

# Variable-step preconditioned conjugate gradient method for partial symmetric eigenvalue problems

Maxim Larin \*      Valery Il'in †‡

## 1 Introduction

There are a variety of scientific and industrial fields, such as mathematics, physics, chemistry, computer science, engineering and medicine, where the following eigenvalue problem often arises

$$A \mathbf{u} = \lambda \mathbf{u}, \quad A = A^T \in \mathbf{R}^{n \times n}, \quad (1)$$

in which  $A$  is a large sparse symmetric positive definite matrix,  $\lambda$  is an eigenvalue and  $\mathbf{u}$  is a corresponding eigenvector. The evaluation of one or more smallest eigenpairs has much practical interest for describing the characteristics of physical phenomena. For example, smallest eigenvalues characterize the base frequencies of vibrating mechanical structures.

Typically, the matrix  $A$  is a discretization matrix, arising as a result of finite-difference, finite element or finite volume discretization of elliptic boundary value problems with self-adjoint differential operators on a mesh with a characteristic meshsize  $h$ , which has  $n$  real positive eigenvalues

$$0 < \lambda_{min} = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n = \lambda_{max}, \quad (2)$$

and their corresponding eigenvectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$ . Usually,  $h = O(n^{-1/d})$ , where  $d = 2$  or  $3$  is a spatial dimension. Moreover, the matrix  $A$  is ill-conditioned, i.e, its condition number defined by

$$\kappa(A) = \frac{\lambda_n}{\lambda_1}, \quad (3)$$

goes to infinity as  $O(h^{-2})$  when  $h$  tends to zero.

In recent years, considerable effort has been devoted to the development of efficient and reliable methods for solving matrix eigenvalue problems, and a large number of algorithms are available which have been successfully applied to a wide range of practical problems. Among these methods, the subspace iteration method and the Lanczos method are considered to be the most efficient [27]. However, the successful application of those methods are based on the exact computation of the inverse of  $A$ , i.e.,  $A^{-1}$ , which for large scale problems give rise to two main problems. In spite of the huge development in the last years for direct sparse matrix methods (permutation, symbolic factorization, elimination tree, etc.) the storage requirements are still extremely high, and may not be affordable even by modern supercomputers. The second problem is a sequential nature of direct methods, and as a result, it is very difficult to obtain a high speed-up in a parallel environment. Up to date the efficient parallelization of direct algorithms still remains a challenging task.

Another possibility to solve (1) is to transform the eigenvalue problem into an optimization one for the Rayleigh quotient

$$\lambda(\mathbf{u}) = \frac{(A \mathbf{u}, \mathbf{u})}{(\mathbf{u}, \mathbf{u})}, \quad (4)$$

---

\*Lehrstuhl für Numerische Mathematik und Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, D-52056 Aachen, Germany, E-mail:larin@igpm.rwth-aachen.de .

†Institute of Computational Mathematics and Mathematical Geophysics, Siberian Division of Russian Academy of Sciences, Lavrentiev ave. 6, 630090 Novosibirsk, Russia, E-mail: ilin@sscc.ru .

‡This work is supported partially by RFBR grant N 02-01-01176

the minimum of which  $\lambda(\mathbf{v})$  is equal to the smallest eigenvalue  $\lambda_1$  and the corresponding vector  $\mathbf{v}$  coincides with the corresponding eigenvector  $\mathbf{u}_1$ . Here,  $(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v}$  is the scalar product, which are induced the corresponding Euclidian norm  $\|\mathbf{u}\| = (\mathbf{u}, \mathbf{u})^{1/2}$ .

During the period 50's to 70's several gradient methods were suggested to find a smallest eigenpair based on minimization of the Rayleigh quotient along its gradient

$$\nabla \lambda(\mathbf{u}) = \frac{2}{(\mathbf{u}, \mathbf{u})} \cdot (A \mathbf{u} - \lambda(\mathbf{u})\mathbf{u}). \quad (5)$$

The most attractive characteristic of these methods is that the iterative process requires only vector-vector and matrix-vector multiplications, which are highly suitable for being parallelized or vectorized. Moreover, these methods can easily handle large scale problems due to their much reduced storage requirements. Unfortunately, all of these methods suffer a very poor convergence properties for ill-conditioned matrices, since the rate of convergence of iterative optimization process depends on the ratio

$$\eta = \frac{\lambda_2 - \lambda_1}{\lambda_n - \lambda_1}, \quad (6)$$

which is small because of large denominator, see for details [12, 13].

On the other hand, there are a lot of efficient preconditioned iterative methods for solving the linear system of equations with discretization matrices. In particular, algebraic multigrid [32] and incomplete factorization methods [1, 7] allow us to construct optimal or nearly optimal methods, i.e., their rate of convergence does not depend on  $n$  or, the same,  $h$  and the total computational complexity is proportional to  $n$ .

In the early 80's it was proposed to use various preconditioning techniques to accelerate the convergence rate of the iterative minimization process for partial eigenvalue problem, see [4, 8] for example. The complete survey and extensive bibliography concerning preconditioned eigensolvers one can found in [12, 13]. The well-known two-term gradient method can be written as

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} - w^{(k)} M (A \mathbf{u}^{(k)} - \lambda(\mathbf{u}^{(k)})\mathbf{u}^{(k)}), \quad (7)$$

where  $M$  is a preconditioner, which approximates the matrix  $A^{-1}$ , and  $w^{(k)}$  is a scalar value computed either from minimization condition for the Rayleigh quotient (4) on the two-dimensional subspace spanned on last iterate  $\mathbf{u}^{(k)}$  and its gradient  $M (A \mathbf{u}^{(k)} - \lambda(\mathbf{u}^{(k)})\mathbf{u}^{(k)})$  or defined as  $w^{(k)} = 1$ . However, the theoretical results given in these works was not so attractive as the corresponding numerical results, since the theoretical estimates of the rate of convergence was still based on (6).

Recently Neymeyr [21, 22, 23] considered the preconditioned gradient method as the standard inverse iteration method, in which the linear system of equation

$$A \mathbf{u}^{(k+1)} = \lambda(\mathbf{u}^{(k)})\mathbf{u}^{(k)} \quad (8)$$

was solved approximately by

$$\mathbf{u}^{(k+1)} = \lambda(\mathbf{u}^{(k)})M \mathbf{u}^{(k)} \quad (9)$$

where the preconditioner  $M$  is such that

$$\|I - MA\|_A \leq \gamma < 1. \quad (10)$$

Here  $\|\cdot\|_A$  denote the energy norm induced by the energy scalar product  $(\mathbf{u}, \mathbf{v})_A = (A\mathbf{u}, \mathbf{v})$ . Using the geometrical interpretation of the method he obtains the following sharp convergence estimate: if  $\lambda = \lambda(\mathbf{u}^{(k)}) \in [\lambda_1, \lambda_2]$ , then

$$\lambda(\mathbf{u}^{(k+1)}) \leq \lambda_{1,2} \quad (11)$$

where

$$\begin{aligned} \lambda_{1,2} &= \lambda \lambda_1 \lambda_2 (\lambda_1 + \lambda_2 - \lambda)^2 \cdot \left( \gamma^2 (\lambda_2 - \lambda) (\lambda - \lambda_1) (\lambda \lambda_2 + \lambda \lambda_1 - \lambda_1^2 - \lambda_2^2) \right. \\ &\quad \left. - 2\gamma \sqrt{\lambda_1 \lambda_2} (\lambda_2 - \lambda) (\lambda - \lambda_1) \sqrt{\lambda_1 \lambda_2 + (1 - \gamma^2) (\lambda_2 - \lambda) (\lambda - \lambda_1)} \right. \\ &\quad \left. - \lambda (\lambda_1 + \lambda_2 - \lambda) (\lambda \lambda_2 + \lambda \lambda_1 - \lambda_1^2 - \lambda_1 \lambda_2 - \lambda_2^2) \right)^{-1} \end{aligned}$$

which is too complicated for understanding main convergence-depended factors. Likely this sharp estimate has another representation, which has been done in [14] and which we will use here (see Theorem 3).

It is well-known that the subspace iteration method for computing  $m$  smallest eigenpairs described in Section 3 is a straightforward generalization of the inverse iteration method for the first eigenpair, from which the preconditioned subspace iteration version of the method is naturally followed. Unfortunately, the rate of convergence, given in [15], for the block version of the preconditioned method does not depend on the number of vectors used as it is for the original subspace iteration method [27].

In the present work we continue the study of optimal eigensolvers, which was began in [18], and define a variable-step preconditioning methods for solving the partial eigenvalue problem (1), see Section 4. In the role of the corresponding variable-step preconditioner we consider two different nonlinear solvers, see Section 5. The first one is based on the iterative incomplete factorization technique, whereas the second one is constructed by the algebraic multigrid method applied to the matrix  $A$ . The numerical results in Section 6 show that both methods are robust and efficient, but the algebraic multigrid method has naturally the better time performance due to its optimal nature for elliptic boundary value problems [31]. Some interesting remarks about the convergence behaviour of the preconditioned eigensolver are given in Appendix A and B. Finally, we compare our preconditioned gradient eigensolver with the direct multigrid eigensolver from [19, 20] and give some conclusions.

## 2 Problem formulation

Consider the elliptic eigenvalue problem

$$-\sum_{i,j=1}^d \frac{\partial}{\partial x_i} \left( a_{ij}(x) \frac{\partial}{\partial x_j} u_k(x) \right) = \lambda_k u_k(x), \quad x \in \Omega, \quad (12)$$

$$u_k(x) = 0, \quad x \in \partial\Omega, \quad k = 1, \dots, p,$$

where  $\Omega$  is a polygonal domain in  $R^d$  with boundary  $\partial\Omega$ , where  $d = 2$  or  $3$  is a spatial dimension, the coefficient matrix  $[a_{ij}(x)]$  is assumed to be uniformly symmetric positive definite for any  $x \in \bar{\Omega}$  and  $p$  is given integer.

The Galerkin variational formulation of the boundary value problem (12) is as follows.

Find

$$u_k \in H_0^1(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}$$

such that

$$\begin{aligned} a(u_k, v) &= \lambda_k (u_k, v), & k &= 1, \dots, p, \\ a(u_k, u_l) &= \delta_{kl}, & l &= 1, \dots, p, \end{aligned}$$

for all  $v \in H_0^1(\Omega)$ , where  $\delta_{kl}$  is the Kronecker symbol, the bilinear form  $a(u, v)$  and the linear functional  $(f, v)$  are defined by

$$a(u, v) = \int_{\Omega} \left[ \sum_{i,j=1}^2 a_{ij}(x) \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} \right] d\Omega, \quad (u, v) = \int_{\Omega} uv \, d\Omega.$$

Let us assume that the triangulation  $\Upsilon$  of the domain  $\Omega$  is such that the coefficient matrix  $[a_{ij}(x)]$  is constant in each element  $T \in \Upsilon$  and can be discontinuous across the elements. Denoting by  $V \subset H^1(\Omega)$  the space of continuous functions, which are linear on each element from  $\Upsilon$ , we obtain the partial eigenvalue problem

$$\begin{aligned} A \mathbf{u}_k &= \lambda_k \mathbf{u}_k, & A &= A^T \in \mathbf{R}^{n \times n}, & 0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_p, \\ (\mathbf{u}_k, \mathbf{u}_l) &= \delta_{kl}, & k, l &= 1, \dots, p \ll n, \end{aligned}$$

where  $A$  is a global stiffness matrix, corresponding to standard piecewise linear functions in  $V$ , which is calculated as

$$A = \{a(\phi_i, \phi_j)\}_{i,j=1}^n,$$

and  $\{\phi_i\}_{i=1}^n$  is a set of standard Lagrangian (nodal) basis functions in  $V$ .

### 3 The subspace iteration

One of the classical methods for computing the smallest eigenvalue and its eigenvector is the inverse iteration method [27], which generates a sequence of iterates  $\mathbf{v}^{(i)}$  by solving the linear system of equations

$$A\mathbf{v}^{(i+1)} = \mathbf{v}^{(i)}. \quad (13)$$

In practice, the iterates are normalized after each step. Such subspace iteration algorithm can be described as follows.

#### Algorithm SUBIT

**Input:**  $m$  starting vectors  $\mathbf{v}_1^{(0)}, \dots, \mathbf{v}_m^{(0)}$

**Devices:** to compute  $A\mathbf{v}$  for a given vector  $\mathbf{v}$   
to compute the scalar product  $(\mathbf{v}, \mathbf{u})$  for given vectors  $\mathbf{v}$  and  $\mathbf{u}$   
to compute the Rayleigh-Ritz projection for a given subspace

**Method:** **for**  $i = 0, 1, \dots$ , untill convergence  
    **for**  $j = 1, \dots, m$   
         $\mathbf{w}_j^{(i+1)} = A^{-1}\mathbf{v}_j^{(i)}$   
    **end for**  
    Apply the Rayleigh-Ritz method on the subspace  
     $Span\{\mathbf{w}_1^{(i+1)}, \dots, \mathbf{w}_m^{(i+1)}\}$  and set  $\mathbf{v}_j^{(i+1)}$ ,  $j = 1, \dots, m$ ,  
    corresponds to the new Ritz vectors  
    **end for**

**Output:** the approximations  $\mu_j^{(i)}$  and  $\mathbf{v}_j^{(i)}$  to the smallest eigenvalues  $\lambda_j$   
and corresponding eigenvectors  $\mathbf{u}_j$ ,  $j = 1, \dots, m$ .

Usually, one can simply extend any eigensolver for computing a single eigenpair to a subspace iteration in order to determine a number of eigenvalues and their corresponding eigenvectors. One can implement a subspace eigensolver by applying a given vector iteration to each vector of an actual eigenspace approximation. Subsequent application of the Rayleigh-Ritz projection computes the new eigenvalues and eigenvectors. The next theorems shows the classical convergence result for the subspace iteration.

**Theorem 1** [27] *Let  $U = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m]$  be the matrix of wanted eigenvectors, and hence,  $\mathcal{U} = \text{span } U$  is the dominant invariant subspace under  $A^{-1}$ . Let  $\mathcal{V}$  be any  $m$ -dimensional subspace of  $\mathbb{R}^n$  and  $\{A^{-i}\mathcal{V} : i = 0, 1, 2, \dots\}$  the associated sequence generated by the subspace iteration. Moreover, we define the quantities  $\psi_j^{(i)}$  by*

$$\psi_j^{(i)} = \angle(\mathbf{u}_j, A^{-i}\mathcal{V}) = \min_{\mathbf{x}_j^{(i)} \in A^{-i}\mathcal{V}} \angle(\mathbf{u}_j, \mathbf{x}_j^{(i)}).$$

Then each eigenvector  $\mathbf{u}_j$ ,  $j \leq m$ , satisfies

$$\tan \psi_j^{(i)} \leq \left( \frac{\lambda_j}{\lambda_{m+1}} \right)^i \tan \angle(\mathcal{U}, \mathcal{V})$$

Theorem 1 shows that a certain sequence  $\{\mathbf{x}_j^{(i)}\} \in A^{-i}\mathcal{V}$  converges to eigenvectors  $\mathbf{u}_j$  when  $i \rightarrow \infty$ . However, it does not address the behavior of the sequence  $\{\mathbf{v}_j^{(i)}\}$  actually computed by Algorithm SUBIT. Really, due to the optimality of  $\mathbf{x}_j^{(i)}$  the approximations  $\mathbf{v}_j^{(i)}$  converge slowly than  $\mathbf{x}_j^{(i)}$ . Likely, the following Theorem 2 shows that  $\mathbf{v}_j^{(i)}$  converge to  $\mathbf{x}_j^{(i)}$  at the same asymptotic rate as  $\mathbf{x}_j^{(i)}$  to  $\mathbf{u}_j$ , and hence, the same convergence factor  $(\lambda_j/\lambda_{m+1})^i$  governs the linear convergence of approximated eigenvectors  $\mathbf{v}_j^{(i)}$  to the actual eigenvectors  $\mathbf{u}_j$ .

**Theorem 2** [27] *When subspace iteration uses Algorithm SUBIT then each Ritz vector  $\mathbf{v}_i^{(k)}$  is related to the vector  $\mathbf{x}_i^{(k)}$  of Theorem 1 as  $k \rightarrow \infty$ , by*

$$\sin \angle(\mathbf{v}_i^{(k)}, \mathbf{x}_i^{(k)}) = O \left[ \left( \frac{\lambda_i}{\lambda_{m+1}} \right)^k \right], \quad i = 1, 2, \dots, m.$$

On the other hand, one can measure the deviation of the subspace  $\mathcal{V}$  and  $\mathcal{U}$  from Theorem 1 through the classical definition of the distance between equidimensional subspaces, see [10], for example. Using this definition the number of interesting results has been obtained by D'yakonov, Knyazev et al. [4, 5, 6, 11, 12, 13] for symmetric positive definite matrices under the natural assumption on the preconditioned matrix  $MA$ , i.e., there exist two positive constants  $c_0$  and  $c_1$  such that

$$0 < c_0(M^{-1}\mathbf{v}, \mathbf{v}) \leq (A\mathbf{v}, \mathbf{v}) \leq c_1(M^{-1}\mathbf{v}, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbb{R}^n.$$

The main convergence result for SUBIT method can be written as follows

$$\frac{\lambda_j^{(i+1)} - \lambda_j}{\lambda_{j+1} - \lambda_j^{(i+1)}} \leq (1 - \xi)^i \frac{\lambda_j^{(0)} - \lambda_j}{\lambda_{j+1} - \lambda_j^{(0)}}, \quad \xi = \frac{c_0}{c_1} \cdot \frac{\lambda_n}{\lambda_{m+1}} \cdot \frac{\lambda_{m+1} - \lambda_j}{\lambda_n - \lambda_j}.$$

Clearly, the present estimate is not as sharp as one in Theorems 1 and 2 and moreover, depends on the meshsize  $h$  as  $\lambda_n/\lambda_{m+1}$ .

Finally, for the completeness of the presentation we give a short description of the Rayleigh-Ritz projection. It is well-known that Ritz projection computes the best set of approximated eigenvectors from a subspace to eigenvectors of the original matrix  $A$  [27]. Here we use them to find all eigenvectors from the subspace spanned onto computed approximations  $\mathbf{w}_1^{(k)}, \dots, \mathbf{w}_m^{(k)}$ . As a result we get a number of orthonormal approximated eigenvectors and their eigenvalues. Here we present only the basic steps of the method

### Algorithm RITZ

- Input:**  $p$  starting vectors  $\hat{\mathbf{v}}_1, \dots, \hat{\mathbf{v}}_p$
- Devices:** to compute  $A\mathbf{v}$  for a given vector  $\mathbf{v}$   
to compute the scalar product  $(\mathbf{v}, \mathbf{u})$  for given vectors  $\mathbf{v}$  and  $\mathbf{u}$   
to compute the orthonormal basis for given vectors  
to compute the eigenvalues and eigenvectors for a given matrix
- Method:**
- 1 Orthonormalize the vectors  $\{\hat{\mathbf{v}}_i\}_{i=1}^p$  and form  $n$ -by- $p$  matrix  $W = [\mathbf{w}_1^{(k)}, \dots, \mathbf{w}_p^{(k)}]$  from new orthonormal vectors  $\{\mathbf{w}_i^{(k)}\}_{i=1}^p$
  - 2 Compute scalar products  $h_{ij} = (\mathbf{w}_j^{(k)}, A\mathbf{w}_i^{(k)})$  and form  $p$ -by- $p$  matrix  $H = \{h_{ij}\} = W^T A W$
  - 3 Compute all eigenpairs of  $H$ :  $H\mathbf{u}_i = \mu_i \mathbf{u}_i$ ,  $i = 1, \dots, p$
  - 4 Compute all Ritz vectors  $\mathbf{v}_i = W\mathbf{u}_i$ ,  $i = 1, \dots, p$ .
- Output:** the approximations  $\mu_j$  and  $\mathbf{v}_j$  to the smallest eigenvalues  $\lambda_j$  and corresponding eigenvectors  $\mathbf{u}_j$ ,  $j = 1, \dots, p$ .

## 4 The LOBPCG method with variable-step preconditioner

One of the most robust preconditioned eigensolver is the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method, which has been suggested and analyzed by Knyazev [13]. The idea of the method for computing the first eigenvalue based on the local optimization of the three-term recurrence by the Rayleigh-Ritz method on a three-dimensional subspace consisting of the previous iterate  $\mathbf{v}^{(i-1)}$ , the current iterate  $\mathbf{v}^{(i)}$  and the preconditioned residual  $\mathbf{w}^{(i)}$ . The generalization to the block version is straightforward.

In contrast to the two-term gradient method (7) the LOBPCG method converges much faster, but still now there is no corresponding theoretical results. Nevertheless, since the Rayleigh quotient for the LOBPCG method minimizing on the extended subspace with respect to the preconditioned inverse iteration method (9), then we can use the results for two-term gradient methods for estimating the convergence behaviour of the LOBPCG method.

**Theorem 3** [14] *Let us assume that*

$$\|I - MA\|_A \leq \gamma < 1,$$

*then for a fixed index  $j \in [1, m]$ , if  $\lambda_j^{(i)} \in [\lambda_{k_j}, \lambda_{k_j+1}[$  then it holds for the Ritz value  $\lambda_j^{(i+1)}$  computed by the LOBPCG algorithm that either  $\lambda_j^{(i+1)} < \lambda_{k_j}$  (unless  $k_j = j$ ), or  $\lambda_j^{(i+1)} \in [\lambda_{k_j}, \lambda_j^{(i)}[$ . In the latter case,*

$$\frac{\lambda_j^{(i+1)} - \lambda_{k_j}}{\lambda_{k_j+1} - \lambda_j^{(i+1)}} \leq (q(\gamma, \lambda_{k_j}, \lambda_{k_j+1}))^2 \frac{\lambda_j^{(i)} - \lambda_{k_j}}{\lambda_{k_j+1} - \lambda_j^{(i)}}$$

where

$$q(\gamma, \lambda_{k_j}, \lambda_{k_j+1}) = \gamma + (1 - \gamma) \frac{\lambda_{k_j}}{\lambda_{k_j+1}}.$$

The main advantage of the present estimate is in that the quantity  $q(\gamma, \lambda_{k_j}, \lambda_{k_j+1})$  does not depend on  $m$  and  $n$ . Thus, Theorem 3 shows an optimal rate of convergence, but only for a certain initial approximation  $\mathbf{x}_j^{(i)}$  for which the following conditions are fulfilled

$$\lambda_j^{(i)} \in [\lambda_j, \lambda_{j+1}[, \quad \lambda_j^{(i+1)} \in [\lambda_j, \lambda_{j+1}[, \quad j = 1, \dots, m.$$

However, these conditions hold only if  $\mathbf{v}_j^{(i)}, j = 1, \dots, m$  are good approximations to  $\mathbf{u}_j, j = 1, \dots, m$ . And there lies the main difficulty because the above conditions demands a good approximation of  $\mathbf{u}_j$  by  $\mathbf{v}_j^{(i)}$  and that is not allowed in a priori analysis. The remedy is to take two estimates from Theorems 1, 2 and 3 and then try to derive modified bounds of the similar form with the quantity  $q$  depending on  $m$ . Unfortunately, it is still an open question.

Here we have to mention that this formulation is quite general and can be used for an arbitrary preconditioning  $M$ . In practice, the preconditioner  $M$  has to be chosen such that the action of the nonlinear operator  $AM$  is in a sense close to the identity operator.

Let  $M$  be a nonlinear mapping  $R^N$  onto  $R^N$ , the action of which depends on a vector  $v \in R^N$ , then the basic scheme of the block variable-step LOBPCG-VS method can be described as follows.

### Algorithm LOBPCG-VS

**Input:**  $m$  starting vectors  $\mathbf{v}_1^{(0)}, \dots, \mathbf{v}_m^{(0)}$

**Devices:** to compute  $A\mathbf{v}$  and  $M\mathbf{v}$  for a given vector  $\mathbf{v}$   
to compute the scalar product  $(\mathbf{v}, \mathbf{u})$  for given vectors  $\mathbf{v}$  and  $\mathbf{u}$   
to compute the Rayleigh-Ritz projection for a given subspace

**Method:** **for**  $i = 0, 1, \dots$ , until convergence  
**for**  $j = 1, \dots, m$   
 $\mu_j^{(i)} = (A\mathbf{v}_j^{(i)}, \mathbf{v}_j^{(i)}) / (\mathbf{v}_j^{(i)}, \mathbf{v}_j^{(i)})$   
 $\mathbf{r}_j^{(i)} = \mu_j^{(i)} \mathbf{v}_j^{(i)} - A\mathbf{v}_j^{(i)}$   
 $\mathbf{w}_j^{(i)} = M\mathbf{r}_j^{(i)}$   
**end**  
Use the Rayleigh-Ritz method on the trial subspace  
 $Span\{\mathbf{w}_1^{(i)}, \dots, \mathbf{w}_m^{(i)}, \mathbf{v}_1^{(i)}, \dots, \mathbf{v}_m^{(i)}, \mathbf{v}_1^{(i-1)}, \dots, \mathbf{v}_m^{(i-1)}\}$   
set  $\mathbf{v}_j^{(i+1)}$  corresponds to  $j$ -th smallest Ritz vector,  $j = 1, \dots, m$   
**end**

**Output:** the approximations  $\mu_j^{(i)}$  and  $\mathbf{v}_j^{(i)}$  to the smallest eigenvalues  $\lambda_j$   
and corresponding eigenvectors  $\mathbf{u}_j$ ,  $j = 1, \dots, m$ .

Here we have to note that in [13] the mathematically equivalent, but more numerically stable version of the LOBPCG method has been proposed. The latter was implemented in our code [18].

## 5 Variable-step preconditioner $M$

The variable-step preconditioner  $M$  to matrix  $A$  defined as follows:

Apply an iterative method to solve the linear system of equations

$$A\mathbf{x} = \mathbf{b} \quad (14)$$

until the following stopping criteria

$$\frac{\|\mathbf{r}^{(\nu)}\|}{\|\mathbf{r}^{(0)}\|} \leq \varepsilon_{in}, \quad \mathbf{r}^{(\nu)} = \mathbf{b} - A\mathbf{x}^{(\nu)}, \quad (15)$$

is satisfied. Hence, we define

$$M\mathbf{b} = \mathbf{x}^{(\nu)}, \quad (16)$$

where  $\mathbf{x}^{(\nu)}$  is the  $\nu$ -th iterate of this method, beginning with  $\mathbf{x}^{(0)} = 0$ .

There are a lot of methods for solving a linear system of equations with optimal or nearly optimal computational complexity. In the present study we use iterative incomplete factorization method [7, 17, 24, 25, 26] and algebraic multigrid method [30, 29, 28, 32] as the inner iterative process to construct the variable-step preconditioner.

Due to the classical theoretical results one can consider the first method as a nearly optimal variable-step preconditioner to the matrix  $A$ , whereas the second one as an optimal one. Indeed, to provide a given accuracy  $\varepsilon_{in} \ll 1$  for solution of (14) by the incomplete factorization method, the number of iterations is proportional to  $N^{5/4}$  and to  $N^{7/6}$  for two- and three-dimensional model boundary value problems, respectively. On the other hand, the convergence rate of AMG methods depends on the smoothing and interpolation properties rather than on the order of the system to be solved.

It is well-known that applying few steps of the conjugate gradient method for solving a linear system of equations we produce a matrix polynomial on the original matrix  $A$ , the coefficients of

which depend on the initial guess and the right-hand side. This polynomial can be represented as a symmetric positive definite matrix, if the original matrix is symmetric and positive definite. Hence, by the definition of solvers for ICF and AMG methods on each iteration we have a symmetric positive definite matrix, which depends on the right-hand side  $\mathbf{b}$  and a zero initial approximation  $\mathbf{x}^{(0)}$ . Thus, on each outer iteration of the LOBPCG-VS method we generate a new preconditioning matrix  $M^{(k)}$ , and hence, the nonlinear operator  $M$  can be considered as a sequence of symmetric positive definite matrices  $\{M^{(k)}\}$ . Unfortunately, we could not use the symmetry and positive definiteness of  $MA$  on each iteration to improve the rate of convergence of the LOBPCG-VS, but it's good to know that it is.

It is well-known that theoretical estimates for the convergence rate of iterative methods for solving a linear system of equations and eigenproblems for general positive definite matrices depend on the energy norm of the matrix  $A$ . However, it is indeed a very strong condition on the matrix  $A$ . Since in all of those methods we really need an upper estimate for a certain scalar product through a corresponding matrix norm on the Krylov subspace only, see [1], for example, which is usually replaced (for convenience of the analysis) by the norm over all space  $\mathbb{R}^n$ . Thus if one shows that the norm of the matrix is bounded for all possible Krylov subspaces, then one can use all well-known results for convergence rate of the iterative methods. Below we will use it to prove the rate of convergence of the LOBPCG-VS method.

**Theorem 4** *The variable-step preconditioned LOBPCG-VS method converges for a fixed index  $j \in [1, m]$  as*

$$\frac{\lambda_j^{(i+1)} - \lambda_{k_j}}{\lambda_{k_j+1} - \lambda_j^{(i+1)}} \leq (q(\varepsilon_{in}, \lambda_{k_j}, \lambda_{k_j+1}))^2 \frac{\lambda_j^{(i)} - \lambda_{k_j}}{\lambda_{k_j+1} - \lambda_j^{(i)}}$$

where  $\lambda_j^{(i)} \in [\lambda_{k_j}, \lambda_{k_j+1}[$  and  $\lambda_j^{(i+1)} \in [\lambda_{k_j}, \lambda_{k_j+1}[$  are the old and new eigenvalue approximation, respectively,  $\lambda_j^{(i+1)}$  computed by above algorithm (14)-(16) and

$$q(\varepsilon_{in}, \lambda_{k_j}, \lambda_{k_j+1}) = \varepsilon_{in} + (1 - \varepsilon_{in}) \frac{\lambda_{k_j}}{\lambda_{k_j+1}}.$$

*Proof* Denote by  $\mathbf{e}^{(0)} = \mathbf{x} - \mathbf{x}^{(0)}$  and  $\mathbf{e}^{(VS)} = \mathbf{x} - \mathbf{x}^{(\nu)}$  the initial and final error for the inner solver, respectively. From the definition of the stopping criteria (15) we have

$$\frac{\|\mathbf{r}^{(\nu)}\|}{\|\mathbf{r}^{(0)}\|} = \frac{\|A\mathbf{x}^{(\nu)} - \mathbf{b}\|}{\|A\mathbf{x}^{(0)} - \mathbf{b}\|} = \frac{\|A(\mathbf{x}^{(\nu)} - \mathbf{x})\|}{\|A(\mathbf{x}^{(0)} - \mathbf{x})\|} = \frac{\|\mathbf{e}^{(VS)}\|_A}{\|\mathbf{e}^{(0)}\|_A} \leq \varepsilon_{in}$$

On the other side, using the choice of the initial guess and the definition of the variable-step preconditioner we obtain

$$\mathbf{x}^{(\nu)} = M\mathbf{b} \Leftrightarrow \mathbf{x} - \mathbf{x}^{(\nu)} = \mathbf{x} - MA\mathbf{x} \Leftrightarrow \mathbf{e}^{(VS)} = (I - MA)\mathbf{e}^{(0)}.$$

Thus, due to the arbitrariness of  $\mathbf{e}^{(0)}$  we derive the following estimate on the actual Krylov space

$$\|I - MA\|_A \leq \varepsilon_{in},$$

from which using Theorem 3 we prove the desired result.  $\square$

**Corollary 1** *Note that if  $\varepsilon_{in}$  tends to 0, then  $MA$  is close to an identity matrix and hence,*

$$q(\varepsilon_{in}, \lambda_{k_j}, \lambda_{k_j+1}) \rightarrow \frac{\lambda_{k_j}}{\lambda_{k_j+1}}.$$

which is the best possible estimates for the convergence rate, since in contrast to results from Theorems 1 and 2 it does not depend on the characteristic meshsize  $h$  and the number of vector used  $m$ . However, this estimate is true only if each approximated eigenvector  $\mathbf{v}_i$  is a good approximation to the corresponding real eigenvector  $\mathbf{u}_i$ . However, it is not true! Thus, in the best possible situation we have an optimal rate of convergence whereas in the poorest case the rate of convergence depend on  $m$ , see Theorem 1.

## 6 Numerical results

All numerical results in this section can be divided into two parts. First of all, we present the numerical results for finding the smallest eigenpair  $\{\lambda_1, \mathbf{u}_1\}$ , The second part of experiments shows efficiency and robustness of the method for computing the smallest eigenspace for various number of vectors.

As a model problem, which has the exact (analytical) solution, we consider the Laplace operator on square (cube) domain with homogeneous Dirichlet boundary conditions, i.e., we solve the following eigenvalue problem

$$A \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad \|\mathbf{v}_i\| = 1, \quad \lambda_i > 0, \quad i = 1, 2, \dots, p \leq m, \quad (17)$$

which corresponds to the piecewise-linear finite-element discretization of the two (three) dimensional second-order elliptic problems

$$-\left[ a_{11} \frac{\partial^2 u}{\partial x^2} + a_{22} \frac{\partial^2 u}{\partial y^2} \left( + a_{33} \frac{\partial^2 u}{\partial z^2} \right) \right] = \lambda u, \quad \text{in } \Omega, \quad (18)$$

$$\|u\| = 1, \quad u = 0 \quad \text{on } \Gamma_D = \partial\Omega,$$

**Table 1.** Number of iterations for LOBPCG-VS method, 2D case,  $m = 1$ ,  $p = 1$

N (2D case)	4	8	16	32	64	128	256
with DIAG preconditioner							
$a_{11} = 1, a_{22} = 1$	8	20	40	74	133	233	418
$a_{11} = 1, a_{22} = 10^{-1}$	10	41	74	138	238	436	624
$a_{11} = 1, a_{22} = 10^{-2}$	33	53	160	343	486	674	1290
$a_{11} = 1, a_{22} = 10^{-3}$	48	59	164	393	904	1924	3597
with ICF preconditioner							
$a_{11} = 1, a_{22} = 1$	4	6	6	5	5	4	4
$a_{11} = 1, a_{22} = 10^{-1}$	7	10	8	8	7	7	5
$a_{11} = 1, a_{22} = 10^{-2}$	7	15	19	18	11	10	10
$a_{11} = 1, a_{22} = 10^{-3}$	7	21	29	38	26	26	26
with AMG preconditioner							
$a_{11} = 1, a_{22} = 1$	3	6	6	5	5	4	4
$a_{11} = 1, a_{22} = 10^{-1}$	3	7	8	8	7	7	5
$a_{11} = 1, a_{22} = 10^{-2}$	3	7	15	18	11	10	10
$a_{11} = 1, a_{22} = 10^{-3}$	3	7	15	31	26	26	24

in the square (cube) domain  $\Omega = [0, 1] \times [0, 1] (\times [0, 1])$  on a uniform Cartesian mesh  $\mathcal{T}_h$  with stepsize  $h = N^{-1}$ , by the above method beginning with the random initial guess  $x^{(0)}$  and continue the iterative process until the following stopping criterion

$$\frac{\delta_k}{\delta_0} < \varepsilon_{out} = 10^{-6}, \quad \delta_k = \max_{1 \leq j \leq p} \|\mathbf{r}_j^{(k)}\|,$$

was satisfied. Here  $p$  is a number of desired eigenvectors and  $m$  is the number of test vectors used. Moreover, the preliminary tests and Corollary 1 shown that the rate of convergence is weakly depend on the inner stopping criteria when  $\varepsilon_{in}$  goes to zero, but at the same time the corresponding computational costs are considerable increased. Thus, in what follows we use  $\varepsilon_{in} = 10^{-1}$ .

In all Tables of this section DIAG denotes the LOBPCG method with the diagonal preconditioner, i.e.,  $M = \text{diag}(A)$ , ICF and AMG denotes the LOBPCG-VS method with incomplete factorization and algebraic multigrid preconditioner, respectively.

**Table 2.** Number of iterations for LOBPCG-VS method, 3D case,  $m = 1$ ,  $p = 1$

N (3D case)	4	8	16	32	64
with DIAG preconditioner					
$a_{11} = 1, a_{22} = 1, a_{33} = 1$	12	25	49	87	160
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-1}$	23	55	99	139	244
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-2}$	25	89	233	349	499
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-3}$	19	86	247	557	1583
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-1}$	23	48	84	124	232
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-2}$	40	114	208	286	504
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-3}$	39	170	437	591	1094
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-2}$	27	83	196	322	610
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-3}$	37	155	395	591	1126
$a_{11} = 1, a_{22} = 10^{-3}, a_{33} = 10^{-3}$	19	79	243	570	1233
with ICF preconditioner					
$a_{11} = 1, a_{22} = 1, a_{33} = 1$	6	7	6	6	5
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-1}$	12	12	10	8	7
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-2}$	12	19	24	18	14
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-3}$	11	25	32	34	32
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-1}$	11	11	9	7	7
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-2}$	18	22	20	14	12
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-3}$	22	35	44	31	29
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-2}$	14	22	22	17	15
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-3}$	25	46	50	32	29
$a_{11} = 1, a_{22} = 10^{-3}, a_{33} = 10^{-3}$	13	28	43	38	35
with AMG preconditioner					
$a_{11} = 1, a_{22} = 1, a_{33} = 1$	5	7	6	6	-
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-1}$	5	12	10	8	-
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-2}$	5	18	24	18	-
$a_{11} = 1, a_{22} = 1, a_{33} = 10^{-3}$	5	18	32	34	-
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-1}$	5	11	9	7	-
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-2}$	5	18	20	14	-
$a_{11} = 1, a_{22} = 10^{-1}, a_{33} = 10^{-3}$	5	18	44	31	-
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-2}$	5	18	22	17	-
$a_{11} = 1, a_{22} = 10^{-2}, a_{33} = 10^{-3}$	5	18	50	32	-
$a_{11} = 1, a_{22} = 10^{-3}, a_{33} = 10^{-3}$	5	18	43	38	-

As it is readily seen from the results in Tables 1 and 2 the number of iterations for LOBPCG-VS method with the diagonal preconditioner is crucially depend on two factors: the anisotropy ratio and order of the system to be solved. It is not a surprise for us, and by these results we only want to show how serious is the problem, which we would like to solve by preconditioning. On the other side the number of iterations for LOBPCG-VS method with another two preconditioners does not depend on  $N$  or even decrease (for details see Appendix A) and only slightly depend on the anisotropy ratio  $\max_{i,j} a_{ii}/a_{jj}$  (see Appendix B).

The number of iterations for LOBPCG-VS method with AMG preconditioner is similar to one with ICF preconditioner. This proof the optimal property of the suggested nonlinear eigensolver, i.e. the number of iteration depends only on the  $\varepsilon_{in}$ . The large number of iterations in the case of diagonal preconditioner for the same  $\varepsilon_{in}$  is caused by the slow convergence behaviour of the inner DIAG-PCG method, and hence, it is aborted by the maximal iteration criteria before the desired accuracy is reached.

The AMG is better from the eigenvalue point of view, since during solving (coarse-grid correction) add to eigenvalue approximation components of minimal eigenvectors even they are not be in this one. On the other hand, LOBPCG-VS method with AMG preconditioner requires more memory than ICF preconditioner. (The last column for  $N = 64$  is out of memory!)

Since all results in Table 1 and 2 shows that the convergence behaviour dose not depend on the size of the eigenproblem to be solved in what follows we present the results only for two-dimensional case to avoid a large number of similar tables.

**Table 3.** Number of iterations for LOBPCG-VS method,  $m = 5, p = 1$

N (2D case)	4	8	16	32	64	128	256
with DIAG preconditioner							
$a_{11} = 1, a_{22} = 1$	3	13	29	56	148	211	381
$a_{11} = 1, a_{22} = 10^{-1}$	9	22	41	76	183	343	> 500
$a_{11} = 1, a_{22} = 10^{-2}$	13	25	110	190	360	> 500	> 500
$a_{11} = 1, a_{22} = 10^{-3}$	19	31	287	> 500	> 500	> 500	> 500
with ICF preconditioner							
$a_{11} = 1, a_{22} = 1$	2	7	7	7	7	6	6
$a_{11} = 1, a_{22} = 10^{-1}$	2	7	7	7	6	7	6
$a_{11} = 1, a_{22} = 10^{-2}$	2	7	12	11	11	10	10
$a_{11} = 1, a_{22} = 10^{-3}$	3	8	32	24	25	23	22
with AMG preconditioner							
$a_{11} = 1, a_{22} = 1$	2	6	5	4	4	6	4
$a_{11} = 1, a_{22} = 10^{-1}$	2	6	6	5	6	5	4
$a_{11} = 1, a_{22} = 10^{-2}$	2	7	11	9	8	7	7
$a_{11} = 1, a_{22} = 10^{-3}$	2	7	15	17	19	16	14

**Table 4.** Number of iterations for LOBPCG-VS method,  $m = 10, p = 1$

N (2D case)	4	8	16	32	64	128	256
with DIAG preconditioner							
$a_{11} = 1, a_{22} = 1$	3	11	23	47	124	201	358
$a_{11} = 1, a_{22} = 10^{-1}$	8	19	30	82	177	326	> 500
$a_{11} = 1, a_{22} = 10^{-2}$	10	24	59	176	324	> 500	> 500
$a_{11} = 1, a_{22} = 10^{-3}$	14	28	94	358	> 500	> 500	> 500
with ICF preconditioner							
$a_{11} = 1, a_{22} = 1$	2	6	6	6	6	6	6
$a_{11} = 1, a_{22} = 10^{-1}$	2	5	6	6	6	6	6
$a_{11} = 1, a_{22} = 10^{-2}$	2	5	8	7	7	7	7
$a_{11} = 1, a_{22} = 10^{-3}$	2	6	10	14	16	15	14
with AMG preconditioner							
$a_{11} = 1, a_{22} = 1$	2	5	4	4	4	4	4
$a_{11} = 1, a_{22} = 10^{-1}$	2	5	5	5	4	4	4
$a_{11} = 1, a_{22} = 10^{-2}$	2	5	7	7	6	6	6
$a_{11} = 1, a_{22} = 10^{-3}$	2	5	7	13	12	12	11

To illustrate the dependness of the convergence rate on the number of trial approximation used we made the corresponding tests for  $m = 5$  and  $m = 10$ . The numerical results are given in Tables 3 and 4. Now one can easily seen that using the large number of test vectors improve the convergence rate. However, the computational costs (time and memory) are increasing too, see Table 5 and 6 for Laplace operator on the fixed grid  $N = 31$ . The present results are in a good agreement with theoretical ones.

**Table 5.** Number of iterations and time for LOBPCG-VS method,  $N = 31$ 

p	1	2	3	4	5
with DIAG preconditioner					
$m = p$	27	65	74	75	58
$m = 5$	26	41	42	51	58
$m = p$	3.07	23.91	60.72	94.62	116.82
$m = 5$	45.69	72.36	79.22	97.20	116.82
with ICF preconditioner					
$m = p$	2	10	10	10	10
$m = 5$	5	5	6	7	10
$m = p$	1.37	11.69	18.74	27.24	52.77
$m = 5$	19.42	19.43	24.37	26.83	52.77
with AMG preconditioner					
$m = p$	2	4	6	6	5
$m = 5$	2	3	4	4	5
$m = p$	6.74	24.61	69.92	90.04	113.32
$m = 5$	40.28	56.17	72.33	88.16	113.32

**Table 6.** Number of iterations and time for LOBPCG-VS method,  $N = 31$ 

p	1	2	3	4	5	6	7	8	9	10
with DIAG preconditioner										
$m = p$	27	65	74	75	58	53	46	50	49	69
$m = 10$	23	23	31	31	34	35	35	51	56	69
$m = p$	3.07	23.91	60.72	94.62	116.82	135.92	170.08	231.44	297.68	490.91
$m = 10$	150.64	222.42	202.03	233.37	263.83	228.85	264.52	349.98	413.67	490.91
with ICF preconditioner										
$m = p$	2	10	10	10	10	10	10	10	10	10
$m = 10$	3	3	4	5	5	5	6	8	8	10
$m = p$	1.37	11.69	18.74	27.24	36.97	47.95	60.45	74.57	110.50	109.43
$m = 10$	32.05	32.14	42.68	52.99	52.84	52.83	63.71	86.65	86.83	109.43
with AMG preconditioner										
$m = p$	2	4	6	6	5	6	6	5	7	7
$m = 10$	2	3	3	4	4	5	5	5	5	7
$m = p$	6.74	24.61	69.92	90.04	113.32	163.65	167.95	177.49	291.89	316.26
$m = 10$	97.16	142.52	141.47	181.88	182.30	226.72	227.59	227.07	227.19	316.26

In Tables 7–9 the accuracy of the LOBPCG-VS method for different choices of trial vectors are given. In all Tables  $\lambda_i$  are computed approximations to exact eigenvalues  $\lambda_{h,i}$ , and hence, the magnitude  $|\lambda_{h,i} - \lambda_i|$  measures the difference between exact and computed eigenvalues, whereas  $\|\mathbf{A}\mathbf{v}_i - \lambda_i\mathbf{v}_i\|$  measures the quality of the approximated pair  $\{\lambda_i, \mathbf{v}_i\}$ . Moreover,  $m$  is a number of eigenvectors used changed in a range from one till ten, NIters is a number of LOBPCG-EV iterations to get desired accuracy  $\varepsilon_{out}$  for all eigenvectors, MaxIters is a maximal number of iterations allowed.

**Table 7.** Number of iterations for LOBPCG-VS method with DIAG preconditioner,  $p = m$ 

m	1	2	3	4	5
NIters	27	65	74	75	58
$ \lambda_{h,1} - \lambda_1 $	$0.484 \cdot 10^{-5}$	$0.213 \cdot 10^{-9}$	$0.355 \cdot 10^{-11}$	$0.336 \cdot 10^{-5}$	$0.208 \cdot 10^{-11}$
$ \lambda_{h,2} - \lambda_2 $	-	$0.563 \cdot 10^{-5}$	$0.153 \cdot 10^{-7}$	$0.750 \cdot 10^{-10}$	$0.544 \cdot 10^{-8}$
$ \lambda_{h,3} - \lambda_3 $	-	-	$0.961 \cdot 10^{-5}$	$0.247 \cdot 10^{-6}$	$0.103 \cdot 10^{-5}$
$ \lambda_{h,4} - \lambda_4 $	-	-	-	$0.329 \cdot 10^{-5}$	$0.537 \cdot 10^{-5}$
$ \lambda_{h,5} - \lambda_5 $	-	-	-	-	$0.516 \cdot 10^{-6}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.18713 \cdot 10^{-2}$	$0.999 \cdot 10^{-5}$	$0.155 \cdot 10^{-5}$	$0.159 \cdot 10^{-2}$	$0.134 \cdot 10^{-5}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	-	$0.167 \cdot 10^{-2}$	$0.811 \cdot 10^{-4}$	$0.679 \cdot 10^{-5}$	$0.686 \cdot 10^{-4}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	-	-	$0.182 \cdot 10^{-2}$	$0.397 \cdot 10^{-3}$	$0.870 \cdot 10^{-3}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	-	-	-	$0.127 \cdot 10^{-2}$	$0.183 \cdot 10^{-2}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	-	-	-	-	$0.480 \cdot 10^{-3}$
m	6	7	8	9	10
NIters	53	46	50	49	69
$ \lambda_{h,1} - \lambda_1 $	$0.288 \cdot 10^{-10}$	$0.103 \cdot 10^{-9}$	$0.172 \cdot 10^{-4}$	$0.222 \cdot 10^{-11}$	$0.783 \cdot 10^{-17}$
$ \lambda_{h,2} - \lambda_2 $	$0.177 \cdot 10^{-7}$	$0.161 \cdot 10^{-7}$	$0.705 \cdot 10^{-10}$	$0.165 \cdot 10^{-10}$	$0.568 \cdot 10^{-15}$
$ \lambda_{h,3} - \lambda_3 $	$0.446 \cdot 10^{-6}$	$0.671 \cdot 10^{-6}$	$0.556 \cdot 10^{-7}$	$0.361 \cdot 10^{-8}$	$0.383 \cdot 10^{-5}$
$ \lambda_{h,4} - \lambda_4 $	$0.525 \cdot 10^{-5}$	$0.132 \cdot 10^{-5}$	$0.423 \cdot 10^{-8}$	$0.410 \cdot 10^{-8}$	$0.925 \cdot 10^{-14}$
$ \lambda_{h,5} - \lambda_5 $	$0.154 \cdot 10^{-6}$	$0.390 \cdot 10^{-6}$	$0.171 \cdot 10^{-4}$	$0.106 \cdot 10^{-8}$	$0.208 \cdot 10^{-13}$
$ \lambda_{h,6} - \lambda_6 $	$0.333 \cdot 10^{-5}$	$0.290 \cdot 10^{-5}$	$0.382 \cdot 10^{-7}$	$0.180 \cdot 10^{-7}$	$0.623 \cdot 10^{-12}$
$ \lambda_{h,7} - \lambda_7 $	-	$0.528 \cdot 10^{-5}$	$0.715 \cdot 10^{-6}$	$0.717 \cdot 10^{-7}$	$0.424 \cdot 10^{-11}$
$ \lambda_{h,8} - \lambda_8 $	-	-	$0.621 \cdot 10^{-5}$	$0.819 \cdot 10^{-6}$	$0.353 \cdot 10^{-10}$
$ \lambda_{h,9} - \lambda_9 $	-	-	-	$0.347 \cdot 10^{-4}$	$0.447 \cdot 10^{-8}$
$ \lambda_{h,10} - \lambda_{10} $	-	-	-	-	$0.164 \cdot 10^{-6}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.476 \cdot 10^{-5}$	$0.904 \cdot 10^{-5}$	$0.115 \cdot 10^{-2}$	$0.182 \cdot 10^{-5}$	$0.272 \cdot 10^{-8}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	$0.115 \cdot 10^{-3}$	$0.122 \cdot 10^{-3}$	$0.837 \cdot 10^{-5}$	$0.391 \cdot 10^{-5}$	$0.247 \cdot 10^{-7}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	$0.604 \cdot 10^{-3}$	$0.750 \cdot 10^{-3}$	$0.206 \cdot 10^{-3}$	$0.649 \cdot 10^{-4}$	$0.184 \cdot 10^{-2}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	$0.153 \cdot 10^{-2}$	$0.102 \cdot 10^{-2}$	$0.534 \cdot 10^{-4}$	$0.625 \cdot 10^{-4}$	$0.958 \cdot 10^{-7}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	$0.238 \cdot 10^{-3}$	$0.484 \cdot 10^{-3}$	$0.118 \cdot 10^{-2}$	$0.332 \cdot 10^{-4}$	$0.139 \cdot 10^{-6}$
$\ A\mathbf{v}_6 - \lambda_6\mathbf{v}_6\ $	$0.142 \cdot 10^{-2}$	$0.109 \cdot 10^{-2}$	$0.156 \cdot 10^{-3}$	$0.126 \cdot 10^{-3}$	$0.682 \cdot 10^{-6}$
$\ A\mathbf{v}_7 - \lambda_7\mathbf{v}_7\ $	-	$0.198 \cdot 10^{-2}$	$0.570 \cdot 10^{-3}$	$0.214 \cdot 10^{-3}$	$0.165 \cdot 10^{-5}$
$\ A\mathbf{v}_8 - \lambda_8\mathbf{v}_8\ $	-	-	$0.172 \cdot 10^{-2}$	$0.755 \cdot 10^{-3}$	$0.490 \cdot 10^{-5}$
$\ A\mathbf{v}_9 - \lambda_9\mathbf{v}_9\ $	-	-	-	$0.197 \cdot 10^{-2}$	$0.495 \cdot 10^{-4}$
$\ A\mathbf{v}_{10} - \lambda_{10}\mathbf{v}_{10}\ $	-	-	-	-	$0.230 \cdot 10^{-3}$

From the results in Table 7 one can see that the number of iterations is only slightly changed when  $m$  is growing. It shows that the theoretical rate of convergence given in Theorem 1 and 2 is not sharp for the block case and give only an upper bound. The accuracy both eigenvalues and eigenvectors is always better when  $m$  is growing, but it is improved nonuniformly. Moreover, there are some problems with multiple eigenvalues. Indeed,  $\lambda_2 = \lambda_3$  and one can see that the accuracy for  $\lambda_3$  and  $\mathbf{v}_3$  is always one or two order of magnitude worse as for  $\lambda_2$  and  $\mathbf{v}_2$ . The present inconsistency between numerical and theoretical results can be explained by the slow convergence behaviour of the variable-step preconditioning method, i.e. we could not reach the desired accuracy  $\varepsilon_{in}$ , and hence, the actual reduction factor  $q$  is greater as theoretical one  $q(\varepsilon_{in}, \lambda_{k_j}, \lambda_{k_{j+1}})$  from Theorem 4.

**Table 8.** Number of iterations for LOBPCG-VS method with ICF preconditioner,  $p = m$ 

m	1	2	3	4	5
NIters	2	10	10	10	10
$ \lambda_{h,1} - \lambda_1 $	$0.392 \cdot 10^{-5}$	$0.243 \cdot 10^{-3}$	$0.118 \cdot 10^{-3}$	$0.952 \cdot 10^{-4}$	$0.779 \cdot 10^{-4}$
$ \lambda_{h,2} - \lambda_2 $	-	$0.487 \cdot 10^{-12}$	$0.498 \cdot 10^{-13}$	$0.233 \cdot 10^{-15}$	$0.927 \cdot 10^{-18}$
$ \lambda_{h,3} - \lambda_3 $	-	-	$0.196 \cdot 10^{-11}$	$0.741 \cdot 10^{-14}$	$0.837 \cdot 10^{-16}$
$ \lambda_{h,4} - \lambda_4 $	-	-	-	$0.313 \cdot 10^{-7}$	$0.953 \cdot 10^{-11}$
$ \lambda_{h,5} - \lambda_5 $	-	-	-	-	$0.546 \cdot 10^{-11}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.15673 \cdot 10^{-2}$	$0.170 \cdot 10^{-1}$	$0.101 \cdot 10^{-1}$	$0.109 \cdot 10^{-1}$	$0.101 \cdot 10^{-1}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	-	$0.572 \cdot 10^{-6}$	$0.188 \cdot 10^{-6}$	$0.138 \cdot 10^{-7}$	$0.809 \cdot 10^{-9}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	-	-	$0.128 \cdot 10^{-5}$	$0.766 \cdot 10^{-7}$	$0.941 \cdot 10^{-8}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	-	-	-	$0.121 \cdot 10^{-3}$	$0.336 \cdot 10^{-5}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	-	-	-	-	$0.157 \cdot 10^{-5}$
m	6	7	8	9	10
NIters	10	10	10	10	10
$ \lambda_{h,1} - \lambda_1 $	$0.553 \cdot 10^{-4}$	$0.304 \cdot 10^{-4}$	$0.184 \cdot 10^{-4}$	$0.168 \cdot 10^{-4}$	$0.899 \cdot 10^{-5}$
$ \lambda_{h,2} - \lambda_2 $	$0.103 \cdot 10^{-18}$	$0.561 \cdot 10^{-20}$	$0.720 \cdot 10^{-20}$	$0.533 \cdot 10^{-20}$	$0.377 \cdot 10^{-20}$
$ \lambda_{h,3} - \lambda_3 $	$0.216 \cdot 10^{-17}$	$0.261 \cdot 10^{-19}$	$0.105 \cdot 10^{-19}$	$0.122 \cdot 10^{-19}$	$0.110 \cdot 10^{-19}$
$ \lambda_{h,4} - \lambda_4 $	$0.273 \cdot 10^{-12}$	$0.235 \cdot 10^{-14}$	$0.171 \cdot 10^{-15}$	$0.743 \cdot 10^{-16}$	$0.639 \cdot 10^{-18}$
$ \lambda_{h,5} - \lambda_5 $	$0.234 \cdot 10^{-11}$	$0.722 \cdot 10^{-13}$	$0.452 \cdot 10^{-13}$	$0.119 \cdot 10^{-15}$	$0.671 \cdot 10^{-17}$
$ \lambda_{h,6} - \lambda_6 $	$0.439 \cdot 10^{-10}$	$0.822 \cdot 10^{-13}$	$0.569 \cdot 10^{-13}$	$0.525 \cdot 10^{-13}$	$0.112 \cdot 10^{-14}$
$ \lambda_{h,7} - \lambda_7 $	-	$0.142 \cdot 10^{-9}$	$0.490 \cdot 10^{-10}$	$0.158 \cdot 10^{-12}$	$0.361 \cdot 10^{-13}$
$ \lambda_{h,8} - \lambda_8 $	-	-	$0.300 \cdot 10^{-9}$	$0.201 \cdot 10^{-9}$	$0.105 \cdot 10^{-1}$
$ \lambda_{h,9} - \lambda_9 $	-	-	-	$0.710 \cdot 10^{-7}$	$0.105 \cdot 10^{-1}$
$ \lambda_{h,10} - \lambda_{10} $	-	-	-	-	$0.347 \cdot 10^{-8}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.913 \cdot 10^{-2}$	$0.849 \cdot 10^{-2}$	$0.668 \cdot 10^{-2}$	$0.646 \cdot 10^{-2}$	$0.508 \cdot 10^{-2}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	$0.325 \cdot 10^{-9}$	$0.836 \cdot 10^{-10}$	$0.804 \cdot 10^{-10}$	$0.651 \cdot 10^{-10}$	$0.481 \cdot 10^{-10}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	$0.172 \cdot 10^{-8}$	$0.181 \cdot 10^{-9}$	$0.865 \cdot 10^{-10}$	$0.925 \cdot 10^{-10}$	$0.905 \cdot 10^{-10}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	$0.515 \cdot 10^{-6}$	$0.393 \cdot 10^{-7}$	$0.137 \cdot 10^{-7}$	$0.109 \cdot 10^{-7}$	$0.907 \cdot 10^{-9}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	$0.154 \cdot 10^{-5}$	$0.202 \cdot 10^{-6}$	$0.111 \cdot 10^{-6}$	$0.112 \cdot 10^{-7}$	$0.256 \cdot 10^{-8}$
$\ A\mathbf{v}_6 - \lambda_6\mathbf{v}_6\ $	$0.507 \cdot 10^{-5}$	$0.264 \cdot 10^{-6}$	$0.250 \cdot 10^{-6}$	$0.280 \cdot 10^{-6}$	$0.368 \cdot 10^{-7}$
$\ A\mathbf{v}_7 - \lambda_7\mathbf{v}_7\ $	-	$0.115 \cdot 10^{-4}$	$0.712 \cdot 10^{-5}$	$0.337 \cdot 10^{-6}$	$0.172 \cdot 10^{-6}$
$\ A\mathbf{v}_8 - \lambda_8\mathbf{v}_8\ $	-	-	$0.167 \cdot 10^{-4}$	$0.144 \cdot 10^{-4}$	$0.168 \cdot 10^{-1}$
$\ A\mathbf{v}_9 - \lambda_9\mathbf{v}_9\ $	-	-	-	$0.171 \cdot 10^{-3}$	$0.168 \cdot 10^{-1}$
$\ A\mathbf{v}_{10} - \lambda_{10}\mathbf{v}_{10}\ $	-	-	-	-	$0.481 \cdot 10^{-4}$

The results in Table 8 for LOBPCG-VS method with the incomplete factorization preconditioner shows the better performance of the ICF preconditioning with respect to the diagonal one. Indeed, the number of iterations is always smaller and the accuracy is always better except only the first eigenvalue. Moreover, the accuracy is uniformly improved when the number of trial vectors is growing. Note that due to the equal to slow convergence for first eigenvalue the maximal number of iterations for the outer iteration process is always reached (MaxIter= 10). Unfortunately, we could not explain the bad convergence for the first eigenvalue when we use ICF preconditioner, but we think that it is based on the special eigenvalue distribution of the preconditioned matrix, under construction of which we use the so-called row-sum criteria  $A\mathbf{e} = M\mathbf{e}$ , where  $\mathbf{e} = \{1\}$  is a unit vector. However, the present results are not contradict to the present theory. They only shows that the spectral properties of preconditioned system also have to play an important role in the analysis. It is an open question for the futher investigation.

**Table 9.** Number of iterations for LOBPCG-VS method with AMG preconditioner,  $p = m$ 

m	1	2	3	4	5
NIters	2	5	6	6	6
$ \lambda_{h,1} - \lambda_1 $	$0.392 \cdot 10^{-5}$	$0.143 \cdot 10^{-11}$	$0.193 \cdot 10^{-15}$	$0.625 \cdot 10^{-17}$	$0.184 \cdot 10^{-17}$
$ \lambda_{h,2} - \lambda_2 $	-	$0.201 \cdot 10^{-5}$	$0.336 \cdot 10^{-8}$	$0.492 \cdot 10^{-9}$	$0.638 \cdot 10^{-10}$
$ \lambda_{h,3} - \lambda_3 $	-	-	$0.235 \cdot 10^{-6}$	$0.369 \cdot 10^{-8}$	$0.859 \cdot 10^{-9}$
$ \lambda_{h,4} - \lambda_4 $	-	-	-	$0.945 \cdot 10^{-5}$	$0.139 \cdot 10^{-5}$
$ \lambda_{h,5} - \lambda_5 $	-	-	-	-	$0.101 \cdot 10^{-6}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.15673 \cdot 10^{-2}$	$0.127 \cdot 10^{-5}$	$0.155 \cdot 10^{-7}$	$0.257 \cdot 10^{-8}$	$0.177 \cdot 10^{-8}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	-	$0.133 \cdot 10^{-2}$	$0.556 \cdot 10^{-4}$	$0.222 \cdot 10^{-4}$	$0.851 \cdot 10^{-5}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	-	-	$0.505 \cdot 10^{-3}$	$0.574 \cdot 10^{-4}$	$0.350 \cdot 10^{-4}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	-	-	-	$0.193 \cdot 10^{-2}$	$0.106 \cdot 10^{-2}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	-	-	-	-	$0.268 \cdot 10^{-3}$
m	6	7	8	9	10
NIters	6	7	6	9	8
$ \lambda_{h,1} - \lambda_1 $	$0.112 \cdot 10^{-17}$	$0.402 \cdot 10^{-19}$	$0.458 \cdot 10^{-18}$	$0.638 \cdot 10^{-20}$	$0.506 \cdot 10^{-20}$
$ \lambda_{h,2} - \lambda_2 $	$0.401 \cdot 10^{-10}$	$0.162 \cdot 10^{-13}$	$0.255 \cdot 10^{-12}$	$0.182 \cdot 10^{-18}$	$0.454 \cdot 10^{-18}$
$ \lambda_{h,3} - \lambda_3 $	$0.148 \cdot 10^{-9}$	$0.478 \cdot 10^{-13}$	$0.266 \cdot 10^{-11}$	$0.281 \cdot 10^{-17}$	$0.729 \cdot 10^{-17}$
$ \lambda_{h,4} - \lambda_4 $	$0.184 \cdot 10^{-6}$	$0.632 \cdot 10^{-10}$	$0.348 \cdot 10^{-9}$	$0.254 \cdot 10^{-15}$	$0.321 \cdot 10^{-14}$
$ \lambda_{h,5} - \lambda_5 $	$0.282 \cdot 10^{-7}$	$0.273 \cdot 10^{-9}$	$0.266 \cdot 10^{-8}$	$0.177 \cdot 10^{-13}$	$0.363 \cdot 10^{-13}$
$ \lambda_{h,6} - \lambda_6 $	$0.715 \cdot 10^{-6}$	$0.143 \cdot 10^{-7}$	$0.560 \cdot 10^{-7}$	$0.491 \cdot 10^{-12}$	$0.194 \cdot 10^{-11}$
$ \lambda_{h,7} - \lambda_7 $	-	$0.809 \cdot 10^{-7}$	$0.120 \cdot 10^{-6}$	$0.619 \cdot 10^{-11}$	$0.165 \cdot 10^{-10}$
$ \lambda_{h,8} - \lambda_8 $	-	-	$0.146 \cdot 10^{-5}$	$0.366 \cdot 10^{-9}$	$0.375 \cdot 10^{-9}$
$ \lambda_{h,9} - \lambda_9 $	-	-	-	$0.278 \cdot 10^{-6}$	$0.116 \cdot 10^{-6}$
$ \lambda_{h,10} - \lambda_{10} $	-	-	-	-	$0.137 \cdot 10^{-5}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.127 \cdot 10^{-8}$	$0.216 \cdot 10^{-9}$	$0.839 \cdot 10^{-9}$	$0.843 \cdot 10^{-10}$	$0.680 \cdot 10^{-10}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	$0.763 \cdot 10^{-5}$	$0.132 \cdot 10^{-6}$	$0.513 \cdot 10^{-6}$	$0.522 \cdot 10^{-9}$	$0.842 \cdot 10^{-9}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	$0.140 \cdot 10^{-4}$	$0.265 \cdot 10^{-6}$	$0.171 \cdot 10^{-5}$	$0.235 \cdot 10^{-8}$	$0.300 \cdot 10^{-8}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	$0.446 \cdot 10^{-3}$	$0.869 \cdot 10^{-5}$	$0.163 \cdot 10^{-4}$	$0.173 \cdot 10^{-7}$	$0.635 \cdot 10^{-7}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	$0.178 \cdot 10^{-3}$	$0.164 \cdot 10^{-4}$	$0.525 \cdot 10^{-4}$	$0.147 \cdot 10^{-6}$	$0.212 \cdot 10^{-6}$
$\ A\mathbf{v}_6 - \lambda_6\mathbf{v}_6\ $	$0.727 \cdot 10^{-3}$	$0.911 \cdot 10^{-3}$	$0.241 \cdot 10^{-3}$	$0.742 \cdot 10^{-6}$	$0.156 \cdot 10^{-5}$
$\ A\mathbf{v}_7 - \lambda_7\mathbf{v}_7\ $	-	$0.261 \cdot 10^{-3}$	$0.280 \cdot 10^{-3}$	$0.244 \cdot 10^{-5}$	$0.412 \cdot 10^{-5}$
$\ A\mathbf{v}_8 - \lambda_8\mathbf{v}_8\ $	-	-	$0.108 \cdot 10^{-2}$	$0.185 \cdot 10^{-4}$	$0.194 \cdot 10^{-4}$
$\ A\mathbf{v}_9 - \lambda_9\mathbf{v}_9\ $	-	-	-	$0.462 \cdot 10^{-3}$	$0.260 \cdot 10^{-3}$
$\ A\mathbf{v}_{10} - \lambda_{10}\mathbf{v}_{10}\ $	-	-	-	-	$0.114 \cdot 10^{-2}$

As it is readily seen from Table 9 the LOBPCG-VS method with the algebraic multigrid preconditioner is an optimal eigensolver as from the accuracy point of view such as from the required computational costs. Indeed, we have

- The number of iterations does not depend on  $m$ .
- The accuracy is always better when  $m$  is growing.
- The accuracy is always better than for LOBPCG-VS with DIAG preconditioner and is similar to one with ICF preconditioner with the same number of iterations except the first eigenvalue.

Finally, we compare our method with another optimal eigensolver from [19, 20]. This method solves the eigenvalue problems on the sequence of nested grids using an interpolant of the solution on each grid as the initial guess for the next one and improving it by the Full Approximation Scheme (FAS) [32] applied as an inner nonlinear multigrid method. It was shown that the present

method can be used instead of the direct eigensolver when a small number of eigenpairs is required, i.e.,  $p < \sqrt{n}$ .

**Table 10.** Number of iterations for LOBPCG-VS method with AMG preconditioner

$N = 32$ $p = m = 5$	MG eigensolver from [19, 20]	LOBPCG-VS method with		
		DIAG	ICF	AMG
$ \lambda_{h,1} - \lambda_1 $	$0.15763 \cdot 10^{-9}$	$0.208 \cdot 10^{-11}$	$0.379 \cdot 10^{-4}$	$0.184 \cdot 10^{-17}$
$ \lambda_{h,2} - \lambda_2 $	$0.82750 \cdot 10^{-9}$	$0.544 \cdot 10^{-8}$	$0.427 \cdot 10^{-9}$	$0.638 \cdot 10^{-10}$
$ \lambda_{h,3} - \lambda_3 $	$0.82751 \cdot 10^{-9}$	$0.103 \cdot 10^{-5}$	$0.817 \cdot 10^{-9}$	$0.859 \cdot 10^{-9}$
$ \lambda_{h,4} - \lambda_4 $	$0.82751 \cdot 10^{-9}$	$0.537 \cdot 10^{-5}$	$0.253 \cdot 10^{-5}$	$0.139 \cdot 10^{-5}$
$ \lambda_{h,5} - \lambda_5 $	$0.23087 \cdot 10^{-8}$	$0.516 \cdot 10^{-6}$	$0.246 \cdot 10^{-6}$	$0.101 \cdot 10^{-6}$
$\ A\mathbf{v}_1 - \lambda_1\mathbf{v}_1\ $	$0.32745 \cdot 10^{-4}$	$0.134 \cdot 10^{-5}$	$0.004 \cdot 10^{-1}$	$0.177 \cdot 10^{-8}$
$\ A\mathbf{v}_2 - \lambda_2\mathbf{v}_2\ $	$0.76292 \cdot 10^{-4}$	$0.686 \cdot 10^{-4}$	$0.801 \cdot 10^{-5}$	$0.851 \cdot 10^{-5}$
$\ A\mathbf{v}_3 - \lambda_3\mathbf{v}_3\ $	$0.76293 \cdot 10^{-4}$	$0.870 \cdot 10^{-3}$	$0.441 \cdot 10^{-4}$	$0.350 \cdot 10^{-4}$
$\ A\mathbf{v}_4 - \lambda_4\mathbf{v}_4\ $	$0.76293 \cdot 10^{-4}$	$0.183 \cdot 10^{-2}$	$0.136 \cdot 10^{-2}$	$0.106 \cdot 10^{-2}$
$\ A\mathbf{v}_5 - \lambda_5\mathbf{v}_5\ $	$0.12983 \cdot 10^{-3}$	$0.480 \cdot 10^{-3}$	$0.197 \cdot 10^{-3}$	$0.268 \cdot 10^{-3}$

The results in Table 10 shows that the present method with multigrid-like preconditioner give us an optimal eigensolver for finding a few number of smallest eigenpairs of the discretization matrix. Moreover, the LOBPCG-VS method has a very big advantage with respect to other modern optimal eigensolvers. It is very simple for implementation and, it is more important, one can use the modern software packages developed for solving linear system of equations to include them as corresponding variable-step preconditioner for solving the partial eigenproblem.

## References

- [1] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, New York, 1994.
- [2] N.S. Bakhvalov and A.V. Knyazev, *Preconditioned Iterative Methods in a Subspace*, In Domain Decomposition Methods in Science and Engineering, Ed. D. Keyes and J. Xu, AMS, pp. 157–162, 1995.
- [3] J.H. Bramble, J.E Pasciak and A.V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math., 6 (1996), pp. 159–189.
- [4] E.G. D'yakonov, *Modified iterative methods in eigenvalue problems*, Tr. Semin. "Metody Vychisl. Prikl. Mat." 3, Novosibirsk, pp. 39–61 (1978).
- [5] E.G. D'yakonov, *Iteration methods in elliptic problems*, CRC Press, Boca Raton, FL, 1996.
- [6] E.G. D'yakonov, *Optimization in solving elliptic problems*, Math. Notes, 34 (1983), pp. 945–953.
- [7] V.P. Il'in, *Iterative Incomplete Factorization Methods*, World Scientific Publishing Co., Singapore, 1992.
- [8] A.V. Gavrilin and V.P. Il'in, *About one iterative method for solving the partial eigenvalue problem*, Preprint 85, Computing Center, Novosibirsk, 1981.
- [9] S.K. Godunov and Yu.M. Nechepurenko, *On the annular separation of a matrix spectrum*, Comput. Math. Math. Phys., 40 (2000), pp. 939–944.
- [10] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, 1996.

- [11] A.V. Knyazev, *New estimates for Ritz vectors*, Math. Comp. 66 (1997), pp. 985–995.
- [12] A.V. Knyazev, *Preconditioned eigensolvers - an oxymoron?*, ETNA, 7 (1998), pp. 104–123.
- [13] A.V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.
- [14] A.V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration, III: A short and sharp convergence estimate for generalized eigenvalue problems*, Linear Algebra Appl., 358 (2003), 95–114.
- [15] A.V. Knyazev and K. Neymeyr, *Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block preconditioned gradient method*, Electron. Trans. Numer. Anal., 15 (2003), 38–55.
- [16] A.V. Knyazev and A.L. Skorokhodov, *The preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problem*, SIAM J. Numer. Anal., 31 (1994), pp. 1226–1239.
- [17] M. Larin, *An algebraic multilevel iterative method of incomplete factorization for Stieltjes matrices*, Comput. Math. Math. Physics, 38 (1998), pp. 1011–1025.
- [18] M. Larin, *Computation of a few smallest eigenvalues and their eigenvectors for large sparse SPD matrices*, Bull. Nov. Comput. Center, Num. Anal., 11 (2002), pp. 75–86.
- [19] M. Larin, *On a multigrid eigensolver for linear elasticity problems*, Lect. Notes in Computer Science, 2542 (2003), pp. 182–191.
- [20] M. Larin, *On a multigrid method for solving partial eigenproblems*, Sib. J. Num. Math., 7 (2004), pp.25–42.
- [21] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp. 71 (2002), pp. 197–216.
- [22] K. Neymeyr, *A geometric theory for preconditioned inverse iteration, I: Extrema of the Rayleigh quotient*, Lin. Alg. Appl. 332 (2001), pp. 61–85.
- [23] K. Neymeyr, *A geometric theory for preconditioned inverse iteration, II: Convergence estimates*, Lin. Alg. Appl. 332 (2001), pp. 67–104.
- [24] Y. Notay, *DRIC: A Dynamic Version of the RIC Methods*, Num. Lin. Alg. Appl., 1 (1994), pp. 511–532.
- [25] Y. Notay, *Optimal order preconditioning of finite difference matrices*, SIAM J. Sci. Comp., 21 (2000), pp. 1991–2007.
- [26] Y. Notay, *Robust parameter-free algebraic multilevel preconditioning*, Numer. Lin. Alg. Appl., 9 (2002), pp. 409–428.
- [27] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Inc., 1980.
- [28] J. Ruge and K. Stuben, *Algebraic Multigrid (AMG)*, In: "Multigrid Methods" (S. McCormick, ed.), Frontiers in Applied Mathematics, Vol.5, Philadelphia, 1986.
- [29] J. Ruge and K. Stuben, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, In: "Multigrid Methods for Integral and Differential Equations" (D. J. Paddon and H. Holstein, eds.), The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp. 169–212.
- [30] K. Stuben, *Algebraic Multigrid (AMG): Experiences and comparisons*, Appl. Math. Comp., 13 (1983), pp. 419–452.

- [31] K. Stuben, *An Introduction to Algebraic Multigrid*, In the book U. Trottenberg, C. Oosterlee and A. Schuller, *Multigrid*, Academic Press, 2001.
- [32] U. Trottenberg, C. Oosterlee and A. Schuller, *Multigrid*, Academic Press, 2001.

## Appendix A

To investigate the strange behaviour of the convergence rate in Tables 1 and 2 we have to investigate the formula for reduction factor  $q(\varepsilon_{in}, \lambda_{k_j}, \lambda_{k_j+1})$ . Since for a fixed value of  $\varepsilon_{in}$  the number of iterations depends only on the eigenvalues ratio, which is for discretization matrices depend on the characteristic meshsize  $h$ . Note that it is a constant for the continue case.

For simplicity of presentation we consider the two-dimensional case. Formulas both for continuous  $\{\lambda_{l,m}, \mathbf{v}^{(l,m)}\}$  and for discret eigenpairs  $\{\lambda_{l,m}^h, \mathbf{v}_h^{(l,m)}\}$  can be easily derived, and they are

$$\begin{aligned}\lambda_{l,m} &= \pi^2 (a_{11}l^2 + a_{22}m^2), \\ \mathbf{v}^{(l,m)} &= \cos(l\pi x) \cos(m\pi y),\end{aligned}\quad l, m = 1, 2, \dots,$$

and

$$\begin{aligned}\lambda_{l,m}^h &= \frac{4}{h^2} \left[ a_{11} \sin^2 \left( \frac{\pi h}{2} l \right) + a_{22} \sin^2 \left( \frac{\pi h}{2} m \right) \right], \\ [\mathbf{v}_h^{(l,m)}]_{i,j} &= \mathbf{v}^{(l,m)}(ih, jh), \quad l, m = s1, \dots, N = h^{-1}.\end{aligned}\quad (19)$$

Hence, in the case  $a_{11} = a_{22} = 1$  the first and second eigenvalues are

$$\begin{aligned}\lambda_1^h &= \lambda_{1,1}^h = \frac{4}{h^2} \left[ \sin^2 \left( \frac{\pi h}{2} \right) + \sin^2 \left( \frac{\pi h}{2} \right) \right], \\ \lambda_2^h &= \lambda_{1,2}^h = \frac{4}{h^2} \left[ \sin^2 \left( \frac{\pi h}{2} \right) + \sin^2 \left( \frac{\pi h}{2} 2 \right) \right].\end{aligned}$$

Denoting  $\alpha_h = \frac{\pi h}{2}$  and using well-known trigonometric formulae one can easily derived that

$$\frac{\lambda_1^h}{\lambda_2^h} = \frac{2}{1 + 4 \cos^2 \alpha_h}.$$

Now consider two meshes with different meshsize  $h$  and  $H > h$  and compare their eigenvalues ratio. Since the function  $f(x) = \cos x$  is a monotonical decreasing function on the interval  $[0, \pi/2]$  we obtain

$$\frac{\lambda_1^h}{\lambda_2^h} = \frac{2}{1 + 4 \cos^2 \alpha_h} < \frac{2}{1 + 4 \cos^2 \alpha_H} = \frac{\lambda_1^H}{\lambda_2^H}.$$

Thus, we proof that

$$q(\varepsilon_{in}, \lambda_1^h, \lambda_2^h) = \varepsilon_{in} + (1 - \varepsilon_{in}) \frac{\lambda_1^h}{\lambda_2^h} < \varepsilon_{in} + (1 - \varepsilon_{in}) \frac{\lambda_1^H}{\lambda_2^H} = q(\varepsilon_{in}, \lambda_1^H, \lambda_2^H),$$

and hence, the number of iterations is decreased when  $h$  goes to zero, and the limit number of iteration is bounded by the ratio  $(\lambda_1/\lambda_2)$  for continue eigenvalues. The similar behaviour can be expected in the general case for elliptic BVPs.

## Appendix B

Without loss of generality, we can assume that  $a_{11} = 1$  and  $a_{22} = \epsilon \leq a_{11}$ . As it is readily seen from (19) the first and second eigenvalues are

$$\begin{aligned}\lambda_1^h &= \lambda_{1,1}^h = \frac{4}{h^2} \left[ \sin^2 \left( \frac{\pi h}{2} \right) + \epsilon \sin^2 \left( \frac{\pi h}{2} \right) \right], \\ \lambda_2^h &= \lambda_{1,2}^h = \frac{4}{h^2} \left[ \sin^2 \left( \frac{\pi h}{2} \right) + \epsilon \sin^2 \left( \frac{\pi h}{2} \right) \right],\end{aligned}$$

and hence, we obtain

$$q(\varepsilon_{in}, \lambda_1^h, \lambda_2^h) = \varepsilon_{in} + (1 - \varepsilon_{in}) \frac{\lambda_1^h}{\lambda_2^h} = \frac{1 + \epsilon}{1 + 4\epsilon \cos^2 \alpha_h} + \frac{(4 \cos^2 \alpha_h - 1) \varepsilon_{in} \epsilon}{1 + 4\epsilon \cos^2 \alpha_h}.$$

On the other side for  $a_{22} = \delta < \epsilon$  we have

$$q(\varepsilon_{in}, \hat{\lambda}_1^h, \hat{\lambda}_2^h) = \varepsilon_{in} + (1 - \varepsilon_{in}) \frac{\hat{\lambda}_1^h}{\hat{\lambda}_2^h} = \frac{1 + \delta}{1 + 4\delta \cos^2 \alpha_h} + \frac{(4 \cos^2 \alpha_h - 1) \varepsilon_{in} \delta}{1 + 4\delta \cos^2 \alpha_h}.$$

Since  $\alpha_h$  is close to zero, then in what follow we can use  $\cos^2 \alpha_h \approx 1$ . Thus, we obtain

$$q(\varepsilon_{in}, \lambda_1^h, \lambda_2^h) - q(\varepsilon_{in}, \hat{\lambda}_1^h, \hat{\lambda}_2^h) = \frac{1 + \epsilon}{1 + 4\epsilon} \cdot \frac{1 + \delta}{1 + 4\delta} \cdot 3 \cdot (1 + \varepsilon_{in}) \cdot (\epsilon - \delta).$$

which tends to zero as  $\epsilon$  when  $\epsilon \rightarrow 0$  and  $\delta \rightarrow 0$ , and hence, the number of iterations does not considerably changed with respect to the anisotropy ratio.