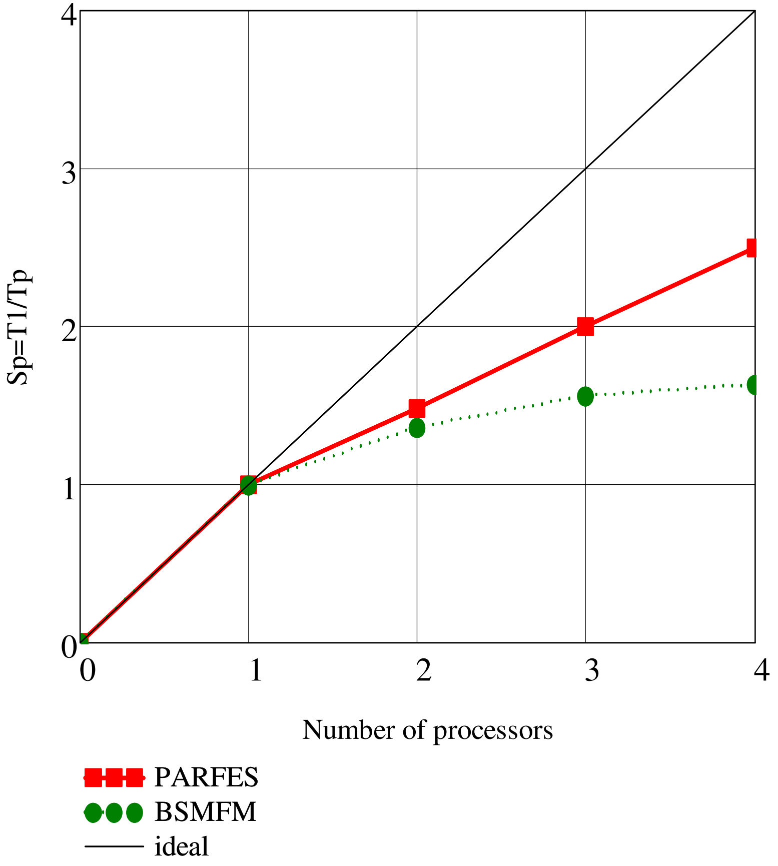
Problem schema_new_1, 3 198 609 equations

Numerical results

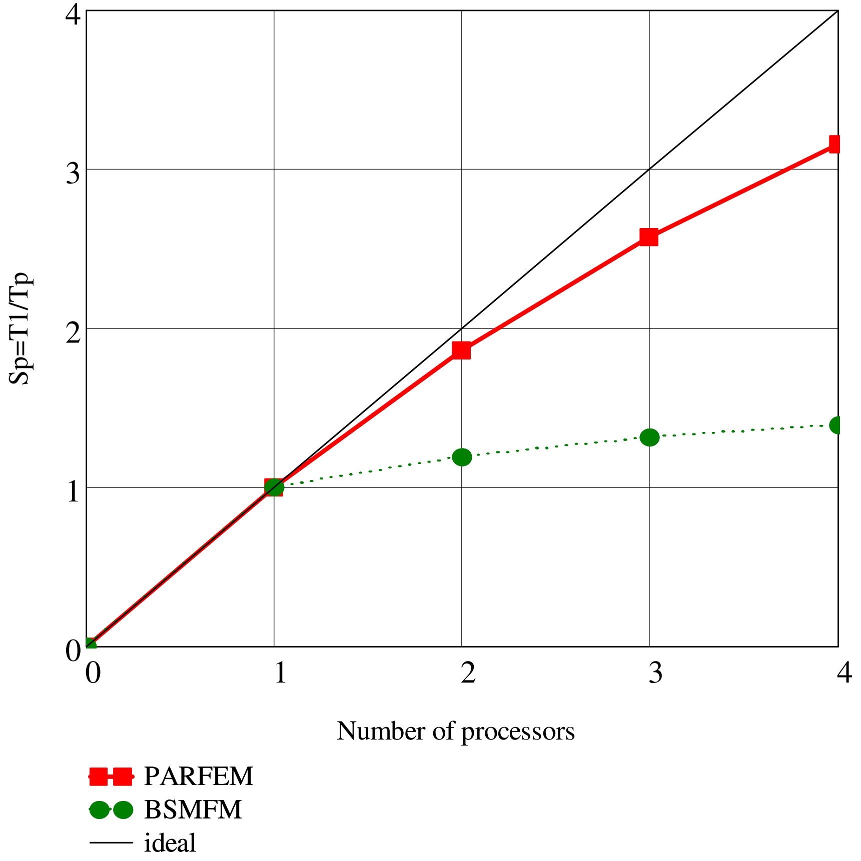
Table 3. Duration of the solution phases of the schema_new_1 problem (3,198,609 equations), a Core™2 Quad based computer [3]

Method	NonZer (L), MB	Analysis, s	Numerical factorization, s				Solution phase, s		Comments
			Number of processors				Number of proc.		
			1	2	3	4	1	4	
PARFES (OOC)	12 186	23.6	1 190	802	594	475	804	526	X64
PARDISO(OOC)	10 662	61.4	Numer. factorization phase: error = -11						X64
BSMFM (OOC)	10 869	9.0	2 011	1 482	1 286	1 232	497		x64

Numerical results
OOC mode [3]

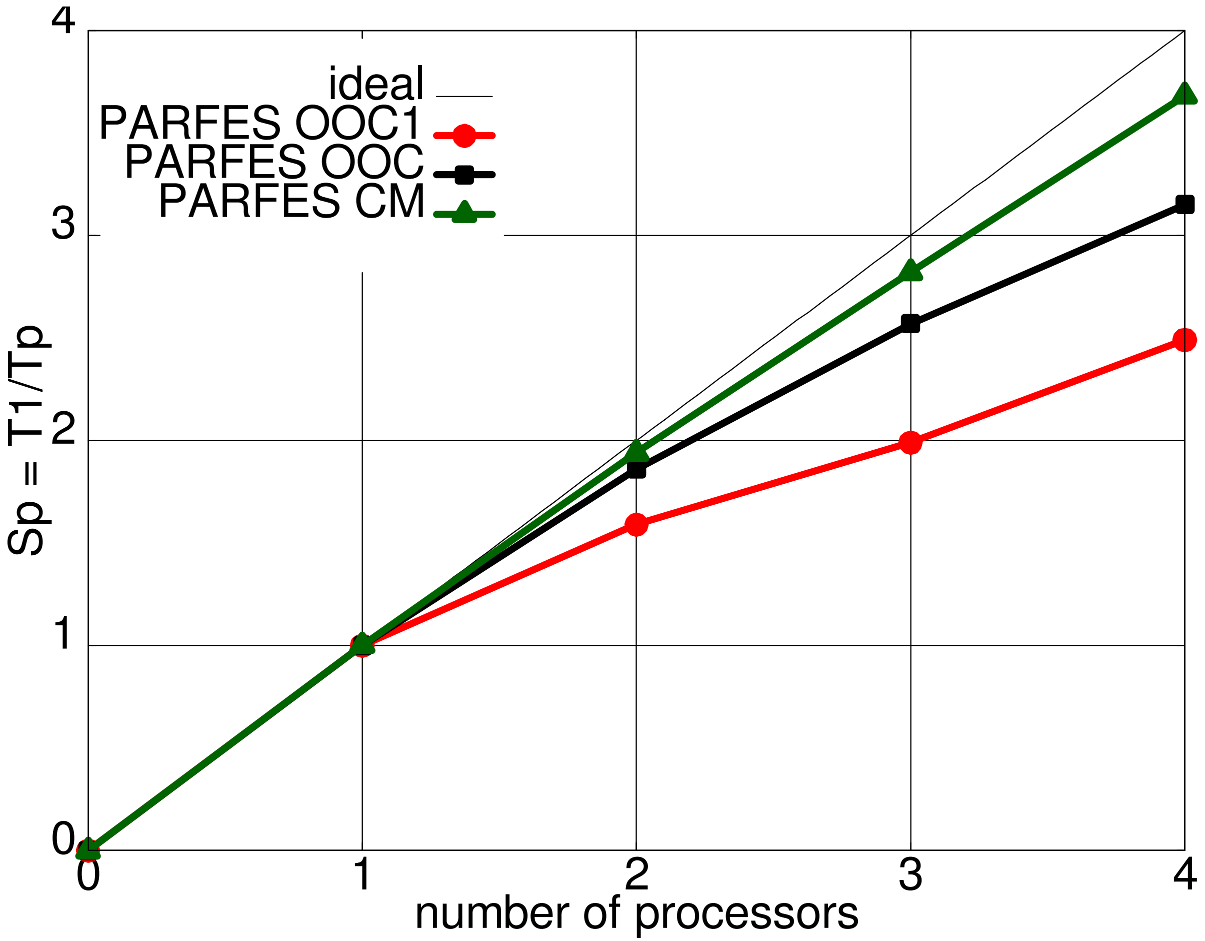


Schema_new_1 problem on a Core™2 Quad based computer (numerical factorization phase)



Schema_new_1 problem on an AMD Phenom™ II x4 995 based computer (numerical factorization phase), RAM restricted to 8 GB

Numerical results



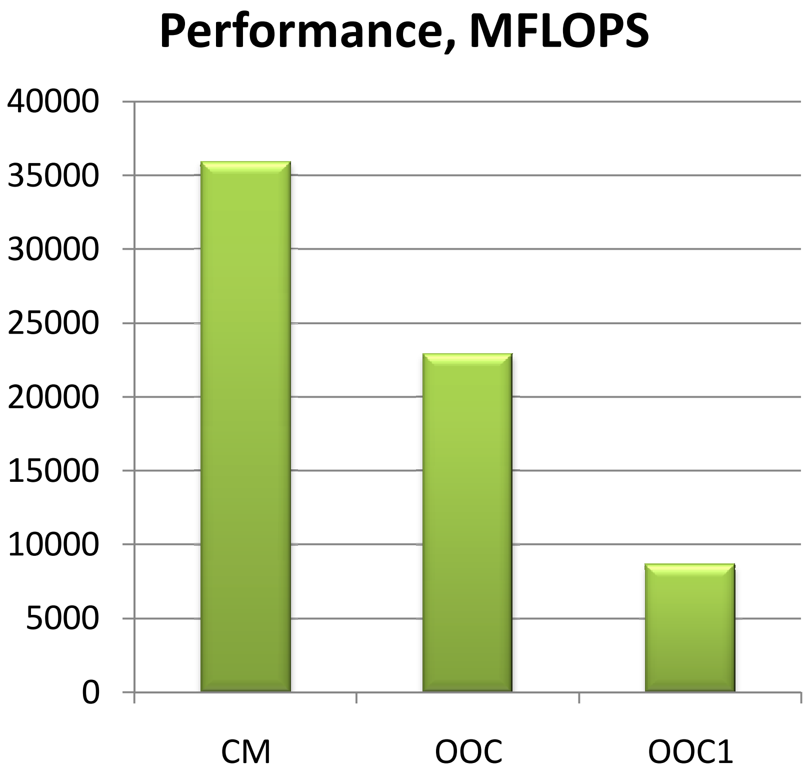
32 453 MFLOPS
12 186 MB RAM

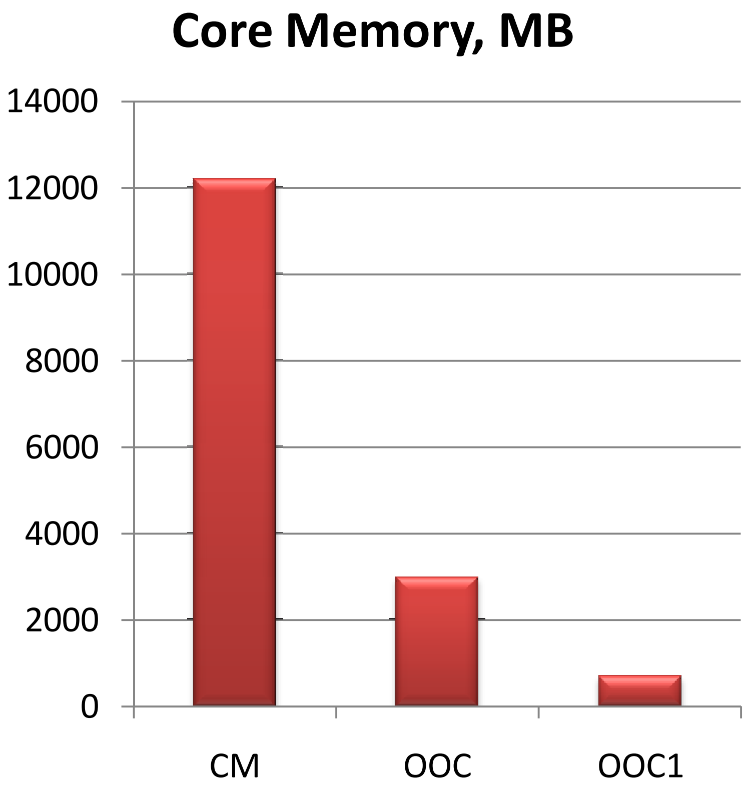
22 898 MFLOPS
2 995 MB RAM

8 610 MFLOPS
706 MB RAM

Schema_new_1 problem on an AMD Phenom™ II x4 995 based computer (numerical factorization phase), RAM - 16 GB

Numerical results





Schema_new_1 problem on an AMD Phenom™ II x4 995 based computer (numerical factorization phase), RAM - 16 GB

Numerical results

Table 4. Duration of factoring phase for problem schema_new_1 (3,198,609 equations), AMD Phenom™ II x4 995 based computer (numerical factorization phase), RAM - 16 GB

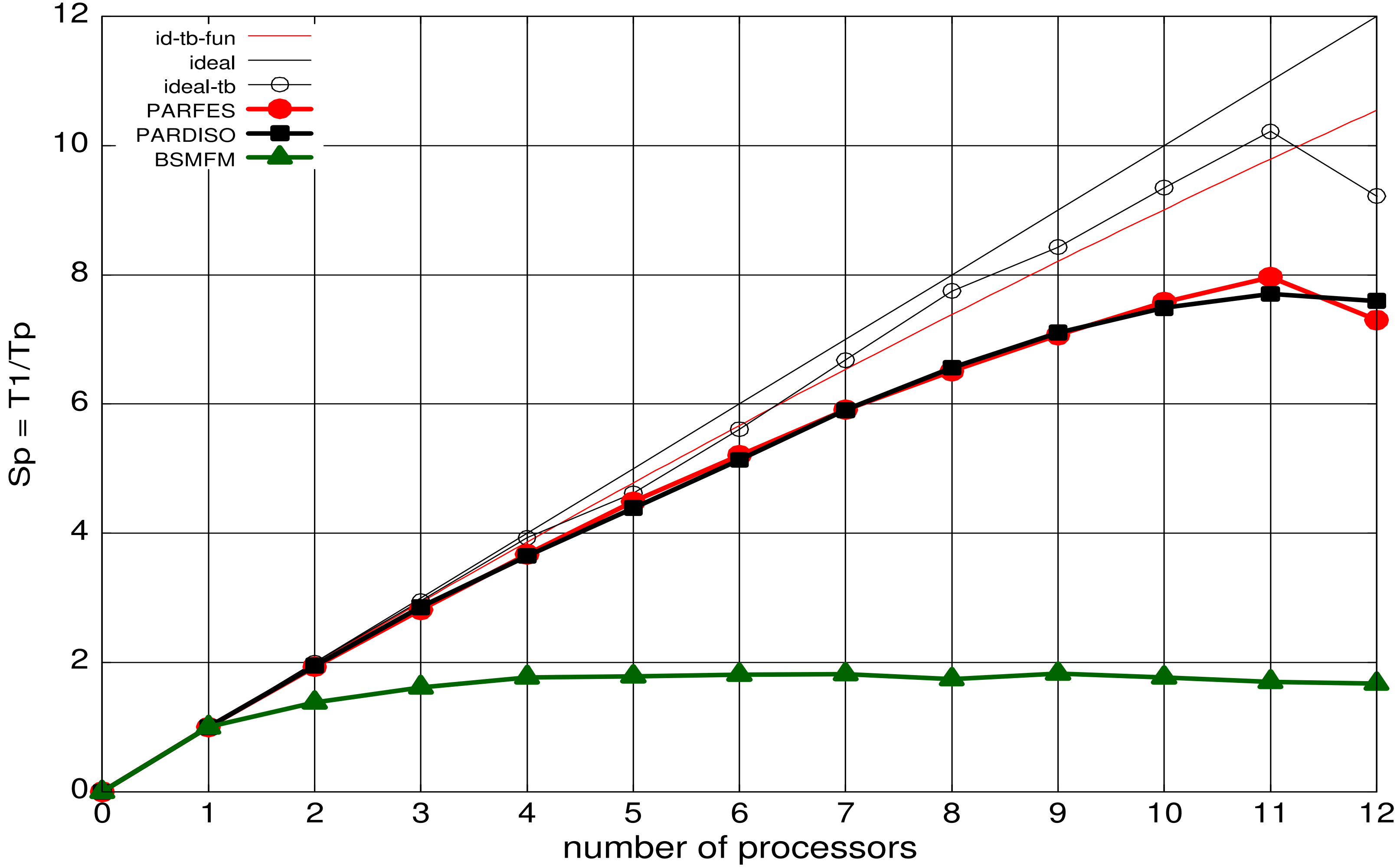
No's of proc.	PARFES			PARDISO		
	Numer. Fact, s	MFLOPS	$S_p = T_1/T_p$	Numer. Fact, s	MFLOPS	$S_p = T_1/T_p$
1	729	8 759	1	697	7 782	1
2	372	17 160	1.96	367.4	14 775	1.90
3	255	25 063	2.86	260	20 861	2.68
4	196.9	32 453	3.70	207.8	26 181	3.35

Numerical results

Табл. 5. Table 8. Duration of factoring phase for problem schema_new_1 (3,198,609 equations), workstation DELL with two processors Intel Xeon X5660 @ 2.8 GHz /3.2 GHz (12 cores), RAM 24 GB, DDR3, Core mode, platform ×64

No's of proc.	PARFES		PARDISO		BSMFM	
	Anal., s	Num. Fact., s	Anal., s	Num. Fact., s	Anal., s	Num. Fact., s
1	16.9	654	31.06	596	13	1406
2	16.9	337.8	23.59	305.3	13	1015
3	16.9	232.1	25.33	208.6	13	869
4	16.9	177.9	23.26	163.3	13	793
5	16.9	145.7	23.79	135.7	13	786
6	16.9	125.6	25.68	116	13	777
7	16.9	110.6	23.11	100.9	13	772
8	16.9	100.5	23.43	90.83	13	807
9	16.9	92.5	23.98	83.85	13	770
10	16.9	86.3	23.71	79.6	13	796
11	16.9	82.1	29.86	77.36	13	825
12	16.9	87.5	28.58	78.45	13	839

Numerical results



Schema_new_1 problem on an Intel Xeon X5660 @ 2.8 GHz based computer (numerical factorization phase) - CM

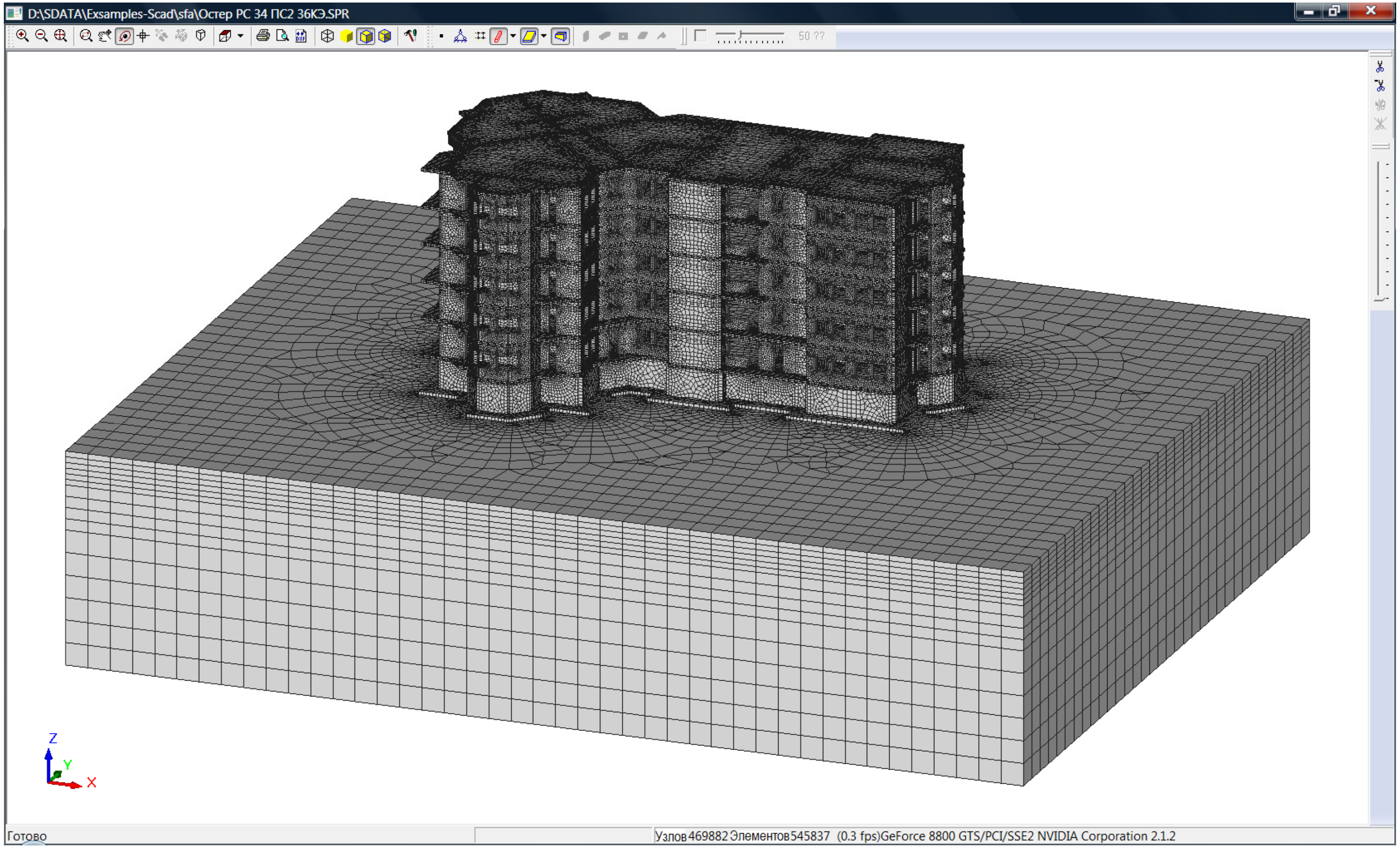
Numerical results

Notebook Toshiba Satellite:

Processor: Intel Pentium Dual CPU T3200 @ 2.00 GHz
Cache: L1: 32 KB, L2: 1024 KB
RAM: DDR2 – 667 MT/s 4 GB
Chipset: Intel GL40 rev. 07
OS – Windows Vista^{TM} Business (64-bit), Service Pack 2

Application ia32 OOC1 mode 2 threads
Analysis : 26 s
Assembling : 196 s = 3 ‘ 16”
Numerical factoring : 3 111 s = 51’ 51”
Forward/Back reduction: 1 194 s = 19‘ 54”
Total time : 4 532 s = 75’ 32”

Numerical results [3]



D:\SDATA\Exsamples-Scad\sfa\Остер PC 34 ПС2 36КЭ.SPR

Готово

Узлов 469882 Элементов 545837 (0.3 fps)GeForce 8800 GTS/PCI/SSE2 NVIDIA Corporation 2.1.2

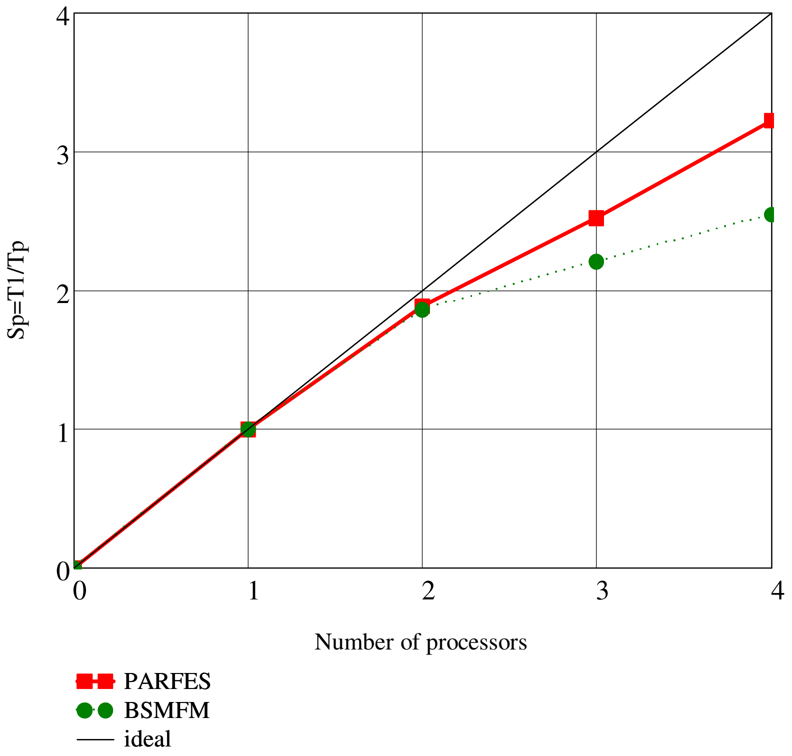
Problem Oster_PC_34_PC2, 2 763 181 equations

Numerical results

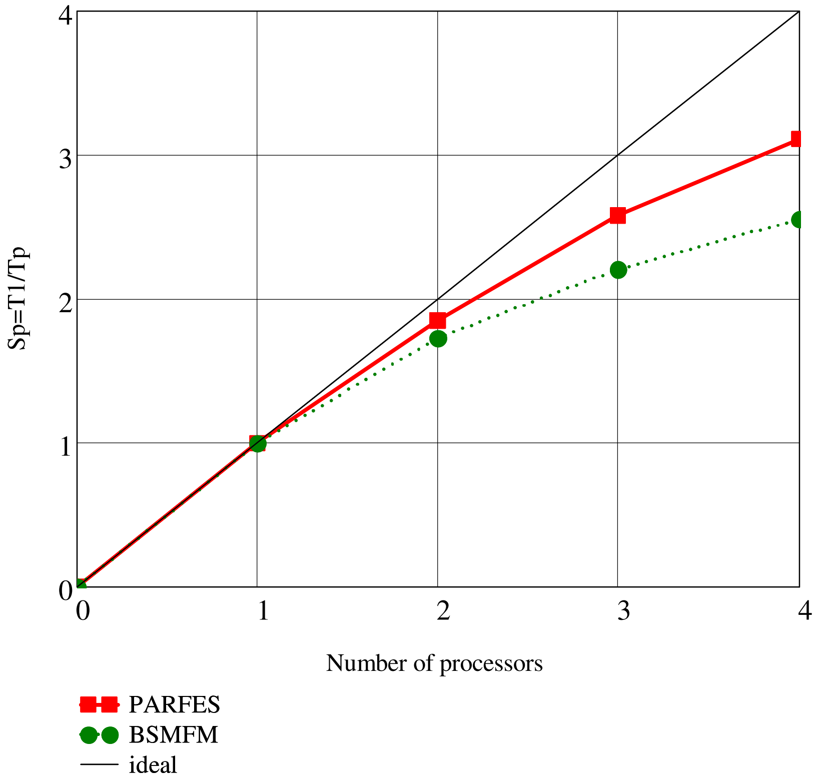
Table 6. Duration of the solution phase of Oster_PC_34_PC2 problem (2,763,181 equations), an AMD Phenom™ II x4 995 based computer [3]

Method	NonZer (L), MB	Analysis s	Numerical factorization, s				Solution phase, s		Comments
			Number of processors				Number of proc.		
			1	2	3	4	1	4	
PARFES (OOC)	15 761	49.1	1 649	891	640	530	592	510	x64
PARDISO(OOC)		Page Fault during analysis phase							x64
BSMFM	14 622	29	2 700	1 563	1 251	1 059	498		x64

Numerical results – OOC mode [3]



Oster_PC_34_PC2 problem on a Core™2 Quad based computer (numerical factorization phase)



Oster_PC_34_PC2 problem on an AMD Phenom™ II x4 995 based computer (numerical factorization phase)
RAM 8 GB

PART II. MODAL & SEISMIC ANALYSIS

A block Lanczos method with spectral transformations for natural vibrations and seismic analysis of large structures

The methods which are applied in modern FEA software most often are:

- Block subspace iteration (*E. Wilson*)
- Block Lanczos method (*Ericsson T., Ruhe A., Grimes R.G., Lewis J.G., Simon H.D., Golub G.H., Underwood R.R.*)

The Lanczos method: main idea

The eigenvalue problem is considered: $\mathbf{K}\varphi - \omega^2\mathbf{M}\varphi = 0$

Factorize the stiffness matrix: $\mathbf{K} = \mathbf{L} \cdot \mathbf{S} \cdot \mathbf{L}^T$

For arbitrary start vector q$_0$ (q$_0$ must have zero components for equations with zero rows in mass matrix) performs an iterative process:

Solve: $\mathbf{K}\hat{\mathbf{q}}_{j+1} = \mathbf{M}\mathbf{q}_j \Rightarrow \hat{\mathbf{q}}_{j+1} \quad \rightarrow \quad \hat{\mathbf{q}}_{j+1} = \mathbf{K}^{-1}\mathbf{M}\mathbf{q}_j; \quad j = 1, 2, \ldots.$

On each step $\hat{\mathbf{q}}_{j+1}$ **is orthogonalized to all previous obtained vectors** $\mathbf{q}_j, \mathbf{q}_{j-1}, \ldots, \mathbf{q}_1$. **In exact arithmetic is needed to orthogonalize explicitly only against** $\mathbf{q}_j, \mathbf{q}_{j-1}$.

So, the recursion is: $\tilde{\mathbf{q}}_{j+1} = \mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1}$

The Lanczos method: main idea

where $\alpha_j = \hat{\mathbf{q}}_{j+1}^T \mathbf{M}\mathbf{q}_j$ **and β**

where $\alpha_j = \hat{\mathbf{q}}_{j+1}^T \mathbf{M}\mathbf{q}_j$ **and β_j is taken from previous step.**

On step *j*+1 : $\beta_{j+1} = \sqrt{\tilde{\mathbf{q}}_{j+1}^T \mathbf{M}\tilde{\mathbf{q}}_{j+1}}$, $\mathbf{q}_{j+1} = \tilde{\mathbf{q}}_{j+1} / \beta_{j+1}$.

The given sequence of vectors creates a Krylov subspace and is a fine basis for Rayleigh–Ritz method. The source problem is presented:

$$\mathbf{K}_\sigma^{-1}\mathbf{M}\boldsymbol{\psi} - \theta\boldsymbol{\psi} = 0, \quad \theta = 1/\lambda$$

Application of Rayleigh–Ritz method leads to:

$$\mathbf{T}_j\mathbf{s}_j - \theta_j\mathbf{s}_j = 0, \quad \mathbf{T}_j = \mathbf{Q}_j^T\mathbf{K}_\sigma^{-1}\mathbf{M}\mathbf{Q}_j = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \alpha_3 & \beta_4 & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & \beta_j & \alpha_j \end{pmatrix}$$

The Lanczos method: main idea

and $\mathbf{Q}_j = \left\{ \mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_j \right\}$ are Lanczos vectors and

$$\mathbf{Y}_j = \mathbf{S}_j \mathbf{Q}_j \ , \quad \mathbf{S}_j = \left\{ \mathbf{s}_j^1, \mathbf{s}_j^2, ..., \mathbf{s}_j^j \right\}$$

are the Ritz vectors. The given algorithm is numerically stable until the first eigenpair is converged. The selective and partial orthogonalizations are introduced to ensure the numerical stability of Lanczos method.

The shifted block Lanczos method [9] is applied to increase a performance of classical one.

We solve: $$\mathbf{K}_{\sigma}\varphi - \omega_{\sigma}^{2}\mathbf{M}\varphi = 0\,,$$

where $$\mathbf{K}_{\sigma} = \mathbf{K} - \sigma\mathbf{M}; \quad \omega^{2} = \omega_{\sigma}^{2} + \sigma; \quad \sigma - \text{shift}$$

The block version of algorithm allows us to reduce the I/O operations during forward – back substitutions due to parallel implementation of the several (block) right-hand-sides (r.h.s.) instead of single r.h.s. It is very important for large problems: 60 000 – 1 500 000 degrees of freedom (DOFs) and more.

The spectral transformations: $\mathbf{K}_{\sigma}^{-1}\mathbf{M}\varphi = \lambda_{\sigma}\varphi, \quad \lambda_{\sigma} = \frac{1}{\omega^2 - \sigma}$

are implemented to split the long frequency interval into a few relatively short ones and reduce the drastic increase of Krylov subspace size, caused by large number of required eigenpairs.

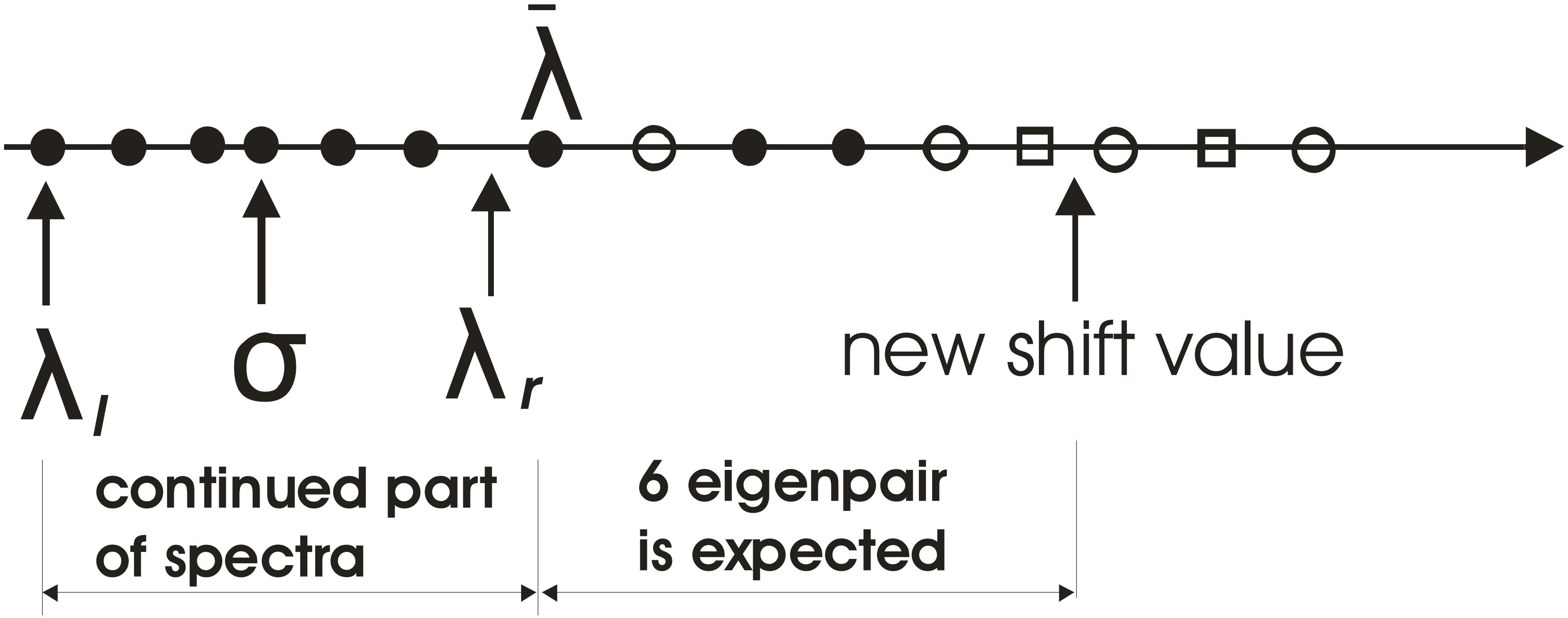
A trust interval [9]: $\lambda \in [\lambda_l, \lambda_r], \quad \lambda_r > \lambda_l$

- All eigenpairs are extracted with precision not worse than:

$$\left\|\mathbf{K}\varphi_i - \omega_i^2\mathbf{M}\varphi_i\right\|_2 / \left\|\omega_i^2\mathbf{M}\varphi_i\right\|_2 = prec \le tol = 10^{-6} \div 10^{-8}, \quad \lambda_i = \frac{1}{\omega_i^2}$$

- The skipped eigenpairs are missing in the trust interval

The extraction of large number eigenpairs consists of expanding of trust interval by means evaluating of relatively small subintervals. The choice of new shift value is based on prediction of the right part of the eigenspectrum:



● **Converged eigenpairs (prec < tol)**

○ **Ritz approximations (0.01 > prec > tol)**

□ **Coarse approximations (0.01 < prec)**

The modes of analysis

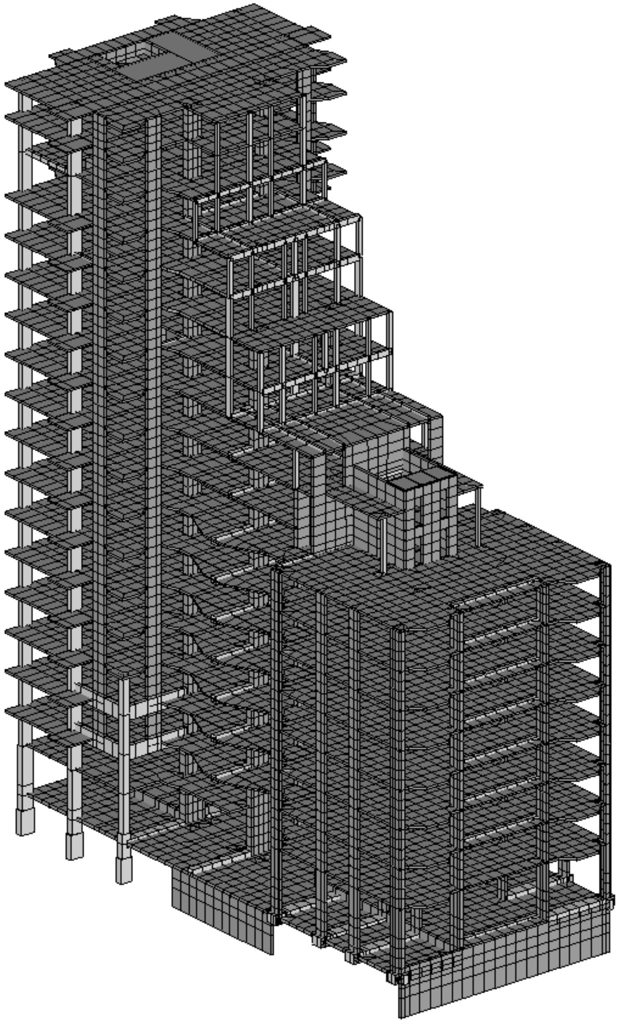
Well-known modes:

1. Modal mode – extraction of the required number of eigenpairs
2. Interval mode – extraction of all eigenpairs in frequency interval [a,b]

Specific modes:

3. Seismic mode [6] – extraction of eigenpairs so long as the Required sum of modal masses will be achieved in each seismic Input direction.
4. Verification mode [7] – allows us to detect hard-to-find errors of a finite element model, such as a local and global dimensional instability, lack of supports and so on.

The modal mode



Multi – storey building in Kiev. FEM model contains 19409 nodes, 19456 finite elements and 115 362 equations

Table 1: The efficiency of different methods

Number of eigen-pairs	Subspace iterations method	Block subspace iterations method	Lanczos method	Block Lanczos method with shifts
25	2 h 28 m 31 s	1 h 49 m 38 s	54 m 24 s	38 m 14 s
50	5 h 18 m 33 s	3 h 06 m 16 s	1 h 22 m 37 s	55 m 56 s
100	> 24 h	~12 h	2 h 22 m 14 s	1 h 52 m 14 s
1 000	----------	-----------	-----------	11 h 25 m 02 s

Computer: P-III CPU Intel 1000 MHz , RAM 512 MB
Precision of eigenpairs is not worse than 10^{-8}

Seismic mode [Fialko S.]

Mass participation factor: $\Gamma_i^{dir} = (\mathbf{M}\varphi_i, \mathrm{I}_{dir}), \quad i = 1,2,\ldots,n, \quad dir = OX, OY, OZ$

i **– mode number,** dir **– seismic input direction .**

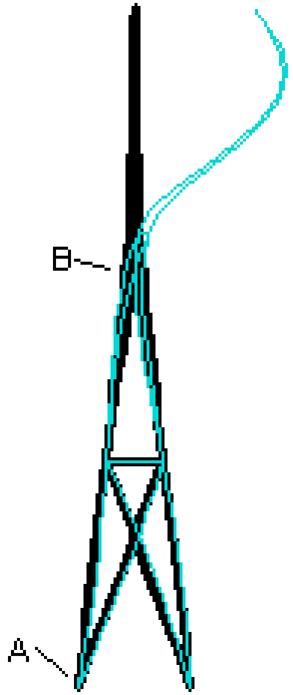
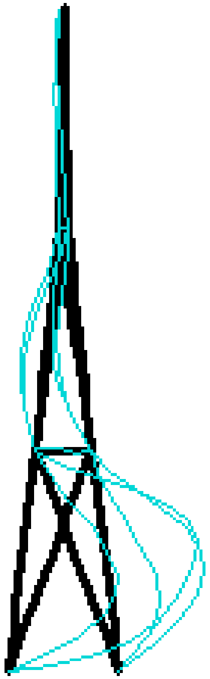
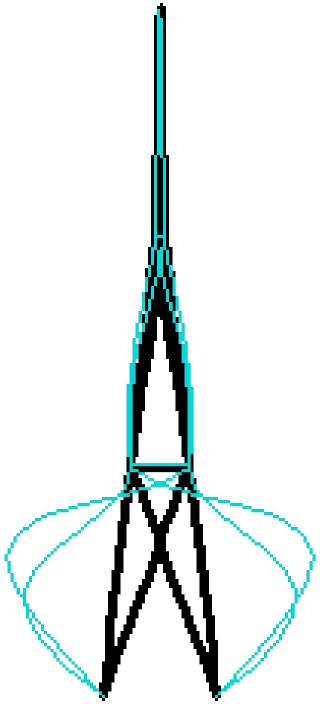
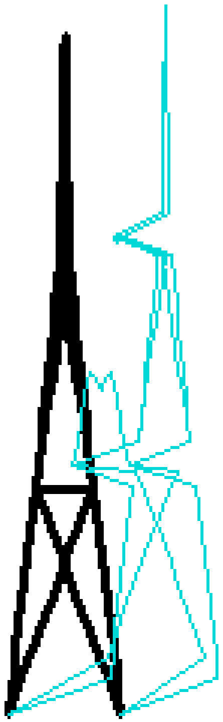
Modal mass: $m_i^{dir} = \left(\Gamma_i^{dir}\right)^2 / M_{tot}^{dir} \times 100\%\ , \qquad M_{tot}^{dir} = \left(\mathbf{MI}_{dir}, \mathrm{I}_{dir}\right)$

Property: $\sum_{i=1}^{N} m_i^{dir} = 100\%, \quad dir = OX, OY, OZ$

N **– number of degrees of freedom of finite element model,**
n **– number of eigenmodes, taken into account, usually *n* << *N***

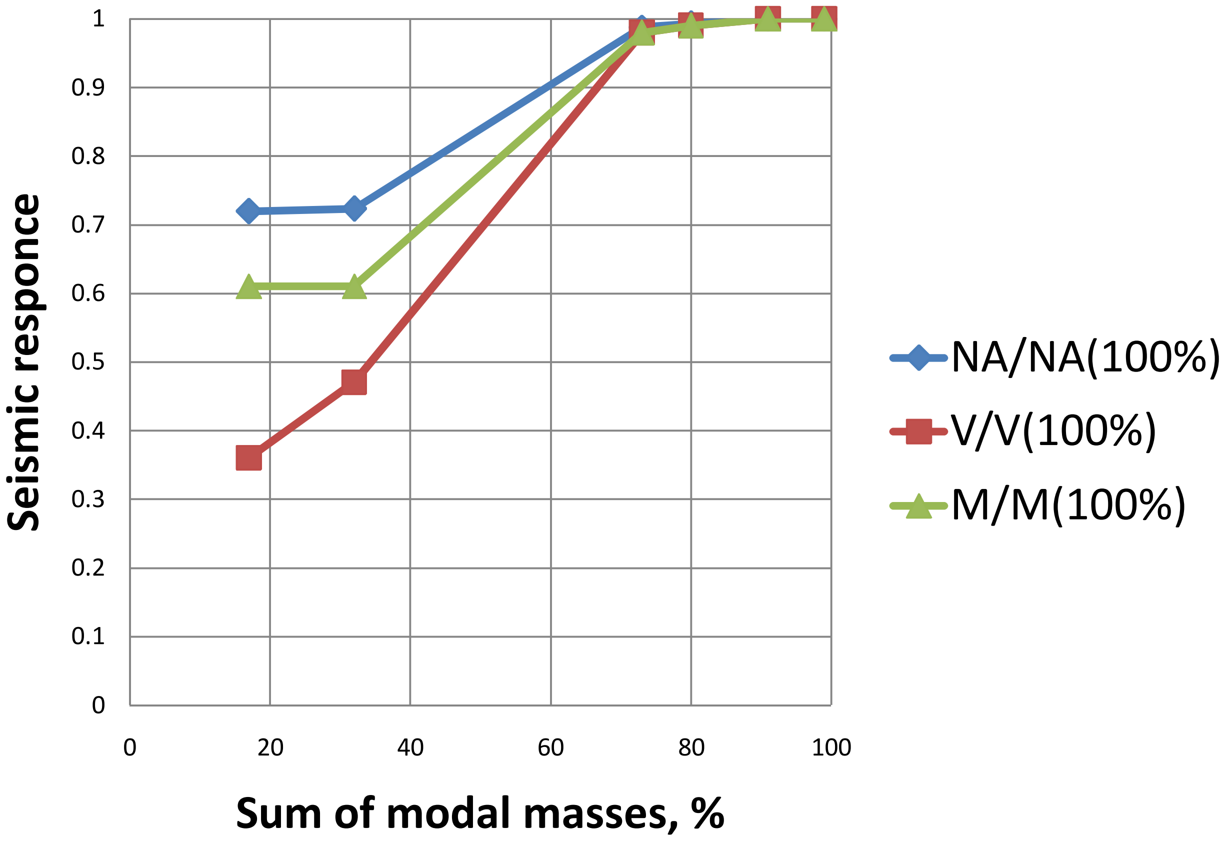
If all eigenmodes are taken into account (*n* = *N*), the sum of modal masses is 100% for each seismic input direction. Otherwise (*n* < *N*), the sum of modal masses is less than 100%. So, the sum of modal masses is a criteria: does the number of eigenmodes taken into account represent the seismic response well enough?

Seismic mode

			
Mode 1	**Mode 2**	**Mode 3**	**SRSS over 100 modes**

Example: 100 DOFs – 100 modes are extracted

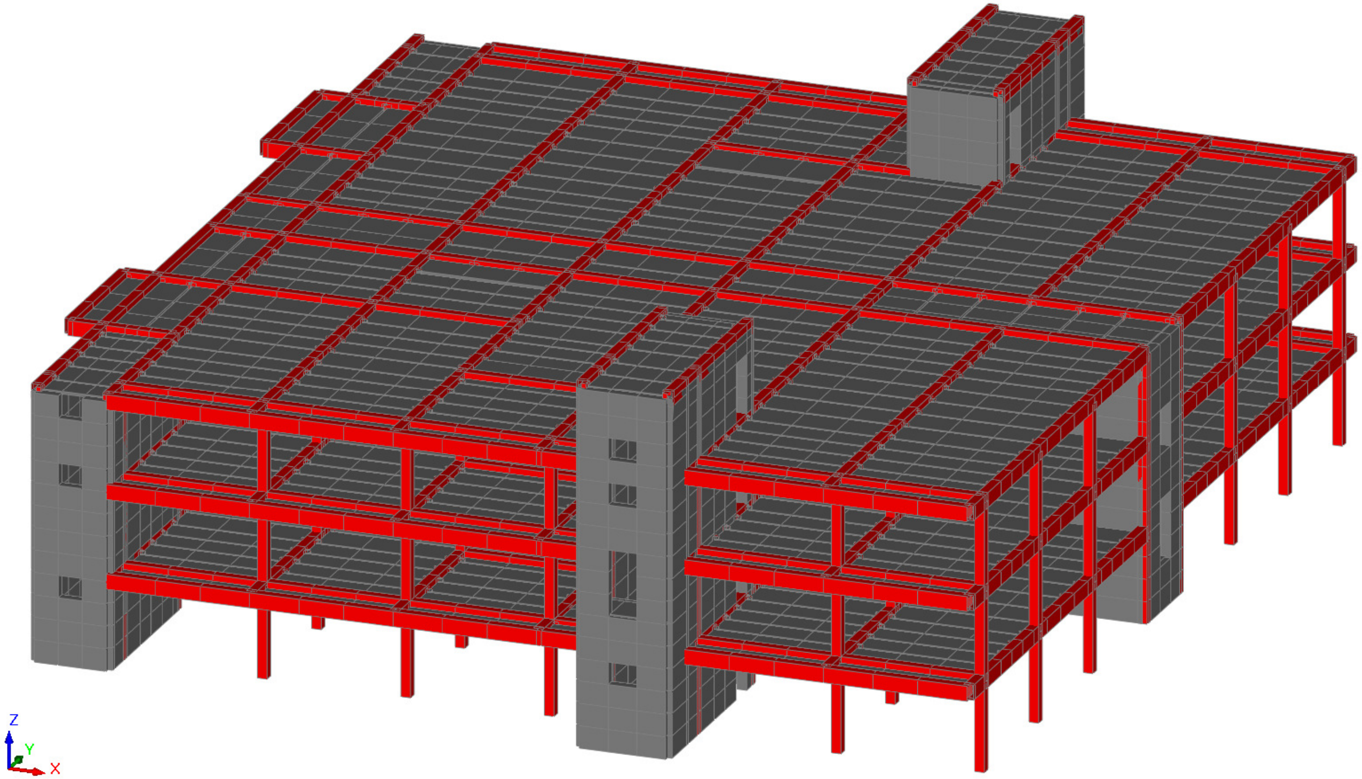
Seismic mode



N_A – axis force, N_A(100%) - axis force for 100% sum of modal masses
V – shear force, V(100%) – shear force for 100% sum of modal masses
M – overturning moment, M(100%) – overturn. moment for 100% sum of m. m.

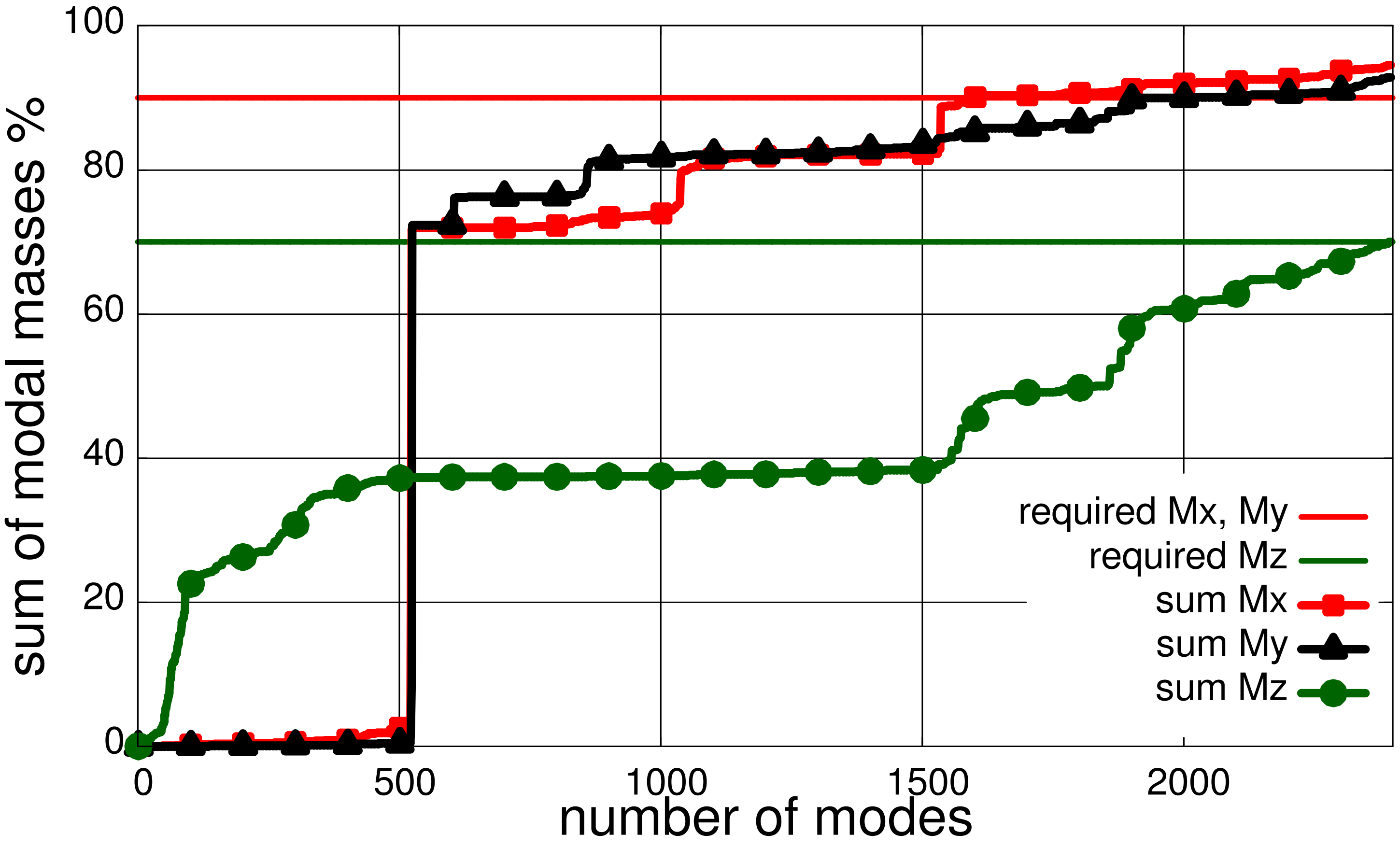
Seismic mode

Example [6]:



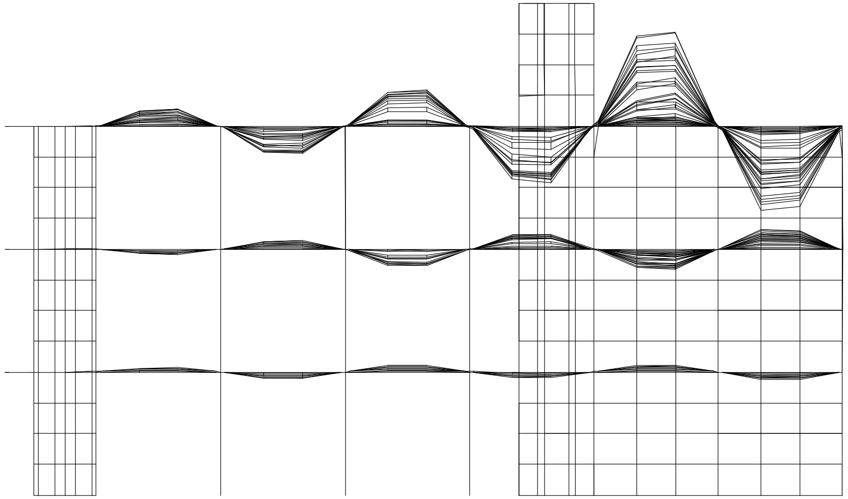
8 937 nodes, 9 073 finite elements and 52 572 equations.

Seismic mode

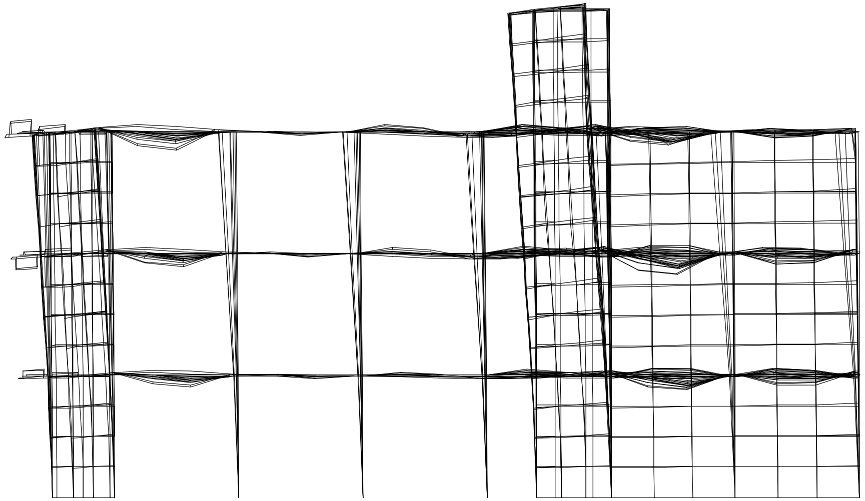


2 399 eigenpairs are required to ensure a sufficient sum of modal masses $\sum m_x = \sum m_y = 90\%$, $\sum m_z = 70\%$.

Seismic mode



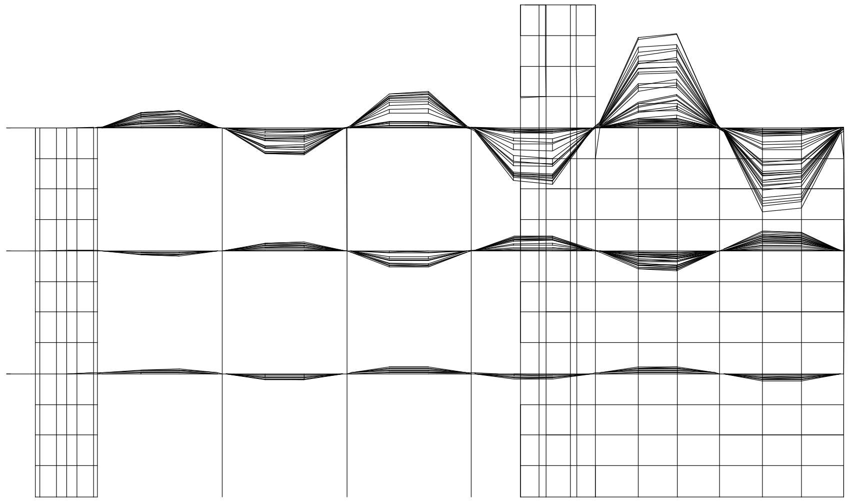
1st eigenmode, f = 4.185 Hz



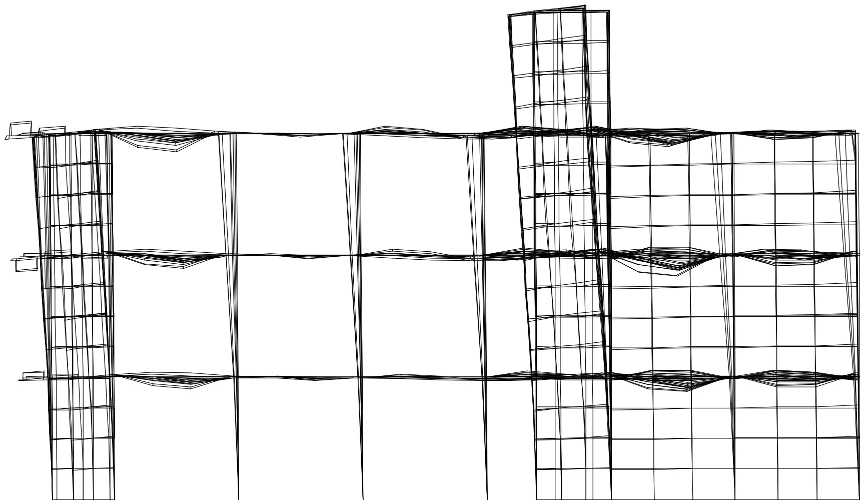
523rd eigenmode, f = 5.67 Hz

$$m_{523}^{OX} = 42\%$$

Seismic mode



1st eigenmode, f = 4.185 Hz



523rd eigenmode, f = 5.67 Hz

$$m_{523}^{OX} = 42\%$$

178 factorizations of the shifted stiffness matrix, 2097 solutions

Verification mode [7]

Is designed to detect the geometric instability.

Main idea: if the model is a geometrically unstable,

$\det\{\mathbf{K}\} = 0$ **and problem** $\mathbf{K}\psi - \lambda\mathbf{M}\psi = 0$

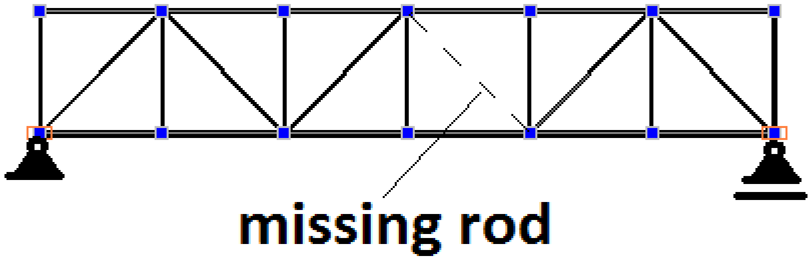
has zero eigenvalues. The corresponding eigenmodes presents the forms of movement mechanism.

The shift technique is applied to avoid a singularity during factorization:

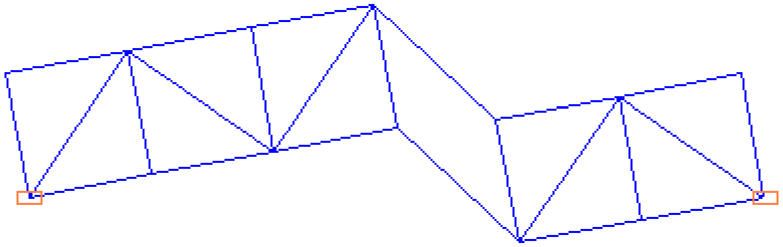
$$\mathbf{K}_\sigma = \mathbf{L}_\sigma \cdot \mathbf{S}_\sigma \cdot \mathbf{L}_\sigma^T$$

Verification mode

FEM model

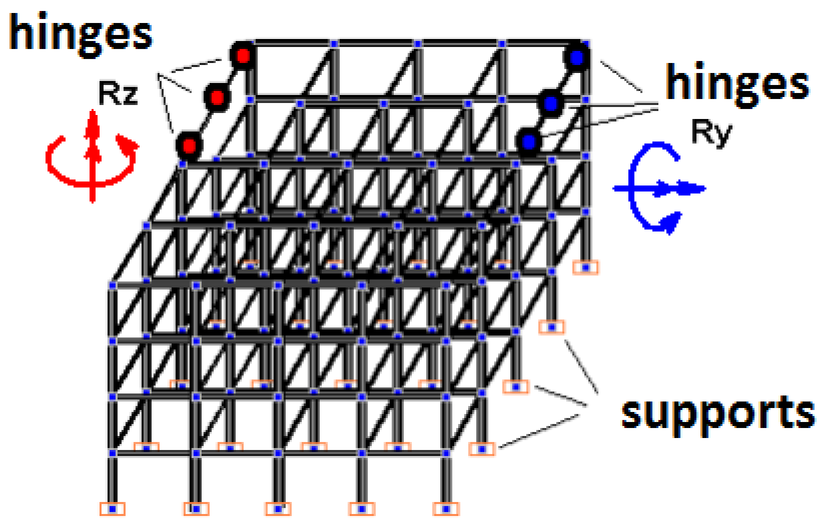


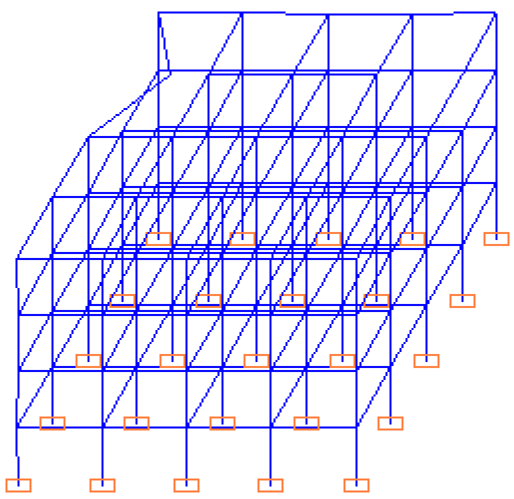
Eigenmode for $\lambda_1 = 0$



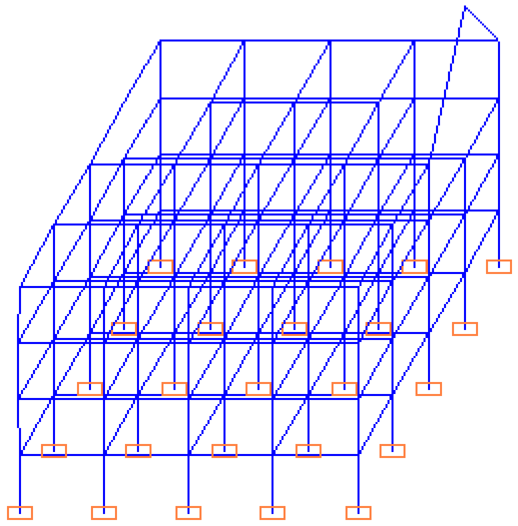
Verification mode

FEM model





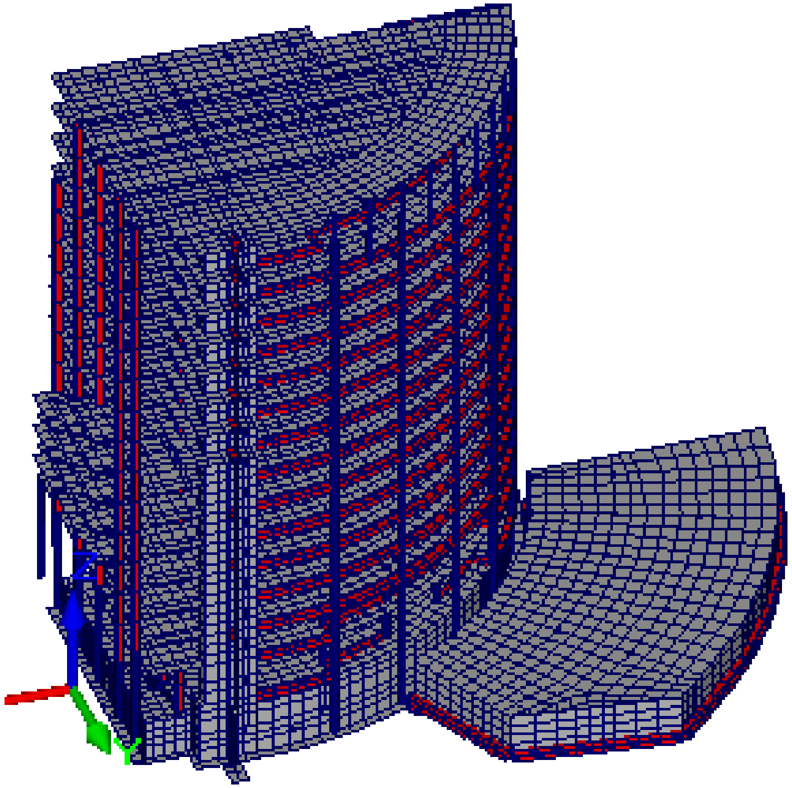
Eigenmode for
$\lambda_1 = 2.09 \cdot 10^{-8}$ Hz

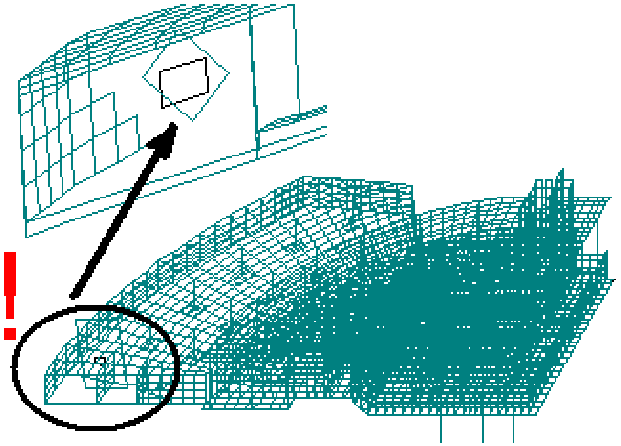


Eigenmode for
$\lambda_2 = 6.91 \cdot 10^{-8}$ Hz

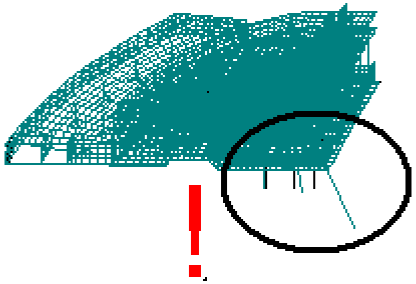
Verification mode

**FEM model: 24 434 nodes,
26 273 finite elements,
127 165 equations**





**6 eigenmodes for
$\lambda_1 < \ldots < \lambda_6 < 1.36 \cdot 10^{-7}$ Hz**



Unconstrained bottoms of columns !!!

CONCLUSIONS

1. The block Lanczos method with a spectral transformation is a powerful tool for modal & seismic analysis of large design models.

2. The presented realization contains the modal, interval, seismic and verification modes.

3. Seismic mode allows us to avoid the multiple repetitions of conventional modal mode when the required number of eigenpairs is is not known in advance.

4. Verification mode allows us to display the forms of mechanism movement and often to detect the another hardly-detected mistakes of design model.

REFERENCES

1. Amestoy PR, Duff IS, L'Excellent J-Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. Comput. Meth. Appl. Mech. Eng., 184: 501–520, 2000.
2. Dobrian F, Pothen A. Oblio: a sparse direct solver library for serial and parallel computations. Technical Report describing the OBLIO software library, 2000.
3. Fialko S. PARFES: A method for solving finite element linear equations on multi-core computers. Advances in Engineering software. v 40, 12, 2010, pp. 1256 – 1265.
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 modern FEM software. Moscow: SCAD SOFT , 2009. (in Russian).

REFERENCES

6. Fialko S. Realization of block Lanczos method with shifts in SCAD software applying to seismic analysis of structures. CADmaster #40/5.2007 (additional), p. 102 – 105. (In Russian).
7. Karypis G, Kumar V. METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, 1995.
8. Schenk O, Gartner K. Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems. Parallel Computing 28: 187–197, 2002.
9. Grimes, R.G. Lewis, J.G., Simon, H.D., A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, SIAM J. Matrix Anal. Appl, V.15, 1: pp. 1-45, 1994.

Thank you very much for your attention !