Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 655–662

DOI: 10.15439/2015F87
ACSIS, Vol. 5

Block Preconditioned Conjugate Gradient Method for Extraction of Natural Vibration Frequencies in Structural Analysis

Sergiy Fialko
Tadeusz Kościuszko Cracow
University of Technology
ul. Warszawska 24 St., 31-155 Kraków, Poland
Email: sergiy.fialko@gmail.com

Filip Żegleń
Tadeusz Kościuszko
Cracow University of Technology
ul. Warszawska 24 St., 31-155 Kraków, Poland
Email: filipzeglen@hotmail.com

***Abstract*—The block preconditioned conjugate gradient method for extraction of eigenfrequencies and eigenmodes is presented for finite element software in structural analysis. The proposed approach is focused on multi-core desktops and laptops and allows us to effectively analyze large design models, when classical methods based on the factoring of stiffness matrix, significantly reduce performance by intensive use of disk memory. The main attention is paid to proper construction of preconditioning, application of shift technique and creation of the block algorithm allowing the improvement of computing stability and multithreaded parallelization.**

I. Introduction

CURRENTLY, the vast majority of methods for determining the frequency and modes of natural oscillations in finite element software, usually based on factorization of stiffness matrix $\mathbf{K}$ or matrix $\mathbf{K}_\sigma = \mathbf{K} - \sigma\mathbf{M}$, where $\mathbf{M}$ is a mass matrix and σ is a shift.

However, if the dimension of the problem becomes large, the factored matrix does not fit in memory and is written to disk. Thus on each iteration in Lanczos method or in subspace iteration method it is necessary to read the factored matrix from a disk twice when performing forward and back substitutions. The performance of the specified methods considerably degrades because the task is carried out at an intensive use of a disk. Especially sharply this problem becomes when desktops and laptops with usually small amount of core memory are used.

It would seem the alternative approach consists in application of conjugate gradient method or steepest descent method for solution of the generalized algebraic eigenvalue problem to which the natural vibration problem is reduced when using the finite element method. The [13] presents survey of some other approaches using preconditioning technique. However, application of such methods to problems of structural mechanics usually results in to the slow convergence of the iterative process. Often there is a locking of convergence of numerical solution owing to a wide scattering of stiffness in design model, existence of a large number of close natural vibration frequencies etc. This is confirmed by the fact that the widely used commercial FEA software offer a wide selection of iterative solvers for static analysis, which leads to solving systems of linear algebraic equations. However for the solution of the generalized algebraic eigenvalue problem as a rule are used the methods based on factorization of a stiffness matrix. The exception is ANSYS in which the preconditioned conjugate gradient method is applied for generation of Lanczos vectors. The survey of existing software packages devoted to solution of algebraic eigenvalue problems in technical and scientific applications is in [10].

In this article, we confine ourselves to consideration of the methods applying to a class of problems, difficult for numerical realization, – to the large problems of structural mechanics arising when the finite element method is used to modeling of tall buildings and constructions. We present the preconditioned conjugate gradient (PCG) method for solution of partial generalized algebraic eigenvalue problem – extraction of natural vibration frequencies and modes

$$\mathbf{K}\mathbf{v}_i - \lambda_i \mathbf{M}\mathbf{v}_i = 0 \tag{1}$$

where $\mathbf{K}$ is a symmetrical positive definite stiffness matrix, $\mathbf{M}$ is a mass matrix, λ_i and $\mathbf{v}_i$ is an eigenpair, $i \in [1, n]$, $n << N$, N is dimension of problem.

For ensuring sustainable convergence of method, we paid the main attention to designing of an effective preconditioning, use of shift technique and creation of a block algorithm of PCG method. In addition, we used the multithreaded parallelization to accelerate computations.

II. Block Preconditioned Conjugate Gradient Solver

A. Sparse incomplete Cholesky conjugate gradient preconditioning

The problems of structural mechanics often demonstrate the slow convergence due to using of different types of finite elements, thin-walled finite elements of floors, roofs and walls, considerable scattering of stiffness etc. [15]. Therefore, it is crucial to construct the effective preconditioning for accelerating of convergence of the PCG method.

We use an incomplete Cholesky factoring "by value" approach, based on sparse matrix technique [6], allowing to keep a small value of drop parameter ψ ($\psi \in [10^{-9}, 10^{-20}]$). The small entries $\mathbf{H}_{ij} < \psi \cdot \mathbf{H}_{ii} \bullet \mathbf{H}_{jj}$ erasing on each step of

incomplete factoring are rejected and correction of diagonal entries $\mathbf{D}_{ii} = \mathbf{D}_{ii} + \sqrt{\mathbf{H}_{ii}/\mathbf{H}_{jj}} \cdot |\mathbf{H}_{ij}|, (i \leftrightarrow j)$ is produced to ensure a positive definiteness of incomplete Cholesky factor **H**. Thus, the amount of calculations and time of incomplete factorization remains within reasonable range. In addition, the secondary rejection of small off-diagonal entries is produced after incomplete factoring is finished: $\mathbf{H}_{ij}^2 < \psi_1 \cdot \mathbf{H}_{ii} \cdot \mathbf{H}_{jj}$, where $0 < \psi < \psi_1 < 1$. Such approach allows reducing the amount of data in preconditioning matrix **H** without considerable deterioration of properties of preconditioning. The details as well as the parallel algorithm of incomplete factoring is in [6].

In the earlier works of one of the authors used a multilevel aggregation preconditioning [3], [4], [15]. Use of sparse matrix technique allowed making an incomplete Cholesky conjugate gradient method competitive with aggregation multilevel PCG method.

Use of preconditioning leads to

$$\mathbf{B}^{-1}\mathbf{K}\mathbf{v}_i - \lambda_i \mathbf{B}^{-1}\mathbf{M}\mathbf{v}_i = 0, \quad (2)$$

where $\mathbf{B} = \mathbf{H} \cdot \mathbf{H}^T$.

B. Shift technique

It is well known that introduction of properly chosen shift drastically accelerate a convergence of inverse iteration method [16]. The influence of shift on ability of preconditioning to accelerate of convergence for gradient methods was considered in [3], [4]. Here we will give only the main conclusions of the theoretical analysis, which is carried out in the specified works, containing detailed substantiation.

Let us consider the preconditioning

$$\mathbf{B} = \mathbf{K} - \sigma\mathbf{M}, \quad \sigma = \lambda_1 + \delta, \quad (3)$$

where λ_1 is a first (minimal) eigenvalue and δ is a small value comparing with λ_1.

First, the preconditioning (3) provides that for σ closer to λ_1, the problem (1) converges faster. Then, when $\delta \to 0$, the procedure

$$\mathbf{x}_{k+1} = \left[\mathbf{I} - 2\alpha_k \mathbf{B}^{-1}(\mathbf{K} - \lambda_k \mathbf{M})\right]\mathbf{x}_k, \quad (4)$$

following from preconditioned gradient method, tends to step of inverse iteration method. Here k and k+1 are two consequent iteration steps, α_k – optimization parameter obtained from minimization of Rayleigh quotient.

Expressions (3) and (4) allow understanding better, in what form it is necessary to search an effective preconditioning.

Therefore, we will construct the preconditioning in form $\mathbf{H} \cdot \mathbf{H}^T - \sigma\mathbf{M}$, where shift σ will be taken as close to defining eigenvalue λ_i, as it is possible.

The preconditioner in PCG method is used for solution of additional equation set

$$(\mathbf{B} - \sigma\mathbf{M}) \cdot \mathbf{z}_k = \mathbf{r}_k, \quad \mathbf{r}_k = -\mathbf{g}_k, \quad (5)$$

where $\mathbf{r}_k$ is residual vector, $\mathbf{g}_k$ – gradient vector and k is an iteration step.

We never evaluate (3) directly, the presented below algorithm is used instead.

Algoritm 1. Implicit solution of equation set respectively preconditioning

$\mathbf{B} \cdot \mathbf{z}_k = \mathbf{r}_k \to \mathbf{z}_k$

do $s = 1, p$

$\mathbf{B} \cdot \mathbf{q} = \sigma\mathbf{M}\mathbf{z}_k \to \mathbf{q}$

$\mathbf{z}_k = \mathbf{z}_k + \mathbf{q}$

end do

Substantiation of presented above algorithm follows from

$$(\mathbf{B} - \sigma\mathbf{M}) \cdot (\hat{\mathbf{z}}_k + \mathbf{q}) = \mathbf{r}_k, \quad (6)$$

where $\hat{\mathbf{z}}_k$ is approximation of solution (5) and **q** is a small correction. When opening brackets, neglecting on the small order terms $\sigma\mathbf{M}\mathbf{q}$ and assuming $\mathbf{B}\hat{\mathbf{z}}_k \approx \mathbf{r}_k$, we obtain

$$\mathbf{B}\mathbf{q} - \sigma\mathbf{M}\hat{\mathbf{z}}_k = 0. \quad (7)$$

As showed numerous calculations, it is enough to produce only one iteration (p = 1).

C. Block PCG eigenvalue solver

Application of PCG method for extracting of few lower eigenpairs is presented in [2], [8], [9], [14], [17] and other works. In addition, [1] presents the subspace iteration method based on steepest descent approach with aggregation multilevel preconditioning. The realization of PCG eigenvalue solver, presented in [3] and [4] based on aggregation multilevel preconditioning, demonstrates a stable convergence on numerous problems of structural mechanics. Application of shifts in preconditioning makes this method efficient even for the problems having a large number of close natural vibration frequencies. However, this method extracts eigenpairs in strictly sequential mode. Each subsequent eigenpair is orthogonalized against to defined earlier when using of Gram – Schmidt orthogonalization procedure on each iteration step.

In this article, we use a different type of preconditioning and offer the block algorithm to reduce the number of iterations and apply a multi-threaded parallelization.

Algorithm 2. Shifted block PCG method (SBPCG)

1. $\sigma = 0$; $nconv = 0$; $k_{tot} = 0$;
2. ***parallel region 1*** *(prepare the block of start vectors,* $ip \in [0, np\text{-}1]$ *– thread number,* $pp = (N-1)/np + 1$, *np – dimension of block which is equal to number of threads).*

$\mathbf{x}_{ip} \leftarrow \mathbf{M} \cdot \mathbf{e}_{ip}, \mathbf{e}_{ip} = \left(e_{ip,j}\right),$

$e_{ip,j} = \begin{cases} 1, \; j \in \left[pp \cdot ip + 1, pp \cdot (ip+1)\right] \\ 0, otherwise \end{cases}$

$\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} \Big/ \sqrt{\left(\mathbf{x}_{ip}\right)^T \mathbf{M}\mathbf{x}_{ip}}$

$\lambda_{ip} = \left(\mathbf{x}_{ip}\right)^T \mathbf{K}\mathbf{x}_{ip}$

$\mathbf{r}_{ip} = \lambda_{ip}\mathbf{M}\mathbf{x}_{ip} - \mathbf{K}\mathbf{x}_{ip}$

$\mathbf{B}\mathbf{z}_{ip} = \mathbf{r}_{ip} \rightarrow \mathbf{z}_{ip}$

$\mathbf{p}_{ip} = -2\mathbf{z}_{ip}$

end of parallel region I

3. ***Extracting all required modes:***
   while (*nconv* < *nmodes*)
   conv = 0; k = 0
4. *while* (!*conv* ∧ $k < k_{max}$)
   parallel region II *(minimization of Rayleigh quotient and line search procedure)*

$$\frac{\partial R\left(\mathbf{x}_{ip} + \alpha_{ip}\mathbf{p}_{ip}\right)}{\partial \alpha_{ip}} = 0 \quad \rightarrow \alpha_{ip}$$

$\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} + \alpha_{ip}\mathbf{p}_{ip}$

$\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} \Big/ \sqrt{\left(\mathbf{x}_{ip}\right)^T \mathbf{M}\mathbf{x}_{ip}}$

end of parallel region II

5. ***Orthogonalization in block***
6. ***parallel region III*** *(shifted PCG procedure)*

$\lambda_{ip,\sigma} = \mathbf{x}_{ip}^T \mathbf{K}_\sigma \mathbf{x}_{ip}$

$\mathbf{r}_{ip} = \lambda_{ip,\sigma}\mathbf{M}\mathbf{x} - \mathbf{K}_\sigma \mathbf{x}_{ip}$

$$if\left(\frac{\left\|\mathbf{r}_{ip}\right\|_2}{\left|\lambda_{ip}\right| \cdot \left\|\mathbf{M}\mathbf{x}_{ip}\right\|_2} < tol\right)$$

conv = 1

else

$\mathbf{B}_\sigma \mathbf{z}_{ip} = \mathbf{r}_{ip} \quad \rightarrow \mathbf{z}_{ip}$

$$\beta_{ip} = \frac{2\mathbf{z}_{ip}^T \mathbf{K}_\sigma \mathbf{p}_{ip}^k}{\mathbf{p}_{ip}^T \mathbf{K}_\sigma \mathbf{p}_{ip}^k}$$

$\mathbf{p}_{ip} \leftarrow \beta_{ip}\mathbf{p}_{ip} - 2\mathbf{z}_{ip}$

Orthogonalize $\mathbf{p}_{ip}$ *against* $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$

end else

end of parallel region III

k++; k_{tot}++;
end of while (*p.4*)

7. *if*(*conv*)

nconv++; $\mathbf{v}_{nconv} \leftarrow \mathbf{x}_{ip}^{k+1}$; $\lambda_{nconv} \leftarrow \lambda_{ip,\sigma} + \sigma$;
if(*flag_auto*) $k_{\max} = k_{tot}/nconv$;
Add vector in block :

$\mathbf{x}_{ip} \leftarrow \mathbf{M} \cdot \mathbf{e}_{ip}$

Orthogonalize $\mathbf{x}_{ip}$ *against* $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$

Orthogonalize $\mathbf{x}_{ip}$ *against another vectors in block*

$\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} \Big/ \sqrt{\mathbf{x}_{ip}^T \mathbf{M}\mathbf{x}_{ip}}$

$\lambda_{ip,\sigma} = \mathbf{x}_{ip}^T \mathbf{K}_\sigma \mathbf{x}_{ip}$

$\mathbf{r}_{ip} = \lambda_{ip,\sigma}\mathbf{M}\mathbf{x}_{ip} - \mathbf{K}_\sigma \mathbf{x}_{ip}$

parallel region IV

$\mathbf{B}_\sigma \mathbf{z}_{ipp} = \mathbf{r}_{ipp} \quad \rightarrow \mathbf{z}_{ipp}, ipp \in \left[0, np-1\right]$

if (*ipp* == *ip*)

$\mathbf{p}_{ipp} = -2\mathbf{z}_{ipp}$

else

$$\beta_{ipp} = \frac{2\mathbf{z}_{ipp}^T \mathbf{K}_\sigma \mathbf{p}_{ipp}^k}{\mathbf{p}_{ipp}^T \mathbf{K}_\sigma \mathbf{p}_{ipp}^k}$$

$\mathbf{p}_{ipp} \leftarrow \beta_{ipp}\mathbf{p}_{ipp} - 2\mathbf{z}_{ipp}$

end if

Orthogonalize $\mathbf{p}_{ipp}$ *against* $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$

end of parallel region IV

end of Add vector in block
end if(conv)

8. *else*

$\sigma \leftarrow \min\left\{\lambda_{ip,\sigma}\right\} + \sigma; \quad k = 0;;$

parallel region V *(modification of shift)*

Orthogonalize $\mathbf{x}_{ip}$ *against* $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$

$\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} \Big/ \sqrt{\mathbf{x}_{ip}^T \mathbf{M}\mathbf{x}_{ip}}$

$\mathbf{K}_\sigma \leftarrow \mathbf{K} - \sigma\mathbf{M}$

$\lambda_{ip,\sigma} = \mathbf{x}_{ip}^T \mathbf{K}_\sigma \mathbf{x}_{ip}$

$\mathbf{r}_{ip} = \lambda_{ip,\sigma}\mathbf{M}\mathbf{x}_{ip} - \mathbf{K}_\sigma \mathbf{x}_{ip}$

$\mathbf{B}_\sigma \mathbf{z}_{ip} = \mathbf{r}_{ip} \rightarrow \mathbf{z}_{ip}$

$\mathbf{p}_{ip} = -2\mathbf{z}_{ip}$

Orthogonalize $\mathbf{p}_{ip}$ *against* $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$

end of parallel region V

end else (p. 8)
end while (*p. 3*)

Here *N* – dimension of problem, *np* – dimension of block (number of threads); *nconv* – number of converged modes, *nmodes* – number of required modes; *k* – iteration number, k_{tot} – total number of iterations for *nconv* modes; σ – shift value; *flag_auto* = *true* – algorithm produces an automatic corrections of k_{max} – number of iterations on which excess the correction of shift is made; *flag_auto* = *false* – algorithm

produces k_{max} iterations between correction of shift imposed by user.

We create a block of mutually orthogonal start vectors (p. 2) in the first parallel region, when each thread *ip* prepares start vector $\mathbf{x}_{ip}$, approximation of eigenvalue λ_{ip}, residual vector $\mathbf{r}_{ip} = -\mathbf{g}_{ip}$ and conjugate direction vector $\mathbf{p}_{ip}$. The normalization of vector $\mathbf{x}_{ip}^T \mathbf{M} \mathbf{x}_{ip} = 1$ allows simplify evaluation of Rayleigh quotient $R(\mathbf{x}_{ip}) = \lambda_{ip} = \mathbf{x}_{ip}^T \mathbf{K} \mathbf{x}_{ip}$.

Loop *while* (p. 3) runs until all required eigenpairs will be extracted. The next loop *while* (p. 4) produces iterations in block. The achievement of convergence of one or several vectors in block as well as exceeding of allowed number of iterations k_{max} interrupt this loop. The Rayleigh quotient minimization results in optimal value of parameter α_{ip} and we produce the line search procedure $\mathbf{x}_{ip} \leftarrow \mathbf{x}_{ip} + \alpha_{ip} \cdot \mathbf{p}_{ip}$ and normalization of $\mathbf{x}_{ip}$.

The orthogonalization of all vectors $\mathbf{x}_{ip}$ in block (p. 5) prevents the convergence of all vectors to the same eigenvector. This procedure is a single sequential fragment in loop *while* (p. 4). All remaining operations in this loop are produced in parallel regions II and III.

The parallel region III evaluates the current approximation of eigenvalue for each vector $\mathbf{x}_{ip}$ in block and residual vector $\mathbf{r}_{ip}$. If relative norm of residual vector is less than tolerance *tol*, we assume that corresponding vector $\mathbf{x}_{ip}$ is converged and set flag *conv* to 1. Otherwise, we resolve the additional equation set relatively preconditioning $\mathbf{B}_\sigma = \mathbf{H} \cdot \mathbf{H}^T - \sigma \mathbf{M}$, using algorithm 1, and derive the new conjugate direction $\mathbf{p}_{ip} \leftarrow \beta_{ip} \mathbf{p}_{ip} - 2\mathbf{z}_{ip}$. Orthogonalization of conjugate direction vector against converged eigenmodes $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$ prevents the duplication of converged eigenpairs.

If loop *while* (p. 4) was interrupted due to convergence of $\mathbf{x}_{ip}$ (p. 7), we increment *nconv* and put the current approximations $\mathbf{x}_{ip}$ and λ_{ip} as a converged eigenpair $\{ \lambda_{nconv}, \mathbf{v}_{nconv} \}$. Then, we put the new start vector $\mathbf{x}_{ip}$ in block instead of converged vector (*Add vector in block*). We orthogonalize $\mathbf{x}_{ip}$ against all converged modes and against all remaining vectors in block, normalize it and evaluate the initial approximation of eigenvalue $\lambda_{ip,\sigma}$. The current value of σ is used. We evaluate residual vector $\mathbf{r}_{ip}$. Then in parallel region IV we solve linear equations relatively preconditioning and for just added vector (*ipp* == *ip*), derive vector $\mathbf{z}_{ip}$ and conjugate direction vector $\mathbf{p}_{ip}$. For remaining vectors in block (*ipp* ≠ *ip*) we obtain the new conjugate directions. Orthogonalization of $\mathbf{p}_{ipp}$, *ipp* ∈ [0, *np*-1] against all converged eigenmodes prevents from duplicating of already defined eigenpairs. In addition, if *flad_auto* is turned on, we correct k_{max}, taking it as an average number of iterations required for achieving of convergence.

If loop *while* (p. 4) was interrupted due to exceeding of k_{max} (slow convergence), block *else* (p. 8) is processed. We accept new shift value as a minimum from all current approximations of eigenvalues. Current approximations of eigenmodes $\mathbf{x}_{ip}$ remain unchanged, but the conjugate directions accumulated earlier are lost because the each changing of shift results in modification of matrix $\mathbf{K}_\sigma$. Since with every iteration step the error in vectors $\mathbf{x}_{ip}$ accumulates, in parallel region V we produce the orthogonalization against previously defined eigenvectors to increase the computing stability of the method. Then, vectors $\mathbf{x}_{ip}$ are normalized, new values of current approximations of eigenvalues $\lambda_{ip,\sigma}$ and residual vectors $\mathbf{r}_{ip}$ are derived. We obtain vectors $\mathbf{z}_{ip}$ and conjugate direction vectors $\mathbf{p}_{ip}$, which should be orthogonolized against converged eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_{nconv}$.

We cannot produce the modification of shift too often because each shift modification destroys the accumulated conjugate directions. In a limit if to make correction of shift on each step of iteration, the conjugate gradient method becomes method of the steepest descent, and convergence degrades. Therefore, there is a question: how often it is necessary to make shift correction? In this version of algorithm, the parameter k_{max} operates by it. We can prescribe k_{max} directly (*flag_auto* == *false*), and can determine k_{max} as average number of iterations, is required to achieve convergence of n_{conv} modes – $k_{max} = k_{tot}/n_{conv}$, where k_{tot} is a number of iterations required to obtain n_{conv} modes (*flag_auto* == *true*).

Proposed block PCG method is different from [9], [12], [17] first of all by fact that use of shifts in preconditioning considerably accelerates convergence. Secondly, the proposed version of block algorithm does not require forming projection matrices on subspace $\mathbf{X}^T\mathbf{K}\mathbf{X}$, $\mathbf{X}^T\mathbf{M}\mathbf{X}$, where $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{np-1}\}$.

D. Different approaches for parallelization

Different strategies of parallelization for PCG method can be based on parallelization of its individual parts – matrix vector multiplication, triangular solution or dot product of two vectors. The most time-consuming operations in PCG method are symmetrical sparse matrix vector multiplication ($\mathbf{K}\mathbf{v}$) and forward and backward substitutions ($\mathbf{B}\mathbf{z} = \mathbf{r} \rightarrow \mathbf{z}$ – triangular solution). These two procedures require about 90% of total solution time. That is why in this article we focus our attention only on them. These operations are implemented in Intel Math Kernel Library (Intel MKL). We create a special option denoted as PCG MKL using *mkl_dcsrsymv*() procedure to compute the matrix vector product and *mkl_dcsrtrsv*() for computing of triangular solution [18]. All mentioned procedures are taken from Intel MKL.

In all other realizations of proposed approach, we use in-house procedures for computing of matrix-vector product as well as for triangular solutions. The upper triangular parts of matrices $\mathbf{K}$ and $\mathbf{H}$ are stored in compressed row format (CRF) which is close to standard CRF from Intel MKL.

We found that parallelization approach which uses PCG MKL option, very poorly accelerate the PCG method for solution of eigenvalue problem. That is why we have created SBPCG method, where parallelization covers relatively large computing regions. This significantly reduces the number of

entries in the parallel regions and thus improve thread management.

III. Numerical Results

Let us consider examples taken from collection of SCAD Soft (http://www.scadsoft.com) — IT Company, developer of the SCAD FEA software, one of the most popular software used in the CIS countries for structural analysis and design, certified according to the regional norms.

We used the following computers:

A. Server, 16-core processor AMD Opteron 6276, 2.3/3.2 GHz, 64 GB DDR3 RAM, OS Windows Server 2008 R2 Enterprise SP1, 64 bit.

B. Laptop DELL XPSL502X, 4-core processor Intel Core™ i7-27600QM, 2.4/3.4 GHz, RAM DDR3 8GB, OS Windows 7 (64 bit) Professional, SP1.

First computer has large amount of RAM that allows applying shifted block Lanczos method [5] in core mode. Sparse direct solver PARFES [7] is used for factoring matrix $\mathbf{K} - \sigma\mathbf{M}$. Second computer due to restricted amount of core memory compels a shifted block Lanczos method to work in out-of-core mode for large problems. In such case, the block PCG method can be very efficient.

We denote the shifted block PCG method and shifted block Lanczos method respectively SBPCG and SBLANC.

A. Problem 1

The design model of multistory building (Fig. 1) contains 2 002 848 equations and consists of three-node triangular and four-node quadrilateral flat shell finite elements as well as bar finite elements, elastic supports and rigid links.

Table I shows the number of iterations and time of eigenvalue solution for SBPCG method with $\psi = 10^{-16}$, $\psi_1 = 10^{-13}$, $tol = 10^{-3}$.

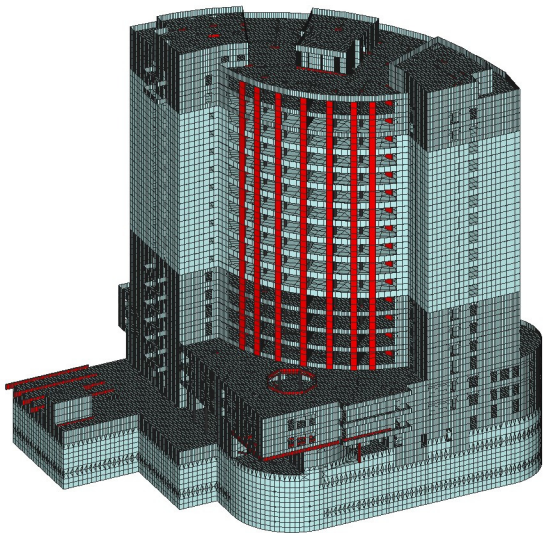

Fig. 1 Design model of multistory building (2 002 848 equations)

Adopted tolerance ensures about four true digits in eigenvalue; it is a very high precision for practice. In addition, we present the solution time of this problem by shifted block PCG method with conventional ICCG0 preconditioning "by position". The sparsity pattern of incomplete Cholesky factoring **H** is the same as initial stiffness matrix **K**. Its positive definiteness is ensured due to decreasing of off-diagonal entries when negative elements arise on main diagonal during incomplete factoring [6].

TABLE I
PROBLEM 1. TIME OF EIGENVALUE SOLUTION. COMPUTER A (AMD OPTERON 6276, 64 GB RAM)

np	Nos of iterations	Time of eigenvalue solution, s
SBPCG without shift ($\sigma = 0$, $k_{max} >> N$)		
1	5 537	25 825
2	2 982	16 529
3	1 969	11 700
4	2 343	18 078
SBPCG with shift correction over each 100 iterations ($k_{max} = 100$)		
1	1451	6 560
2	871	4 390
3	920	5 289
4	796	6 061
SBPCG with shift correction over automatically accepted number of iterations *(flag_auto = true)*		
1	769	5 421
2	501	3 831
3	405	3 366
4	424	3 877
SBPCG (ICCG0 preconditioning) with shift correction over automatically accepted number of iterations *(flag_auto = true)*		
3	133 135	302 025
SBLANC (core mode)		
16	–	1 662 s

It appeared that for this problem when using a conventional ICCG0 preconditioning the convergence is unacceptably slow. This result emphasizes, it is how important to create an effective preconditioning for the successful solution of the considered class of problems.

Also, we present a solution time demonstrated by SBLANC solver.

All results have been obtained on computer A, 30 eigenpairs were extracted.

Application of shift in preconditioning accelerates a convergence in multiple times. The best results for the block PCG method is obtained when automatic correction of k_{max} is applied and dimension of block $n_p = 3$.

SBLANC runs in core mode. Thus, it solved this problem about two times faster than SBPCG method.

Table II shows results obtained on computer B. For SBPCG method we take $\psi = 10^{-14}$, $\psi_1 = 10^{-11}$, $tol = 10^{-3}$.

TABLE II
PROBLEM 1. TIME OF EIGENVALUE SOLUTION. COMPUTER B (INTEL CORE™ I7-27600QM, 8 GB RAM)

np	Nos of iterations	Time of eigenvalue solution, s
	SBPCG with shift correction over automatically accepted number of iterations (*flag_auto* = *true*)	
3	502	1 831
	SBLANC (out-of-core mode)	
4	–	14 253

Size of factorized stiffness matrix is 8.31847 GB (METIS reordering [11] is used) and exceeds the capacity of core memory. Therefore, solution time for SBLANC method is much longer than solution time for SBPCG.

B. Problem 2

The design model of supermarket (Fig. 2) comprises 80 244 equations and consists of finite elements of spatial frame modelling spatial truss and quadrilateral flat shell finite elements.

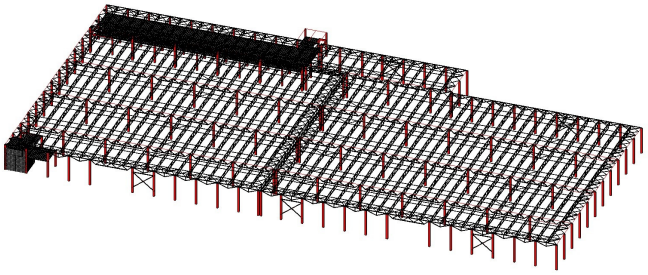

Fig. 2 Design model of supermarket (80 244 equations)

Such types of structures have lot of very close eigenvalues caused by local vibrations of rods of spatial truss (Table III).

TABLE III
NATURAL VIBRATION FREQUENCIES (HZ) OF STADIUM MODEL

mode	1	2	3	4	5
frequency	0.7133	0.7277	0.7996	0.8555	0.8644
mode	6	7	8	9	10
frequency	0.8835	0.8919	0.9280	1.0209	1.0485
mode	11	12	132	14	15
frequency	1.0519	1.0908	1.1421	1.1445	1.1546
mode	16	17	18	19	20
frequency	1.2219	1.3015	1.3024	1.3034	1.3051

This task is very difficult for the iterative method, and we use it as a test that verifies reliability of SBPCG. It is not a large problem, and results have been obtained on computer B.

We accept $\psi = 10^{-20}$, $\psi_1 = 10^{-17}$, $tol = 10^{-3}$. Table IV demonstrates following.

TABLE IV
PROBLEM 2. TIME OF EIGENVALUE SOLUTION. COMPUTER B (INTEL CORE™ I7-27600QM, 8 GB RAM)

np	Nos of iterations	Time of eigenvalue solution, s
	SBPCG 10 eigenmodes, without shift ($\sigma = 0$, $k_{max} >> N$)	
1	Lock of convergence on 3-th mode, solution is failed	
2	Lock of convergence on 6-th mode, solution is failed	
3	Lock of convergence on 7-th mode, 7 mode is missing	
4	Lock of convergence on 7-th mode, 7 mode is missing	
8	258	19.2
	SBPCG 10 eigenmodes, with shift correction (*flag_auto* = *true*)	
1	Lock of convergence on 3-th mode, solution is failed	
2	Lock of convergence on 6-th mode, solution is failed	
3	149	5.5
4	160	6.8
	SBPCG 50 eigenmodes, with shift correction (*flag_auto* = *true*)	
4	771	67.2

When the method is non-block (*np* = 1) or dimension of block is two (*np* = 2), lock of convergence occurs on 3-th mode. Shift correction is not used ($\sigma = 0$). Thus, computation process goes to infinite looping, and the solution cannot be obtained.

When we increase the dimension of the block up to 3-4, cycling it is possible to avoid, however the 7-th mode is skipped. Only at dimension of the block 8 the problem is solved correctly.

When using shift in a preconditioning the correct solution manages to be received at dimension of the block starting with three. As a test on the computational stability of the algorithm for this problem was defined 50 eigenpairs, dimension of block was taken four.

Thus, we see that in case of the difficult tasks having a large number of close natural vibration frequencies, the increase in dimension of the block improves computing stability of a method. However, the most effective is use of shift technique at iteration in the block.

C. Problem 3

The design model of stadium (Fig. 3) comprises 4 033 620 equations and consists of finite elements of spatial frame modelling spatial trusses, triangular and quadrilateral flat shell finite elements, elastic supports and rigid links.

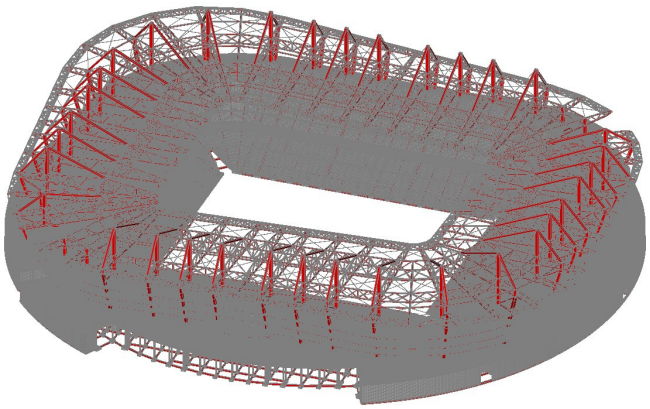

Fig. 3 Design model of stadium (4 033 620 equations)

Such types of structures as well as previously considered problem 2 have lot of very close eigenvalues caused by local vibrations of rods of spatial truss (Table V).

TABLE V
NATURAL VIBRATION FREQUENCIES (HZ) OF STADIUM MODEL

mode	1	2	3	4	5	6
frequency	0.0878	0.0878	0.6621	0.6621	0.7036	0.7036
mode	7	8	9	10	11	12
frequency	0.7289	0.7289	0.7969	0.7969	0.8564	0.8564
mode	13	14	15	16	17	18
frequency	0.8891	0.8892	0.9660	0.9660	1.007	1.007
mode	19	20				
frequency	1.025	1.025				

Solution of this problem on computer A with 64 GB RAM (Table VI) allows keeping parameters of SBPCG method as $\psi = 10^{-16}$, $\psi_1 = 10^{-13}$, $tol = 10^{-3}$, 100 eigenpairs are extracted.

TABLE VI
PROBLEM 3. TIME OF EIGENVALUE SOLUTION. COMPUTER A (AMD OPTERON 6276, 64 GB RAM)

np	**Nos of iterations**	**Time of eigenvalue solution, s**
SBPCG without shift ($\sigma = 0$, $k_{max} >> N$)		
1	Lock of convergence on 27-th mode, solution is failed	
2	Lock of convergence on 27-th mode, solution is failed	
3	Lock of convergence on 45-th mode, solution is failed	
4	Lock of convergence on 48-th mode, solution is failed	
SBPCG with shift correction over each 100 iterations ($k_{max} = 100$)		
1	5 295	43 151
2	3 471	29 775
3	3 126	34 869
4	2 873	32 859
SBPCG with shift correction over automatically accepted number of iterations *(flag_auto = true*)		
1	2 828	35 271
2	1 373	19 989
3	1 452	22 089
4	1 082	20 708
SBLANC (core mode)		
16	–	6 228

The introduction of properly chosen shift in preconditioning is a factor of crucial importance in achieving of stable convergence. Considered problem with large number of very close eigenvalues underline this.

When we use SBPCG method without shift ($\sigma = 0$, $k_{max} >> N$), the convergence of method is not stable. The increasing of block dimension *np* slightly improve a situation, but does not solve fully this problem.

Only a shift in the preconditioning provides a stable convergence of the method and significant reduces the number of iterations (k_{max} = 100 and *flag_auto* = *true* options affect only to the technique of shift correction).

It is interesting to note that in all of the examples, as well as in other similar tests performed by authors, when using shift correction an optimal result is achieved with a block size of *np* = 3 – 4. Further increase the dimension of the block slightly reduces the number of iterations required to reach convergence for all required eigenmodes. We believe it is because the shift is as close as possible to the current approximation of eigenvalue corresponding to the lowest frequency for iterated vectors in block. Therefore, the higher the frequency for the iterated vector in block, the weaker the influence of shear.

Shifted block Lanczos method runs in core mode and solved this problem about 3 times faster than SBPCG method.

Solution of this problem on computer B with 8 GB RAM allows keeping parameters of SBPCG method as $\psi = 10^{-13}$, $\psi_1 = 10^{-8}$, $tol = 10^{-3}$, np = 3, 20 eigenpairs are extracted. We compare these results with shifted block Lanczos method, which run in out-of-core mode. The comparison of eigenvalue solution time presented in Table VII demonstrates that SBPCG method is about 3 times faster.

TABLE VII
PROBLEM 3. TIME OF EIGENVALUE SOLUTION. COMPUTER B (INTEL CORE™ I7-27600QM, 8 GB RAM)

np	**Nos of iterations**	**Time of eigenvalue solution, s**
SBPCG with shift correction over automatically accepted number of iterations (*flag_auto = true*)		
3	798	3 913
SBLANC (out-of-core mode)		
4	–	12 765

D. Comparison of different methods of parallelization

In this section, we consider the PCG eigenvalue solver with the same type of preconditioning and shift technique, presented in given article, but with another type of parallelization on SMP multicore computers. The parallelization is produced internally in procedures *mkl_dcsrsymv*() and *mkl_dcsrtrsv*() taken from Intel MKL. First procedure performs matrix vector product **Kv** and second – triangular solution $\mathbf{Bz} = \mathbf{r} \rightarrow \mathbf{z}$. We denoted earlier such option of analysis as PCG MKL.

The *Problem 1* (Fig. 1) with the same parameters of preconditioning is considered. Analysis is done on computer A. Table VIII shows the duration of matrix vector product procedure during all iterations, duration of triangular solution, total duration of iterations and number of iterations depending on thread numbers *np*. We accept $\psi = 10^{-16}$, $\psi_1 = 10^{-13}$, $tol = 10^{-3}$.

TABLE VIII
PROBLEM 1. PCG MKL APPROACH –PARALLELIZATION INTERNALLY MATRIX VECTOR PRODUCT AND TRIANGULAR SOLUTION PROCEDURES. COMPUTER A (AMD OPTERON 6276, 64 GB RAM)

np	**Time of K*v**	**Time of B*z = r → z**	**Time of iterations, s**	**Nos of iterations**
PCG MKL with shift correction over each 100 iterations ($k_{max} = 100$)				
1	242.2	8 137.5	9 342	1144
2	163.5	8 150.2	9 277	1144
3	127.8	8 121.3	9 212	1144
4	113.1	8 121.1	9 196	1144

We found that increasing of thread numbers accelerates only matrix vector product procedure **Kv**. The duration of

triangular solution procedure practically does not depend on number of threads. The duration of triangular solution for considered class of problem of structural mechanics, requiring a preconditioner of high ability to improve of convergence, is essentially larger than duration of **Kv** and other remaining procedures. Therefore, namely this procedure should be accelerated firstly. However, it is a large problem on SMP multicore computers [6].

Table IX depicts results obtained by non-block sequential PCG eigenvalue solver presented in [3], [4].

TABLE IX
PROBLEM 1. NAÏVE SEQUENTIAL APPROACH. COMPUTER A (AMD OPTERON 6276, 64 GB RAM)

np	Time of K*v	Time of B*z = r → z	Time of iterations, s	Nos of iterations
PCG with shift correction over each 100 iterations (k_{max} = 100)				
1	326	6107	7601	1144

This method uses naïve algorithms of matrix vector product and triangular solution, taking into account only symmetry of sparse matrices. Besides, the loop unrolling is used in triangular solution procedure.

Comparison between Tables VIII and IX show that the application of the matrix vector product procedure in the Intel MKL runs up to 3 times faster. However, the procedure of a triangular solution of Intel MKL is slower than naïve. This leads to the fact that the naïve sequential PCG method solves the problems of this class faster.

The problem with acceleration of triangular solution procedure does inefficient the approach based on internal parallelization of leading procedures of PCG method.

III. CONCLUSION

The block conjugate gradient method with shifts in sparse incomplete Cholesky factorization preconditioning is proposed for extraction of lower eigenpairs applying to natural vibration problems arising due to application of finite element method to problems of structural mechanics.

On examples of typical problems of structural mechanics, it is shown that on achievement of high computing stability of a method the specific construction of a preconditioning, introduction of shifts to a preconditioning, and iterations in the block have a crucial importance.

The comparison with shifted block Lanczos method based on parallel sparse direct solver PARFES shows that proposed SBPCG method may be very efficient on computers with restricted amount of core memory, when factorized stiffness matrix block-by-block is stored on disk. In such a situation, the proposed SBPCG method in contrast to Lanczos method runs in core memory.

REFERENCES

[1] V. E. Bulgakov, M. E. Belyi and K. M. Mathisen, "Multilevel aggregation method for solving large-scale generalized eigenvalue problems in structural dynamics," *Int. J. Numer. Methods Eng.,* vol. 40. pp. 453–471, 1997, http://DOI: 10.1002/(SICI)1097-0207(19970215)40:33.0.CO;2-2.

[2] Y. T. Feng and D. R. J. Owen, "Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems," *Int. J. Numer. Methods Eng.,* vol. 39. pp. 2209 – 2229, 1996, http://DOI: 10.1002/(SICI)1097-0207(19960715)39:13<2209::AID-NME951>3.0.CO;2-R.

[3] S. Yu. Fialko, "Natural vibrations of complex bodies," *Int. Applied Mechanics,* vol. 40, no. 1, pp. 83 – 90, 2004, http://DOI:10.1023/B:INAM.0000023814.13805.34.

[4] S. Fialko, "Aggregation Multilevel Iterative Solver for Analysis of Large-Scale Finite Element Problems of Structural Mechanics: Linear Statics and Natural Vibrations", in *PPAM 2001*, R. Wyrzykowski et al. (Eds.), *LNCS* 2328, Springer-Verlag Berlin Heidelberg, 2002, pp. 663–670, http://DOI: 10.1007/1-4020-5370-3_41.

[5] S. Yu. Fialko, E. Z. Kriksunov and V. S. Karpilovskyy, "A block Lanczos method with spectral transformations for natural vibrations and seismic analysis of large structures in SCAD software," in *Proc. CMM-2003 – Computer Methods in Mechanics*, Gliwice, Poland, 2003, pp. 129 —130.

[6] S. Yu. Fialko, "Iterative methods for solving large-scale problems of structural mechanics using multi-core computers," Archieves of Civil and Mechanical Engineering, vol. 14, pp. 190 – 203, 2014, http:// doi:10.1016/j.acme.2013.05.009.

[7] S. Yu. Fialko, "PARFES: A method for solving finite element linear equations on multi-core computers," Advances in Engineering software, vol. 40, no. 12, pp. 1256-1265, 2010, http:// doi:10.1016/j.advengsoft.2010.09.002.

[8] G. Gambolati, G. Pini and F. Sartoretto, "An improved iterative optimization technique for the leftmost eigenpairs of large symmetric matrices," *J. Comp. Phys.,* no 74, pp. 41 – 60, 1988, http://doi: 10.1016/0021-9991(88)90067-8.

[9] C. K. Gan, P. D. Haynes and M. C. Payne, "Preconditioned conjugate gradient method for sparse generalized eigenvalue problem in electronic structure calculations," *Computer Physics Communications*, vol 134, nr. 1, pp. 33 – 40, 2001, http://DOI: 10.1016/S0010-4655(00)00188-0.

[10] V. Hernbadez, J. E. Roman, A. Tomas and V. Vidal, "A survey a software for sparse eigenvalue problems," Universitat Politecnica De Valencia, SLEPs technical report STR-6, 2009.

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

[12] A. V. Knyazev and K. Neymayr, "Efficient solution of symmetric eigenvalue problem using multigrid preconditioners in the locally optimal block conjugate gradient method," *Electronic Transactions on Numerical Analysis*, vol. 15, pp. 38 – 55, 2003.

[13] R. B. Morgan, "Preconditioning eigenvalues and some comparison of solvers," *Journal of* computational *and applied mathematics*, vol. 123, pp. 101 – 115, 2000, http://doi: 10.1016/S0377-0427(00)00395-2.

[14] M. Papadrakakis, "Solution of partial eigenproblem by iterative methods," *Int. J. Num. Meth Eng*., vol. 20. pp. 2283—2301, 1984, http://DOI: 10.1002/nme.1620201209.

[15] A. V. Perelmuter, S. Yu. Fialko, "Problems of computational mechanics relate to finite-element analysis of structural constructions," *International Journal for Computational Civil and Structural Engineering*, vol. 1, no 2, 2005, pp. 72 – 86.

[16] Y. Saad, *Numerical methods for large eigenvalue problems, Revised edition, Classics in applied mathematics*. SIAM, 2011, http://dx.-doi.org/10.1137/1.9781611970739.

[17] S. Tomov, J. Langou, A. Canning, Lin-Wang Wang, J. Dongarra, "Conjugate-gradient eigenvalue solver in computing electronic properties of nanostructure architecture," Int. J. Computational Science and Engineering, vol. 2, nr. 3-4, pp. 205 – 212, 2006.

[18] Intel Math Kernel Library Reference Manual. URL: ttps://software.intel.com/sites/products/documentation/ doclib/iss/2013/mkl/mklman/index.htm (Last access: 16.04.2015).