On a multigrid method for solving partial eigenproblems

Maxim Larin*

Abstract

In the early eighties the direct application of a multigrid technique for solving the partial eigenvalue problem of computing few of the smallest eigenvalues and their corresponding eigenvectors of a large symmetric positive definite matrix A was proposed by Brandt, McCormick and Ruge [4]. 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) [3] applied as an inner nonlinear multigrid method.

In the present paper an experimental study of the method for model elliptic and linear elasticity problems is carried out. Based on these results the quality of the method is improved by using the nonlinear Gauss-Seidel iteration as pre- and postsmoothing steps. Finally, we give some practical advice for a good choice of multigrid-related parameters.

KEY WORDS multigrid methods, eigenvalue problems, sparse matrices.

1 Introduction

In this paper, we are interested in computing few smallest eigenvalues and their corresponding eigenvectors of a large symmetric positive definite matrix A, which arises as a result of the finite element approximation of boundary value problems. Typically, the matrix A is large and sparse. To be able to solve the problem (5) with a reasonable computing time one must use *optimal* (or *nearly optimal*) techniques, i.e., methods for which the computational complexity grows *linear* (or *almost linear*) with increasing problem size.

The usual methods for solving eigenvalue problems are often based on the effect of excitation of the smallest eigenvalues by repeated multiplication of the inverse matrix A^{-1} on a vector. This applies to such popular techniques as a subspace iteration, the Rayleigh quotient and the Lanczos method [23]. However, in the large–scale finite element problems, it is often desirable to avoid a costly inversion or, to be more precise, exact factorization of the matrix A. The simplest way is to use some preconditioned iterative procedure instead of the direct method for solving the linear system with A whenever it is required in the algorithm. In particular, algebraic multilevel and multigrid methods allow us to construct *optimal* preconditioners for sufficiently wide number of industrial applications (see [26] and references therein).

*Institut für Geometrie und Praktische Mathematik, RWTH Aachen, D-52056 Aachen.

Today among of a lot of literature published about efficient eigensolvers one can extract two main directions. The first one is based on the implicit application of algebraic multilevel or multigrid methods to construct an approximated inverse of A and use them as a preconditioner [1, 2, 6, 7, 12, 13, 14, 15, 16], when the second one directly applies the main multigrid ideas of the fine grid relaxation and the coarse grid correction [4, 8, 9, 10, 11].

However, all of these methods are treating this eigenvalue problem as a purely algebraic, and hence, we lose some valuable information about the desired eigenpair. For example, the smallest eigenvector of the Laplace operator is very smooth, i.e., it can be well approximated on coarser grids, and hence, one can use this information during the solution process. The nested iteration multigrid process, which take the advantage of this smoothness, was proposed by Brandt, McCormick and Ruge [4]. The idea of this method is that the eigenvalue problems are solved on the sequence of finer grids using an interpolant of the solution on each level as the initial guess for the next one and improving it by an inner nonlinear multigrid method, i.e., suppressing the high frequencies oscillations arising as a result of the interpolation process.

However, in spite of the good numerical results and the fact that the method was suggested in the early eighties, there has been no substantial further research and development in this direction for many years. It can be intuitively motivated by the lacking of a strong theoretical background due to the nonlinearity of the problem to be solved. The present work is an experimental study of this methods to understand the main points of the method, which have an important meaning as for the accuracy of computed eigenpairs and as for its computational complexity. Moreover, based on first results we slightly modify the relaxation step in the inner iteration process to avoid possible disconvergence when more than one relaxation step used.

The paper is organized as follows. In Section 2 the formal definition of the elliptic boundary value and the linear elasticity eigenvalue problems is given. Moreover, the method for constructing the sequence of matrices is discussed. Further the paper is naturally divided into two parts. In the first part (Section 3) we consider the simplest eigenproblem for finding the smallest eigenvalue and its eigenvector to present a general framework of the full multigrid method or, shortly, the FMG-EV method. Based on numerical results we will make some recommendation for chosing effective user-defined parameters. The FMG-EV(p) method for finding p smallest eigenvalues and their eigenvectors will be discussed in the second part (Section 4). The corresponding numerical results shows its robustness at least for model problems. The final conclusions are given in Section 5.

2 Problem formulation

A lot of physical lows and govering processes can be formulated by means of partial differential equations. In practice, most such problems concern the elliptic self-adjoint boundary value problem

$$\mathbf{div}(k(x)\mathbf{grad}\, u(x)) = \lambda\, u(x),$$

in a bounded domain of R^d with appropriate boundary conditions. Here $d = 2$ or 3 is a spatial dimension. The coefficient matrix $k(x)$ denotes a symmetric positive definite second or fourth order tensor, which is frequently piecewise constant and/or highly anisotropic. The displacement equation of linear elasticity problems have the similar form, where $u(x)$ now denotes the displacement vector $\mathbf{u}(x) = (u_1, ..., u_d)^T$ and the coefficient matrix $k(x)$ denotes the elasticity tensor. Below we will give differential and variational formulations for two model problem of this type, which we will study numerically. The corresponding matrix formulation, which has a unique form for both cases, will be also presented.

2.1 The elliptic boundary value problem

Consider the elliptic boundary value problem with homogeneous boundary conditions

$$-\sum_{i,j=1}^{2} \frac{\partial}{\partial x_i}\left(k_{ij}(x)\frac{\partial}{\partial x_j}u(x)\right) = \lambda u(x), \quad x \in \Omega, \quad u(x) = g(x), \quad x \in \partial\Omega, \tag{1}$$

where Ω is a polygonal domain in R^3 with boundary $\partial\Omega$, where the coefficient matrix $[k_{ij}(x)]$ is assumed to be uniformly symmetric positive definite for any $x \in \overline{\Omega}$, and $g(x)$ is a given function. Although we assume Dirichlet boundary conditions, the results of this paper are valid for general boundary conditions. Under these assumption all eigenvalues λ of (1) are real and positive.

The Galerkin variational formulation of the boundary value problem (1) is as follows.

Find $\lambda_k \in \mathbf{R}$ and $u_k \in H_0^1(\Omega) = \{v \in H^1(\Omega) \,:\, v = 0 \text{ on } \partial\Omega\}$ such that

$$\begin{aligned} a(u_k, v) &= \lambda_k b(u_k, v), \quad k = 1, 2, \ldots, p, \ldots \\ b(u_k, u_l) &= \delta_{kl}, \qquad\quad\;\; l = 1, 2, \ldots, p, \ldots \end{aligned} \tag{2}$$

for all $v \in H_0^1(\Omega)$, where $H^1(\Omega)$ is the usual Sobolev space, δ_{ij} is the Kronecker symbol and bilinear forms $a(u,v)$ and $b(u,v)$ are defined by

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

2.2 The linear elasticity problem

The linear elasticity eigenvalue problem in a domain Ω in $\mathbf{R}^3$ with boundary Γ can be formulated in terms of the displacement vector $\overline{u} = (u_1, u_2, u_3)$, stress tensor $\sigma = (\sigma_{ij})$ and strain tensor $\epsilon = (\epsilon_{ij})$ as follows

$$\begin{aligned} &div\, \sigma = \lambda \overline{u}, \\ &\epsilon_{ij} = \frac{1}{2}\Big(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\Big), \\ &\sigma_{ij}(\overline{u}) = \sum_{k,l=1}^{3} k_{ijkl}(x)\, \epsilon_{kl}(\overline{u}), \\ &\overline{u} = 0 \text{ on } \Gamma_D, \quad \sigma \cdot \overline{n} = 0 \text{ on } \Gamma_N = \Gamma \backslash \Gamma_D. \end{aligned} \tag{3}$$

Here $k_{ijkl}(x)$ is elasticity tensor depending on positive material (Lamé) coefficients, $\overline{n}$ is the outward pointing normal on Γ_N and Γ_D, Γ_N denote Dirichlet and Neumman parts of the boundary, respectively. We assume that $\Gamma_D \neq \emptyset$ and the Korn inequality holds. These assumptions are needed to ensure that all eigenvalues λ of the problem (3) are positive. For more details regarding the problem formulation we refer to [19] or other books on elasticity.

The variational formulation of the linear elasticity eigenvalue problem (3) can be derived in a standard way and is stated as follows:

Find $\lambda_k \in \mathbf{R}$ and $\overline{u}_k \in [H_0^1(\Omega)]^3 = \{\overline{v} \in [H^1(\Omega)]^3 \,:\, \overline{v} = 0 \text{ on } \Gamma_D\}$ such that

$$\begin{array}{ll} a(\overline{u}_k, \overline{v}) = \lambda_k b(\overline{u}_k, \overline{v}), & k = 1, 2, \ldots, p, \ldots \\ b(\overline{u}_k, \overline{u}_l) = \delta_{kl}, & l \;= 1, 2, \ldots, p, \ldots \end{array} \tag{4}$$

for all $\overline{v} \in [H_0^1(\Omega)]^3$ and two bilinear forms $\hat{a}(\overline{u}, \overline{v})$ and $\hat{b}(\overline{u}, \overline{v})$ are defined by

$$\hat{a}(\overline{u}, \overline{v}) = \int\limits_{\Omega} \left[\sum_{i,j,k,l=1}^{3} k_{ijkl} \frac{\partial u_i}{\partial x_j} \frac{\partial v_k}{\partial x_l} \right] d\Omega, \qquad \hat{b}(\overline{u}, \overline{v}) = \int\limits_{\Omega} \left[\sum_{i=1}^{3} u_i v_i \right] d\Omega.$$

2.3 Matrix formulation and the sequence of $A^{(k)}$

Let us assume that the domain Ω is decomposed by a set of finite elements Υ. Introducing $V_h \subset H^1(\Omega)$ or $V_h \subset [H^1(\Omega)]^3$ the space of vector functions with local support, associated with the vertices of Υ, and applying to (2) or (4) the standard Galerkin procedure, we obtain the following eigenvalue problem

$$\begin{array}{lll} A\,\mathbf{u}_k = \lambda_k \mathbf{u}_k, & A \;= A^T \in \mathbf{R}^{n \times n}, & 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_p, \\ (\mathbf{u}_k, \mathbf{u}_l) = \delta_{kl}, & k, l = 1, \ldots, p, & \end{array} \tag{5}$$

where the matrix A corresponds to the bilinear form $a(\cdot,\cdot)$ or $\hat{a}(\cdot,\cdot)$ and $(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v}$.

In order to define a multilevel process we have to construct a sequence of symmetric positive definite matrices $\{A^{(k)}\}$, $k = 0, 1, \ldots, L-1, L$ of an decreasing order n_k such that $A^{(0)} = A$. For last two decade many scientists concerns to the question for solving the linear system of equations: how to construct the *best* (or *optimal*) sequence of matrices $A^{(k)}$ (in terms of computational costs of the corresponding iterative process)? The problem was successfully solved for sufficiently wide nimber of industrial problems (see [26] and references therein). There are two major types of the methods, which are based on either *problem*-dependent geometrical information or *matrix*-dependent numerical information. The choice of the method to compute $\{A^{(k)}\}$ usually depends on the problem to be solved, and hence, is up to the user priority. However, in the case of the eigenvalue problem the situation is quite different.

Let $N_k \in \{1, \ldots, n_k\}$ be the set of indeces of all unknowns (or nodes) on the level k, $k \geq 0$ and is partitioned into two non-intersection subsets, e.g., $N_k^f \subset \{1, \ldots, n_k\}$ and $N_k^c \subset \{1, \ldots, n_k\}$ such that $N_k = N_k^c \bigcup N_k^f$ and $N_k^c \bigcap N_k^f = \emptyset$. This partititioning yields the following block matrix form of $A^{(k)}$

$$A^{(k)} = \left[\begin{array}{cc} A_{11}^{(k)} & A_{12}^{(k)} \\ & \\ A_{21}^{(k)} & A_{22}^{(k)} \end{array} \right] \begin{array}{l} \}n_k \backslash n_{k+1} \\ \\ \}n_{k+1} \end{array} .$$

where the first group of unknowns correspond to the indeces in N_k^f and the second one forms a new set of indices on the next level, i.e., $N_{k+1} = N_k^c$. For the given partititioning, we define a prolongation matrix $P_{k+1}^k \in \mathbf{R}^{n_k \times n_{k+1}}$ of the following form

$$P_{k+1}^k = \left[\begin{array}{c} J_k \\ I_{k+1} \end{array} \right] \begin{array}{l} \}n_k \backslash n_{k+1} \\ \}n_{k+1} \end{array} ,$$

where I_{k+1} is an identity matrix of order n_{k+1} and the choice of blocks J_k depend on the method used for construction of the sequence of matrices $A^{(k)}$.

For example, the matrix-dependent methods are based on the Galerkin coarse-grid matrix formulation:

$$A^{(k+1)} = R_k^{k+1} A^{(k)} P_{k+1}^k, \quad R_k^{k+1} = (P_{k+1}^k)^T,$$

where B^T is a transpone of B and the block J_k in the definition of an interpolation matrix P_{k+1}^k is a sparse approximation of $-(A_{11}^{(k)})^{-1} A_{12}^{(k)}$. Assume "ideal" situation for iterative mehods, i.e.

$$J_k = -(A_{11}^{(k)})^{-1} A_{12}^{(k)},$$

then we get

$$A^{(k+1)} = A_{22}^{(k)} - A_{21}^{(k)} (A_{11}^{(k)})^{-1} A_{12}^{(k)}.$$

and hence, we solve the following eigenproblem on the coarse level

$$A^{(k+1)} \mathbf{u}^{(k+1)} = \lambda^{(k+1)} \mathbf{u}^{(k+1)} \Leftrightarrow (A_{22}^{(k)} - A_{21}^{(k)} (A_{11}^{(k)})^{-1} A_{12}^{(k)}) \mathbf{u}_2^{(k)} = \lambda^{(k+1)} \mathbf{u}_2^{(k)}. \tag{6}$$

On the other hand, from

$$\begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ & \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^{(k)} \\ \\ \mathbf{u}_2^{(k)} \end{bmatrix} = \lambda^{(k)} \begin{bmatrix} \mathbf{u}_1^{(k)} \\ \\ \mathbf{u}_2^{(k)} \end{bmatrix},$$

we obtain the equivalent eigenproblem on the coarse level

$$(A_{22}^{(k)} - A_{21}^{(k)} (A_{11}^{(k)} - \lambda^{(k)} I)^{-1} A_{12}^{(k)}) \mathbf{u}_2^{(k)} = \lambda^{(k)} \mathbf{u}_2^{(k)},$$

which is different (!!!) from (6). Thus, the direct application of matrix-dependent multilevel technique developed for solving the linear system of equations is not a good idea, and hence, the method for computing a good sequence of matrices $A^{(k)}$ is required.

From the first point of view the problem-dependent approach have to work much better than matrix-dependent one, since the problem information about the behaviour of the smallest eigenvectors are used (at least implicitly). Nevertheless, the smallest eigenvalue on two adjoint levels are still different due to different characteristic mesh sizes on different grids.

In the present paper following [4] we use the standard multigrid method, although any kind of multilevel techniques specially developed for eigenproblems can be used to construct that sequence of matrices.

Let us assume that the initial (coarse) triangulation Υ_L of the domain Ω is such that the coefficient matrix $k(x)$ is constant in each element $T_i \in \Upsilon_L$ and can be discontinuous across the elements. To obtain a sufficiently accurate solution of the problem (2) or (4) we make a uniform refinement procedure to construct a sequence of nested meshes (triangulations) $\Upsilon_L \subset \Upsilon_{L-1} \subset \ldots \subset \Upsilon_1 \subset \Upsilon_0 = \Upsilon$. Introducing $V_k \subset H^1(\Omega)$ or $V_k \subset [H^1(\Omega)]^3$ the space of linear vector functions, associated with the vertices of Υ_k, and denote by $A^{(k)}$ the global stiffness matrix on level k, which is calculated as

$$A^{(k)} = \{a(\phi_i^{(k)}, \phi_j^{(k)})\},$$

where $\{\phi_i^{(k)}\}$ is a set of standard (nodal) basis functions in V_k.

Now the Galerkin procedure applied to (2) or (4) on each grid Υ_k leads to the sequence of eigenvalue problems with the corresponding matrices $A^{(k)}$ of the decreasing order:

$$\begin{aligned}&A^{(k)}\mathbf{u}_i^{(k)} = \lambda_i \mathbf{u}_i^{(k)}, \quad A^{(k)} = A^{(k)^T} \in \mathbf{R}^{n\times n}, \quad 0 < \lambda_1^{(k)} \le \lambda_2^{(k)} \le \ldots \le \lambda_p^{(k)},\\ &(\mathbf{u}_i^{(k)}, \mathbf{u}_j^{(k)}) = \delta_{ij}, \quad i,j = 1,\ldots,p, \quad k = 0,1,\ldots,L,\end{aligned}$$

where the matrix $A^{(k)}$ corresponds to the bilinear form $a(\cdot,\cdot)$ or $\hat{a}(\cdot,\cdot)$ on the level k.

For the given partititioning, coefficients for the interpolation of the value $\mathbf{u}^{(k)}$ from $\mathbf{u}^{(k+1)}$ are computed using the standard (geometrical) linear interpolation [8, 27]. Note that by the definition of P_{k+1}^k we have for $k < l$

$$P_l^k = P_{k+1}^k P_l^{k+1} = \ldots = P_{k+1}^k P_{k+2}^{k+1} \ldots P_{l-1}^{l-2} P_l^{l-1},$$

and vica-versa

$$R_k^l = R_{l-1}^l R_k^{l-1} = \ldots = R_{l-1}^l R_{l-2}^{l-1} \ldots R_{k+1}^{k+2} P_k^{k+1}.$$

3 The FMG-EV method for the first eigenvalue

The idea of the FMG-EV method based on the idea of the well-known full multigrid approach, i.e., we find some approximated solution of the analogous problem on the coarse level and using its interpolant as the initial guess for some inner iterative process on the next (finer) one.

3.1 The basic method

Now we formulate the FMG-EV method for solving the sequence of intermediate problems

$$A^{(k)}\mathbf{u}^{(k)} = \lambda^{(k)}\mathbf{u}^{(k)}, \qquad \|\mathbf{u}^{(k)}\| = 1, \qquad k = L, L-1, \ldots, 1, 0, \tag{7}$$

beginning with the coarse level ($k = L$) and including the finest level ($k = 0$) as the final stage.

```
{$\mu^{(0)}, \mathbf{v}^{(0)}$} =  FMG-EV($A^{(l)}, \ldots, A^{(0)}$):
            Compute the coarse level approximation {$\mu^{(L)}, \mathbf{v}^{(L)}$}
            for   $k = L-1, \ldots, 1, 0$
                  $\mathbf{v}_0^{(k)} = P_{k+1}^k \mathbf{v}^{(k+1)}$
                  $\mu_0^{(k)} = \mu^{(k+1)}$
                  for   $it = 1, \ldots, q$     (inner iteration)
                        {$\mu_{it}^{(k)}, \mathbf{v}_{it}^{(k)}$} = FAS-EV ($A^{(k)}, \mu_{it-1}^{(k)}, \mathbf{v}_{it-1}^{(k)}$)
                  $\mathbf{v}^{(k)} = \mathbf{v}_q^{(k)} / \|\mathbf{v}_q^{(k)}\|$
                  $\mu^{(k)} = (A^{(k)}\mathbf{v}^{(k)}, \mathbf{v}^{(k)})/(\mathbf{v}^{(k)}, \mathbf{v}^{(k)})$
```

Here **FAS-EV** $(A, \mu, \mathbf{v})$ denotes the inner multigrid method applied for solving the eigenvalue problem $A\mathbf{u} = \lambda\mathbf{u}$ using $\{\mu, \mathbf{v}\}$ as initial guess.

In order to start the multilevel (recurrent) process we have to compute the coarse level approximation $\{\mu^{(L)}, \mathbf{v}^{(L)}\}$ to the desired eigenpair $\{\lambda^{(L)}, \mathbf{u}^{(L)}\}$ by solving the coarse level eigenvalue problem. Due to the order n_L of the coarse level system is sufficiently small, we can compute it exactly with the small number of arithmetical operations.

Now assume that we have an approximated solution $\{\mu^{(k+1)}, \mathbf{v}^{(k+1)}\}$ on the level $k+1$. The transition to the previous level k starts with the interpolation of the approximated eigenvector $\mathbf{v}^{(k+1)}$ on the next fine level and computing the corresponding initial approximation to $\lambda^{(k)}$ by using the Rayleigh ratio as follows

$$\mathbf{v}_0^{(k)} = P_{k+1}^k \mathbf{v}^{(k+1)}, \qquad \mu_0^{(k)} = \frac{(A^{(k)}\mathbf{v}_0^{(k)}, \mathbf{v}_0^{(k)})}{(\mathbf{v}_0^{(k)}, \mathbf{v}_0^{(k)})}. \tag{8}$$

Next to eliminate the high frequencies oscillations, which arise in result of the interpolation process (8), we apply q times the inner multigrid method, which is based on the full approximation scheme (FAS) approach [3] and used the same sequence of matrices $\{A^{(k)}\}$. The specific nature of the inner solver will be discussed below. Usually, one or two inner multigrid sweeps is enough to suppress all (or at least all) undesired frequencies in the approximated solution. Finally, the iterate $\mathbf{v}_q^{(k)}$ is normalized and a new updated value of $\mu^{(k)}$ is computed. The process is then repeated on the next finer level until the finest level ($k = 0$) is reached.

The remainder of this section is devoted to a detailed discussion of certain aspects of the inner multigrid solver, its relaxation step and numerical results.

3.2 The inner multigrid solver

As stated before, once the vector $\mathbf{v}^{(k)}$ have been interpolated by (8) on the level $k-1$, the vector $\mathbf{v}^{(k-1)}$ is a *weak* approximation to $\mathbf{u}^{(k-1)}$ that we seek. Moreover, the eigenvalue problem is a *nonlinear* problem. Finally, the normalization condition does not ensure the *uniqueness* of the solution since in this case $\pm\mathbf{u}^{(k)}$ are solutions of (7). Nevertheless, since $\mathbf{v}^{(k-1)}$ is still the approximation to $\mathbf{u}^{(k-1)}$ we would like to converge to a new vector $\mathbf{v}^{(k-1)}$ with the same "*sign*" or in the similar direction. Thus to obtain a unique solution one can choose the following normalization condition

$$(\mathbf{v}^{(k-1)}, \mathbf{w}^{(k-1)}) = 1, \tag{9}$$

where $\mathbf{w}^{(k-1)}$ is some *another* approximation to the desired eigenvector $\mathbf{u}^{(k-1)}$.

One possible way to solve all above mentioned problems is to use a nonlinear multigrid method as the full approximation scheme, which is responsible quickly and qualitatively eliminate the high frequency components of the corresponding error $\mathbf{u}^{(k-1)} - \mathbf{v}^{(k-1)}$.

The idea of FAS is the construction of the coarse level problem so that its solution is a just the fine level solution transfered to the coarse level. Let the currently finest problem on the level l be written as follows

$$A^{(l)}\mathbf{u}^{(l)} = \lambda^{(l)}\mathbf{u}^{(l)}, \quad \|\mathbf{u}^{(l)}\| = 1. \tag{10}$$

Assume that we have an approximation $\mathbf{v}^{(l)}$ to the exact solution and the error $\mathbf{u}^{(l)} - \mathbf{v}^{(l)}$ has been smoothed by ν_1th relaxation steps (*presmoothing*), then the problem (10) is transferred on the next level as follows

$$A^{(l+1)}\mathbf{w}^{(l+1)} = \hat{\mu}^{(l+1)}\mathbf{w}^{(l+1)} + \mathbf{b}^{(l+1)}, \quad \phi_{l+1}(\mathbf{w}^{(l+1)}) = \sigma_{l+1}, \tag{11}$$

where

$$\mathbf{b}^{(l+1)} = \left(A^{(l+1)}R_l^{l+1} - R_l^{l+1}A^{(l)}\right)\mathbf{v}^{(l)},$$

and ϕ_{l+1} is a normalization condition, which guarantee the uniqueness of (11) and specifies the size of the solution σ_{l+1}. Note that it is not actually necessary that σ_{l+1} be equal to 1 since a solution of any reasonable size is acceptable. The specific nature of ϕ_l will be discussed below. Note that if $\mathbf{v}^{(l)} = \mathbf{u}^{(l)}$, then $\mathbf{w}^{(l+1)} = R_l^{l+1}\mathbf{u}^{(l)}$ and $\hat{\mu}^{(l+1)} = \lambda^{(l)}$.

Now continue the transfer process to next levels using the same idea yields the sequence of nonlinear problems

$$\begin{array}{lll} A^{(k)}\mathbf{w}^{(k)} = \hat{\mu}^{(k)}\mathbf{w}^{(k)} + \mathbf{b}^{(k)}, & \mathbf{b}^{(k)} = R_{k-1}^k\mathbf{b}^{(k-1)} + \left(A^{(k)}R_{k-1}^k - R_{k-1}^kA^{(k-1)}\right)\mathbf{v}^{(k-1)}, & \\ \phi_k(\mathbf{w}^{(k)}) = \sigma_k, & \mathbf{b}^{(l)} = 0, \qquad k = l, l+1, \ldots, L. & \end{array} \tag{12}$$

To maintain the "*sign*" of the coarse level solution and due to the fact that $R_l^k\mathbf{v}^{(l)}$ is in some sense an approximation to $\mathbf{w}^{(k)}$ one can choose the following normalization condition similar to (9)

$$\phi_k(\mathbf{v}^{(k)}) = (\mathbf{v}^{(k)}, R_l^k\mathbf{v}^{(l)}). \tag{13}$$

and the corresponding definition for the size of the solution

$$\sigma_k = (R_{k-1}^k\mathbf{v}^{(k)}, R_l^k\mathbf{v}^{(l)}). \tag{14}$$

```
{$\mu^{(l)}, \mathbf{v}^{(l)}$} =   **FAS-EV** $(A^{(l)}, \mu_0^{(l)}, \mathbf{v}_0^{(l)})$:
    $\mathbf{b}^{(l)} = \mathbf{0}$
    for $k = l, \ldots, L-1$
        for $it = 1, \ldots, \nu_1$    (presmoothing)
            $\{\mu_{it}^{(k)}, \mathbf{v}_{it}^{(k)}\} = \mathbf{Relax}\ (A^{(k)}, \mu_{it-1}^{(k)}, \mathbf{v}_{it-1}^{(k)}, \mathbf{b}^{(k)})$
        $\{\mu^{(k)}, \mathbf{v}^{(k)}\} = \{\mu_{\nu_1}^{(k)}, \mathbf{v}_{\nu_1}^{(k)}\}$
        $\mathbf{b}^{(k+1)} = R_k^{k+1}\mathbf{b}^{(k)} + \left(A^{(k+1)}R_k^{k+1} - R_k^{k+1}A^{(k)}\right)\mathbf{v}^{(k)}$
        $\mathbf{v}_0^{(k+1)} = R_k^{k+1}\mathbf{v}^{(k)}$
        $\mu_0^{(k+1)} = \mu^{(k)}$
    Solve the coarse level problem    $A^{(L)}\mathbf{v}^{(L)} = \mu^{(L)}\mathbf{v}^{(L)} + \mathbf{b}^{(L)}$
    with the normalization condition $(\mathbf{v}^{(L)}, R_l^L\mathbf{v}^{(l)}) = (R_{L-1}^L\mathbf{v}^{(L)}, R_l^L\mathbf{v}^{(l)})$
    for $k = L-1, \ldots, l$
        $\mathbf{v}_0^{(k)} = \mathbf{v}^{(k)} + P_{k+1}^k(\mathbf{v}^{(k+1)} - R_k^{k+1}\mathbf{v}^{(k)})$
        $\mu_0^{(k)} = \mu^{(k+1)}$
        for $it = 1, \ldots, \nu_2$    (postsmoothing)
            $\{\mu_{it}^{(k)}, \mathbf{v}_{it}^{(k)}\} = \mathbf{Relax}\ (A^{(k)}, \mu_{it-1}^{(k)}, \mathbf{v}_{it-1}^{(k)}, \mathbf{b}^{(k)})$
        $\{\mu^{(k)}, \mathbf{v}^{(k)}\} = \{\mu_{\nu_2}^{(k)}, \mathbf{v}_{\nu_2}^{(k)}\}$

Here **Relax** $(A, \mu, \mathbf{v}, \mathbf{b})$ denotes a nonlinear iteration step applied for solving
the nonlinear system $(A - \hat{\mu}I)\mathbf{w} = \mathbf{b}$ using $\{\mu, \mathbf{v}\}$ as initial guess.
```

Once a suitable approximation $\mathbf{v}^{(L)}$ to $\mathbf{w}^{(L)}$ on the coarse level is found, the approximation $\mathbf{v}^{(L-1)}$ on the previous (finer) level $L-1$ can be corrected as follows

$$\mathbf{v}_{new}^{(L-1)} = \mathbf{v}_{old}^{(L-1)} + P_L^{L-1}(\mathbf{v}^{(L)} - R_L^{L-1}\mathbf{v}_{old}^{(L-1)}),$$

and finally, $\mathbf{v}^{(L-1)}$ is again smoothed by ν_2th relaxation steps (*postsmoothing*).

To proceed, we apply a similar operations to each $\mathbf{v}^{(k)}$, $l \leq k \leq L$, and repeat this process until the fine level l is reached. Finally, note that similary to the linear solvers the prolongation operator in the FAS-EV method does not need to be identical to the prolongation operator in the FMG-EV method, and, moreover, the FAS-EV method can be considered as eigenvalue solver itself.

3.3 Nonlinear relaxation step

The relaxation procedure is as follows. First we linearize the nonlinear problem (12) by fixing the current approximation $\mu_{i-1}^{(k)}$ to $\mu^{(k)}$. Now applying one linear Gauss-Seidel iteration step for solving this linear problem we obtain a new approximation $\mathbf{v}_i^{(k)}$ to the solution $\mathbf{w}^{(k)}$. Next based on definitions (13) and (14) we make normalization

$$\mathbf{v}_i^{(k)} = \frac{(R_{k-1}^k \mathbf{v}^{(k-1)}, R_l^k \mathbf{v}^{(l)})}{(\mathbf{w}^{(k)}, R_l^k \mathbf{v}^{(l)})} \mathbf{w}^{(k)},$$

and finally we update the current approximation to $\mu^{(k)}$ as follows

$$\mu_i^{(k)} = \frac{(A^{(k)} \mathbf{v}_i^{(k)} - \mathbf{b}^{(k)}, \mathbf{v}_i^{(k)})}{(\mathbf{v}_i^{(k)}, \mathbf{v}_i^{(k)})}.$$

$$\begin{aligned}
\{\mu_{it}^{(k)}, \mathbf{v}_{it}^{(k)}\} = \quad & \textbf{Relax}\ (A^{(k)}, \mu_{it-1}^{(k)}, \mathbf{v}_{it-1}^{(k)}, \mathbf{b}^{(k)}) \\
& \mathbf{w}^{(k)} = \textbf{GSI}\ (A^{(k)} - \mu_{it-1}^{(k)} I_k, \mathbf{v}_{it-1}^{(k)}, \mathbf{b}^{(k)}) \\
& \mathbf{v}_{it}^{(k)} = \frac{(R_{k-1}^k \mathbf{v}^{(k-1)}, R_l^k \mathbf{v}^{(l)})}{(\mathbf{w}^{(k)}, R_l^k \mathbf{v}^{(l)})} \mathbf{w}^{(k)} \\
& \mu_{it}^{(k)} = \frac{(A^{(k)} \mathbf{v}_{it}^{(k)} - \mathbf{b}^{(k)}, \mathbf{v}_{it}^{(k)})}{(\mathbf{v}_{it}^{(k)}, \mathbf{v}_{it}^{(k)})}
\end{aligned}$$

Here **GSI** $(A, \mathbf{y}, \mathbf{b})$ denotes the Gauss-Seidel iteration step applied for solving the linear system $A\mathbf{x} = \mathbf{b}$ using $\mathbf{y}$ as initial guess.

3.4 Coarse level solvers

To start the nested iteration process we have to find the coarse level approximation $\{\mu^{(L)}, \mathbf{v}^{(L)}\}$ to the solution $\{\lambda^{(L)}, \mathbf{u}^{(L)}\}$. To solve these problems we use the Gauss-Seidel iteration process followed by the standard normalization condition. There are two possibilities to stop this iterative process. The first one is in that we make a fixed number of relaxation steps ν_{coarse}, and the second one is in that we solve the coarse level problem with a prescribed accuracy ε_{coarse}. As it is readily seen the second version required one additional matrix-vector multiplication and one additional vector-vector subtraction per iteration to compute the current residual $\|\mathbf{r}_i^{(L)}\|$. However, the number of coarse level iteration depend on the order of matrix $A^{(L)}$, and hence, we could not known in advance an optimal number of iteration to get a desired accuracy of solution. Thus, both approaches has their advantages and disadvantages, which was analized by numerical experiments. Based on the results it is recommended to use the second version as a more flexible to the problem to be solved and as a result less time consuming.

```
$\{\mu^{(L)}, \mathbf{v}^{(L)}\} =$ **OuterCoarseLevelSolver** $(A^{(L)})$
    $\mu_0^{(L)} = 0$, $\mathbf{v}_0^{(L)} = \textit{random vector}$, $it = 0$
    $\mathbf{r}_0^{(L)} = A^{(L)}\mathbf{v}_0^{(L)} - \mu_0^{(L)}\mathbf{v}_0^{(L)}$
    **while** $(\ (\ \|r_{it}^{(L)}\| \ / \ \|\mathbf{r}_0^{(L)}\| \geq \varepsilon_{coarse}\ )$ **and** $(\ it \leq \nu_{coarse}\ )\ )$ **do**
        $\mathbf{w}_{it}^{(L)} =$ **GSI** $(A^{(L)} - \mu_{it-1}^{(L)} I_L, \mathbf{v}_{it-1}^{(L)}, \mathbf{b}^{(L)})$
        $\mathbf{v}_{it}^{(L)} = \mathbf{w}_{it}^{(L)} / \|\mathbf{w}_{it}^{(L)}\|$
        $\mu_{it}^{(L)} = (A^{(L)}\mathbf{v}_{it}^{(L)}, \mathbf{v}_{it}^{(L)}) / (\mathbf{v}_{it}^{(L)}, \mathbf{v}_{it}^{(L)})$
        $\mathbf{r}_{it}^{(L)} = A^{(L)}\mathbf{v}_{it}^{(L)} - \mu_{it}^{(L)}\mathbf{v}_{it}^{(L)}$
        $it = it + 1$
    $\mu^{(L)} = \mu_{it}^{(L)}$, $\mathbf{v}^{(L)} = \mathbf{v}_{it}^{(L)}$
```

Finally, we have to note that similar relaxation process as for the pre- and postsmoothings with similar stopping criteria can be used for solving the coarse level problem in the inner multilevel process:

```
$\{\mu^{(L)}, \mathbf{v}^{(L)}\} =$ **InnerCoarseLevelSolver** $(A^{(L)}, \mu_0^{(L)}, \mathbf{v}_0^{(L)}, \mathbf{b}^{(L)})$
    $it = 0$
    $\mathbf{r}_0^{(L)} = A^{(L)}\mathbf{v}_0^{(L)} - \mu_0^{(L)}\mathbf{v}_0^{(L)}$
    **while** $(\ (\ \|r_{it}^{(L)}\| \ / \ \|\mathbf{r}_0^{(L)}\| \geq \varepsilon_{coarse}\ )$ **and** $(\ it \leq \nu_{coarse}\ )\ )$ **do**
        $\{\mu_{it}^{(L)}, \mathbf{v}_{it}^{(L)}\} =$ **Relax** $(A^{(L)}, \mu_{it-1}^{(L)}, \mathbf{v}_{it-1}^{(L)}, \mathbf{b}^{(L)})$
        $\mathbf{r}_{it}^{(L)} = A^{(L)}\mathbf{v}_{it}^{(L)} - \mu_{it}^{(L)}\mathbf{v}_{it}^{(L)}$
        $it = it + 1$
    $\mu^{(L)} = \mu_{it}^{(L)}$, $\mathbf{v}^{(L)} = \mathbf{v}_{it}^{(L)}$
```

3.5 Computational complexity

In order to investigate the whole computational complexity of the FMG-EV method we recall that the solution of eigenvalue problems by the FAS-EV method requires the largest computational costs. By its definition, such a solution breaks up into a set of problems with the matrices $A^{(k)}$ on all levels.

Assume that the number of nodes decreases in a geometrical ratio with a factor ρ defined by

$$\frac{n_{k+1}}{n_k} = \rho_k \leq \rho < 1, \quad k = 0, 1, \ldots, L-1.$$

and hence, we have

$$\sum_{k=0}^{\infty} n_k \leq (1-\rho)^{-1}. \tag{15}$$

Denote by $W_V^{(l)}$ the whole computational complexity of the FAS-EV method applied for solving (10) on the level l. Let C_A and C_P denote the upper bound of the arithmetic work per unknowns defined by the matrix-vector multiplication with matrices $A^{(k)}$ and P_{k+1}^k (R_k^{k+1}), respectively. Then $2C_A n_k$ is approximately the number of operations for one Gauss-Seidel iteration step with $A^{(k)}$, and moreover, $2n_k$ is the computational cost of the scalar product on the level k. Moreover, we assume that the computational costs for solving the coarse level problem is proportional to the number of unknowns on the fine level, i.e., there is a positive constant C_0 such that

$$W_V^{(L)} \le C_0 n_0. \tag{16}$$

Thus, taking into account (15) and (16) we have

$$\begin{aligned} W_V^{(l)} &\le \sum_{k=l}^{L-1} \Big\{ (\nu_1+\nu_2)\Big[(3C_A+7)n_k + (1-\rho)^{-1} C_P n_l\Big] + (2C_A+6C_P+4)n_k \Big\} + C_0 n_0 \\ &\le \left[(\nu_1+\nu_2)(3C_A+7+(L-l)C_P) + 2C_A + 6C_P + 4\right](1-\rho)^{-1} n_l + C_0 n_0. \end{aligned}$$

Now taking into account the prolongation, normalization and computation of the Rayleigh ratio on the level l we obtain the computational complexity of the FMG-EV method for the transfer from level $l+1$ to level l

$$\begin{aligned} W^{(l)} &\le q\left[(\nu_1+\nu_2)(3C_A+7+(L-l)C_P) + 2C_A + 6C_P + 4\right](1-\rho)^{-1} n_l \\ &+ (C_A+C_P+5)n_l + C_0 n_0. \end{aligned}$$

Next summing by levels we obtain that the total computational complexity denoted by W is proportional to the number of nodes on the fine level and the number of level used, i.e.,

$$\begin{aligned} W = \sum_{l=0}^{L} W^{(l)} &\le q\left[(\nu_1+\nu_2)(3C_A+7+\frac{L}{\ln \rho^{-1}}C_P) + 2C_A + 6C_P + 4\right](1-\rho)^{-2} n_0 \\ &+ (C_A+C_P+5)(1-\rho)^{-1} n_0 + L C_0 n_0 = C(q,\nu_1,\nu_2,\rho,C_A,C_P,C_0) L n_0 = O(L n_0), \end{aligned}$$

where $C(\mu,\nu_1,\nu_2,\rho,C_A,C_P)$ is a constant depending on the user-defined parameters only. Hence, the total computational costs of the FMG-EV method for the first eigenpair is nearly optimal.

3.6 Numerical results

The present numerical results can be naturally divided into two parts. At the first part we will test our method for the problem, which have an exact solution, to see the approximation properties of the method and to choose an optimal set of parameters from different points of view, i.e., the accuracy of the solution and the whole computational cost. Then we will check the robustness of the method for the model linear elasticity problem with the optimal set of parameters. All calculations were performed on IBM SP2 in double precision.

To test the method we first consider the eigenvalue problem (5) for the small eigenpair $\{\lambda, \mathbf{u}\}$, which corresponds to the piecewise–linear finite–element discretization of the three dimensional second–order elliptic problems

$$\begin{aligned} -\Delta\, u &= \lambda\, u && \text{in } \Omega, \\ u &= 0 && \text{on } \Gamma_D = \partial\Omega, \\ \|u\| &= 1 \end{aligned} \tag{17}$$

in the cube domain $\Omega = [0,1]^3$ on a uniform Cartesian mesh Υ_h with stepsize h.

Formulas both for continuous $\{\lambda_{l,m,n}, \mathbf{u}^{(l,m,n)}\}$ and for discret eigenpairs $\{\lambda_{l,m,n}^h, \mathbf{u}_h^{(l,m,n)}\}$ can be easily derived, and they are

$$\begin{aligned} \lambda_{l,m,n} &= \pi^2\left(l^2+m^2+n^2\right), \\ \mathbf{u}^{(l,m,n)} &= \sin(l\pi x)\,\sin(m\pi y)\,\sin(n\pi z), \end{aligned} \qquad l,m,n = 1,2,\ldots,$$

for the continuous case, and

$$\begin{aligned} \lambda^h_{l,m,n} &= \frac{4}{h^2}\left[\sin^2\left(\frac{\pi h}{2} l\right) + \sin^2\left(\frac{\pi h}{2} m\right) + \sin^2\left(\frac{\pi h}{2} n\right)\right], \\ [\mathbf{u}_h^{(l,m,n)}]_{i,j,k} &= \mathbf{u}^{(l,m,n)}(ih, jh, kh), \qquad l,m,n = 1,\dots,N, \end{aligned} \tag{18}$$

for the discrete case ($N = h^{-1}$).

To illustrate the quality of the eigensolver used we shall show that the accuracy of the eigenvalue approximation is tended to zero when we increase the number of the relaxation sweeps per level and the number of inner FAS-EV iterations and at the same moment the computational complexity is still optimal with respect to the different parameters used.

The first group of numerical experiments were performed for test the accuracy. It is well-known that in the case of multigrid method when the number of smoothing steps increase and when we find an exact solution on coarse levels, then the multigrid solver for linear system of equation is going to be a direct solver, i.e., the difference between exact and computed solutions tends to a mashine accuracy. We expect the similar behaviour for our eigensolver. To guarantee the exact solvers on the coarse level we define the following values for ν_0, ν_{coarse}, ε_0 and ε_{coarse}. The maximal number of coarse level iteration ν_0 is fixed and is always equal to 100. The maximal number of nonlinear Gauss-Seidel iteration used for computing the initial coarse level approximation ν_{coarse} is also fixed and is always equal to 1000. The prescribed accuracy for the coarse level solvers has been choosen $\varepsilon_{coarse} = 10^{-31}$ and $\varepsilon_0 = 10^{-16}$. Moreover, we use the following generalized parameter ν defined by $\nu = \nu_1 = \nu_2$.

In Tables 3.1–3.2 the results of the experiments for the FMG-EV method for different choices of relaxation steps and inner iterations are given. In all tables of the paper ν is the number of pre- and post- smoothing iterations on each level, q is the number of inner FAS-EV iterations applied on each level and *Lvls* is the number of levels used. Moreover, μ is the computed approximation to the exact eigenvalue λ, and hence, the magnitude $|\lambda-\mu|$ measures the difference between exact and computed eigenvalues, whereas $\|\mathbf{Av} - \mu\mathbf{v}\|$ measures the quality of the approximated pair $\{\mu, \mathbf{v}\}$.

N=16		$q = 1$		$q = 2$	
Lvls	ν	$\lvert\lambda-\mu\rvert$	$\lVert\mathbf{Av}-\mu\mathbf{v}\rVert$	$\lvert\lambda-\mu\rvert$	$\lVert\mathbf{Av}-\mu\mathbf{v}\rVert$
4	1	$0.27669 \cdot 10^{-2}$	$1.21508 \cdot 10^{-1}$	$0.10345 \cdot 10^{-4}$	$0.80434 \cdot 10^{-2}$
	2	$0.12475 \cdot 10^{-4}$	$0.89570 \cdot 10^{-2}$	$0.27632 \cdot 10^{-11}$	$0.43173 \cdot 10^{-5}$
	3	$0.48938 \cdot 10^{-8}$	$0.17608 \cdot 10^{-3}$	$0.19165 \cdot 10^{-18}$	$0.11023 \cdot 10^{-8}$
	4	$0.30555 \cdot 10^{-11}$	$0.44175 \cdot 10^{-5}$	$0.47042 \cdot 10^{-24}$	$0.67100 \cdot 10^{-12}$
	5	$0.12371 \cdot 10^{-14}$	$0.88592 \cdot 10^{-7}$	$0.98828 \cdot 10^{-28}$	$0.62181 \cdot 10^{-14}$
	10	$0.65898 \cdot 10^{-26}$	$0.50767 \cdot 10^{-13}$	$0.19644 \cdot 10^{-31}$	$0.54118 \cdot 10^{-23}$
3	1	$0.27664 \cdot 10^{-2}$	$1.21503 \cdot 10^{-1}$	$0.10345 \cdot 10^{-4}$	$0.80434 \cdot 10^{-2}$
	2	$0.12475 \cdot 10^{-4}$	$0.89570 \cdot 10^{-2}$	$0.27632 \cdot 10^{-11}$	$0.43173 \cdot 10^{-5}$
	3	$0.48929 \cdot 10^{-8}$	$0.17608 \cdot 10^{-3}$	$0.18973 \cdot 10^{-18}$	$0.11020 \cdot 10^{-8}$
	4	$0.30537 \cdot 10^{-11}$	$0.44174 \cdot 10^{-5}$	$0.44469 \cdot 10^{-25}$	$0.53316 \cdot 10^{-12}$
	5	$0.12272 \cdot 10^{-14}$	$0.88570 \cdot 10^{-7}$	$0.48726 \cdot 10^{-31}$	$0.21738 \cdot 10^{-15}$
	10	$0.20800 \cdot 10^{-31}$	$0.29617 \cdot 10^{-15}$	$0.19259 \cdot 10^{-31}$	$0.42379 \cdot 10^{-24}$
2	1	$0.27473 \cdot 10^{-2}$	$1.21096 \cdot 10^{-1}$	$0.10344 \cdot 10^{-4}$	$0.80432 \cdot 10^{-2}$
	2	$0.12475 \cdot 10^{-4}$	$0.89570 \cdot 10^{-2}$	$0.27629 \cdot 10^{-11}$	$0.43172 \cdot 10^{-5}$
	3	$0.48929 \cdot 10^{-8}$	$0.17608 \cdot 10^{-3}$	$0.18970 \cdot 10^{-18}$	$0.11020 \cdot 10^{-8}$
	4	$0.30537 \cdot 10^{-11}$	$0.44174 \cdot 10^{-5}$	$0.44468 \cdot 10^{-25}$	$0.53315 \cdot 10^{-12}$
	5	$0.12272 \cdot 10^{-14}$	$0.88570 \cdot 10^{-7}$	$0.54311 \cdot 10^{-31}$	$0.22586 \cdot 10^{-15}$
	10	$0.95429 \cdot 10^{-31}$	$0.31748 \cdot 10^{-15}$	$0.36111 \cdot 10^{-32}$	$0.26471 \cdot 10^{-16}$

Table 3.1. Accuracy of the computed eigenpair $\{\mu, \mathbf{v}\}$ as a function of ν and q, $p = 1$, Laplace problem

N=32		$q = 1$		$q = 2$	
Lvls	ν	$\lvert\lambda - \mu\rvert$	$\lVert\mathbf{A}\mathbf{v} - \mu\mathbf{v}\rVert$	$\lvert\lambda - \mu\rvert$	$\lVert\mathbf{A}\mathbf{v} - \mu\mathbf{v}\rVert$
5	1	$0.29207 \cdot 10^{-3}$	$0.39956 \cdot 10^{-1}$	$0.72755 \cdot 10^{-6}$	$0.21859 \cdot 10^{-2}$
	2	$0.78026 \cdot 10^{-6}$	$0.22781 \cdot 10^{-2}$	$0.39828 \cdot 10^{-13}$	$0.51935 \cdot 10^{-6}$
	3	$0.15763 \cdot 10^{-9}$	$0.32312 \cdot 10^{-4}$	$0.10628 \cdot 10^{-20}$	$0.84033 \cdot 10^{-10}$
	4	$0.42304 \cdot 10^{-13}$	$0.53015 \cdot 10^{-6}$	$0.73599 \cdot 10^{-28}$	$0.19323 \cdot 10^{-13}$
	5	$0.87054 \cdot 10^{-17}$	$0.75985 \cdot 10^{-8}$	$0.17333 \cdot 10^{-32}$	$0.14855 \cdot 10^{-16}$
	10	$0.20316 \cdot 10^{-29}$	$0.45221 \cdot 10^{-15}$	$0.34474 \cdot 10^{-31}$	$0.23540 \cdot 10^{-26}$
4	1	$0.29206 \cdot 10^{-3}$	$0.39956 \cdot 10^{-1}$	$0.72755 \cdot 10^{-6}$	$0.21859 \cdot 10^{-2}$
	2	$0.78025 \cdot 10^{-6}$	$0.22781 \cdot 10^{-2}$	$0.39828 \cdot 10^{-13}$	$0.51935 \cdot 10^{-6}$
	3	$0.15763 \cdot 10^{-9}$	$0.32312 \cdot 10^{-4}$	$0.10626 \cdot 10^{-20}$	$0.84033 \cdot 10^{-10}$
	4	$0.42301 \cdot 10^{-13}$	$0.53015 \cdot 10^{-6}$	$0.55919 \cdot 10^{-28}$	$0.19277 \cdot 10^{-13}$
	5	$0.86888 \cdot 10^{-17}$	$0.75984 \cdot 10^{-8}$	$0.82814 \cdot 10^{-32}$	$0.39426 \cdot 10^{-17}$
	10	$0.17911 \cdot 10^{-31}$	$0.44587 \cdot 10^{-17}$	$0.15600 \cdot 10^{-31}$	$0.61014 \cdot 10^{-31}$
3	1	$0.29204 \cdot 10^{-3}$	$0.39954 \cdot 10^{-1}$	$0.72755 \cdot 10^{-6}$	$0.21859 \cdot 10^{-2}$
	2	$0.78025 \cdot 10^{-6}$	$0.22781 \cdot 10^{-2}$	$0.39828 \cdot 10^{-13}$	$0.51935 \cdot 10^{-6}$
	3	$0.15763 \cdot 10^{-9}$	$0.32312 \cdot 10^{-4}$	$0.10626 \cdot 10^{-20}$	$0.84033 \cdot 10^{-10}$
	4	$0.42301 \cdot 10^{-13}$	$0.53015 \cdot 10^{-6}$	$0.55899 \cdot 10^{-28}$	$0.19277 \cdot 10^{-13}$
	5	$0.86888 \cdot 10^{-17}$	$0.75984 \cdot 10^{-8}$	$0.14829 \cdot 10^{-31}$	$0.39426 \cdot 10^{-17}$
	10	$0.11362 \cdot 10^{-31}$	$0.44587 \cdot 10^{-17}$	$0.17911 \cdot 10^{-31}$	$0.61580 \cdot 10^{-31}$
2	1	$0.28836 \cdot 10^{-3}$	$0.39700 \cdot 10^{-1}$	$0.72754 \cdot 10^{-6}$	$0.21859 \cdot 10^{-2}$
	2	$0.78025 \cdot 10^{-6}$	$0.22781 \cdot 10^{-2}$	$0.39827 \cdot 10^{-13}$	$0.51935 \cdot 10^{-6}$
	3	$0.15763 \cdot 10^{-9}$	$0.32312 \cdot 10^{-4}$	$0.10626 \cdot 10^{-20}$	$0.84032 \cdot 10^{-10}$
	4	$0.42301 \cdot 10^{-13}$	$0.53015 \cdot 10^{-6}$	$0.55913 \cdot 10^{-28}$	$0.19277 \cdot 10^{-13}$
	5	$0.86888 \cdot 10^{-17}$	$0.75984 \cdot 10^{-8}$	$0.57777 \cdot 10^{-33}$	$0.39426 \cdot 10^{-17}$
	10	$0.71259 \cdot 10^{-32}$	$0.44587 \cdot 10^{-17}$	$0.19451 \cdot 10^{-31}$	$0.60945 \cdot 10^{-31}$

Table 3.2. Accuracy of the computed eigenpair $\{\mu, \mathbf{v}\}$ as a function of ν and q, $p = 1$, Laplace problem

The first results shows that the accuracy of the approximated solution dose not depend on the order of the eigenproblem to be solved, and hence, in what follows we present the results only for $N = 32$ to avoid a large number of similar figures and tables for different grids.

From the presented numerical results we see that if we increase the number of the relaxation steps per level without changes of other parameters, then the accuracy of the approximated solution is improved. Comparing results for $q = 1$ with corresponding ones for $q = 2$ in all Tables one can see that in the second case the accuracy of the approximated solution is *always* better, i.e, increasing the number of inner FAS-EV steps lead to the improving accuracy of the approximated eigenpair $\{\lambda, \mathbf{v}\}$. Moreover, the results for different number of levels are similar, i.e., the accuracy of the approximated solution does not depend on the number of level used. Hence, we show that the FMG-EV method can compute the first eigenvalue with a desired order of accuracy by corresponding choices of eigensolver parameters, when we solve the coarse level problems exactly. Hence, in practice, $q = 1$ and $\nu = 2$ or close to these values are sufficient to ensure the convergence of FMG-EV method.

The second group of numerical experiments were performed for various combinations of the values ε_0 and ε_{coarse}, when the number of inner iterations and relaxation steps are fixed and is always taken as $q = 1$ and $\nu = 2$. The behavior both of the accuracy of eigenvalue approximation and of the time of the whole iterative process with respect to ε_0 and ε_{coarse} for different number of levels are shown in Figures 3.3.1–3.3.2. Here, the value of ε_{coarse} and ε_0 are changed in a range from 10^{-1} to 10^{-10}, and the power of 10 are given along the left axis for ε_{coarse} and along the right axis for ε_0.

From the presented numerical results we see that the accuracy of approximated solution depends only slightly on the values of ε_0 and there is a strong dependence on the values of ε_{coarse}. Here we have to note that starting from some values of ε_{coarse} and ε_0 the accuracy does not changed, i.e., we reach the maximal accuracy for the given number of relaxation steps ν. On the other hand, there is a strong dependence of the computing time on the values of ε_0 and, at the same time, a weak dependence on ε_{coarse}. Hence, in practice, the optimal choice of the iterative parameters seems to be $\varepsilon_{coarse} = 10^{-4}$ and $\varepsilon_0 = 10^{-2}$ or close to these values.

The next group of numerical experiments were performed for various problem sizes and for different numbers of relaxation steps used, when ε_{coarse} and ε_0 are optimal in above mentioned sense and are fixed. The time of the whole iterative process for various values of $Lvls$ with respect to the number of unknowns on the finest mesh and the number of relaxation steps are given in Figures 3.4 and 3.5, respectively. Moreover, in Figure 3.6 we present the dependness of the time of eigenvalue solver and the accuracy of the computed eigenvalue with respect to the number of levels used.

As it is readily seen from Figures 3.4–3.6 the computing time of the whole iterative process essentially depends on the number of level used. For example, in a case of $Lvls = 2$ the computing time of the whole iterative process grows faster than a linear function, i.e., the computational costs per iteration step increases faster than the number of unknowns on the finest mesh. On the other hand, in if $Lvls$ is equal to 3, 4 or 5 the computing time of the whole iterative process grows almost linearly. However, this difference in a behaviour is of no surprise, since in the case $Lvls = 2$ the arithmetic cost per outer iteration is not optimal, inasmuch as the optimal condition (16) on the computational complexity of the coarse level solver is not satisfied. The results of the Table 3.7 show that the number of iterations for the inner coarse level solver depends on the number of nodes as $O(\sqrt{n_L})$. Indeed, we have

$$\rho_1^{-1} = \frac{3375}{343} \approx 9.84, \qquad \rho_2^{-1} = \frac{343}{27} \approx 12.70.$$

At the same time the number of iterations grows approximately with factor $\sqrt{9.84} \approx 3.14$ between second and third columns and with factor $\sqrt{12.70} \approx 3.56$ between third and fourth columns. Hence, the computational complexity, grows as $O(n_L\sqrt{n_L})$, and must satisfy (16) or, the same,

$$O(n_L\sqrt{n_L}) \approx O(n_0) \Leftrightarrow O(n_L) \approx O(n_0^{2/3}),$$

from which taking into account (15) follows that the computational costs per outer process are optimal if and only if

$$O(\rho^L n_0) \approx O(n_0^{2/3}) \Leftrightarrow O(n_0^{1/3}) \approx O(\rho^{-L})$$

holds, and hence,

$$L \geq -\frac{1}{3}\log_\rho n_0.$$

Thus, in the two-level case ($L = 1$) we must satisfy

$$L = 1 \geq -\frac{1}{3}\log_{1/8} n_0,$$

from which follows that the computational costs per outer iteration are optimal if and only if $n_0 < 8^3 = 512$ holds. For the other cases we have

$$L = 2 \geq \frac{1}{3}\log_8 n_0 \Longrightarrow n_0 < 8^6 = 262144,$$

$$L = 3 \geq \frac{1}{3}\log_8 n_0 \Longrightarrow n_0 < 8^9 = 134217728,$$

$$L = 4 \geq \frac{1}{3}\log_8 n_0 \Longrightarrow n_0 < 8^{12} = 68719476736.$$

Thus, the optimal computational complexity can be attained by increasing the number of levels used, and it is proved by numerical results.

Outer coarse level solver				Inner coarse level solver			
Number of unknowns	27	343	3375	Number of unknowns	27	343	3375
$\varepsilon_{coarse} = 10^{-1}$	1.2	1.0	1.0	$\varepsilon_0 = 10^{-1}$	1.9	2.8	5.8
$\varepsilon_{coarse} = 10^{-2}$	4.1	2.2	1.0	$\varepsilon_0 = 10^{-2}$	2.6	4.8	12.8
$\varepsilon_{coarse} = 10^{-3}$	9.3	6.0	8.0	$\varepsilon_0 = 10^{-3}$	3.7	7.5	23.1
$\varepsilon_{coarse} = 10^{-4}$	17.1	11.7	21.0	$\varepsilon_0 = 10^{-4}$	4.2	10.3	34.7
$\varepsilon_{coarse} = 10^{-5}$	23.8	19.4	35.8	$\varepsilon_0 = 10^{-5}$	5.4	13.6	50.7
$\varepsilon_{coarse} = 10^{-6}$	33.0	30.1	63.4	$\varepsilon_0 = 10^{-6}$	6.4	17.7	67.3
$\varepsilon_{coarse} = 10^{-7}$	42.5	39.5	91.9	$\varepsilon_0 = 10^{-7}$	7.6	19.8	82.5
$\varepsilon_{coarse} = 10^{-8}$	50.2	48.2	134.7	$\varepsilon_0 = 10^{-8}$	8.7	23.3	97.1
$\varepsilon_{coarse} = 10^{-9}$	59.5	58.1	164.2	$\varepsilon_0 = 10^{-9}$	9.6	26.9	≥ 100
$\varepsilon_{coarse} = 10^{-10}$	60.8	58.9	176.1	$\varepsilon_0 = 10^{-10}$	10.1	27.4	≥ 100

Table 3.7. The average number of iterations for outer and inner coarse level solvers.

On the other hand, it is also follows from Table 3.7 the number of iterations for the outer coarse level solver does not depends on the number of unknowns on the coarse level and is only depended on the desired accuracy. However, this behaviour is of no surprise, since it is only prove the well-known fact from the convergence theory of the preconditioned inverse iteration method (see [16, 17, 20, 21, 22] and references therein), that the convergence factor depend on the original eigenvalue distibution and the condition number of the preconditioned matrix.

Based on the presented numerical results we can see that the FMG-EV method has an nearly optimal computational complexity, i.e., the time of the whole iterative process does not depend on the number of unknowns on the finest level and does only slightly depend on the number of level used.

Now to illustrate the efficiency of the techniques used the similar experiments for the three dimensional linear elasticity problem with the homogeneous Dirichlet boundary conditions in the cube domain $\Omega = [0,1] \times [0,1] \times [0,1]$ on a uniform Cartesian mesh $\mathcal{T}_h$ with stepsize $h = N^{-1}$ were performed. The "exact" smallest eigenvalue λ has been computed by the Gauss-Seidel iteration method beginning with the random initial guess $\mathbf{v}^{(0)}$ and continue the iterative process until the following stopping criterion was satisfied

$$\frac{\|\mathbf{r}^{(k)}\|}{\|\mathbf{r}^{(0)}\|} < 10^{-12}, \quad \mathbf{r}^{(k)} = A^{(k)}\mathbf{v}^{(k)} - \mu^{(k)}\mathbf{v}^{(k)},$$

where $r^{(0)}$ and $r^{(k)}$ are the initial and the final residuals, respectively.

N=16		$q = 1$			$q = 2$		
L	ν	$\|\lambda - \mu\|$	$\|\|\mathbf{Av} - \mu\mathbf{v}\|\|$	Time (sec)	$\|\lambda - \mu\|$	$\|\|\mathbf{Av} - \mu\mathbf{v}\|\|$	Time (sec)
4	1	$0.18766 \cdot 10^{-2}$	$0.37333 \cdot 10^{-1}$	8.60	$0.13093 \cdot 10^{-3}$	$0.89867 \cdot 10^{-2}$	15.17
	2	$0.13861 \cdot 10^{-3}$	$0.90307 \cdot 10^{-2}$	13.56	$0.13880 \cdot 10^{-5}$	$0.67454 \cdot 10^{-3}$	25.12
	3	$0.90189 \cdot 10^{-5}$	$0.19036 \cdot 10^{-2}$	18.46	$0.18045 \cdot 10^{-7}$	$0.75458 \cdot 10^{-4}$	34.99
	4	$0.13531 \cdot 10^{-5}$	$0.67833 \cdot 10^{-3}$	23.45	$0.13159 \cdot 10^{-8}$	$0.15376 \cdot 10^{-4}$	44.89
	5	$0.11025 \cdot 10^{-6}$	$0.19556 \cdot 10^{-3}$	28.42	$0.25888 \cdot 10^{-9}$	$0.55911 \cdot 10^{-5}$	54.69
	10	$0.14618 \cdot 10^{-8}$	$0.12328 \cdot 10^{-4}$	53.03	$0.34772 \cdot 10^{-11}$	$0.65020 \cdot 10^{-6}$	104.07
3	1	$0.19283 \cdot 10^{-2}$	$0.39311 \cdot 10^{-1}$	9.50	$0.10903 \cdot 10^{-3}$	$0.81701 \cdot 10^{-2}$	16.04
	2	$0.12757 \cdot 10^{-3}$	$0.85886 \cdot 10^{-2}$	14.19	$0.18321 \cdot 10^{-5}$	$0.78770 \cdot 10^{-3}$	25.79
	3	$0.13519 \cdot 10^{-4}$	$0.23281 \cdot 10^{-2}$	18.89	$0.72569 \cdot 10^{-7}$	$0.10573 \cdot 10^{-3}$	34.82
	4	$0.18275 \cdot 10^{-5}$	$0.77967 \cdot 10^{-3}$	23.55	$0.21767 \cdot 10^{-7}$	$0.26623 \cdot 10^{-4}$	44.57
	5	$0.28713 \cdot 10^{-6}$	$0.24979 \cdot 10^{-3}$	28.32	$0.10061 \cdot 10^{-7}$	$0.16873 \cdot 10^{-4}$	54.17
	10	$0.36640 \cdot 10^{-7}$	$0.32665 \cdot 10^{-4}$	52.02	$0.54730 \cdot 10^{-9}$	$0.37209 \cdot 10^{-5}$	101.09
2	1	$0.18780 \cdot 10^{-2}$	$0.39511 \cdot 10^{-1}$	13.08	$0.11463 \cdot 10^{-3}$	$0.84105 \cdot 10^{-2}$	22.25
	2	$0.15070 \cdot 10^{-3}$	$0.97192 \cdot 10^{-2}$	17.95	$0.19856 \cdot 10^{-5}$	$0.75905 \cdot 10^{-3}$	31.70
	3	$0.19502 \cdot 10^{-4}$	$0.25037 \cdot 10^{-2}$	23.58	$0.17259 \cdot 10^{-5}$	$0.21648 \cdot 10^{-3}$	41.76
	4	$0.21043 \cdot 10^{-5}$	$0.81073 \cdot 10^{-3}$	25.97	$0.10470 \cdot 10^{-5}$	$0.15833 \cdot 10^{-3}$	49.52
	5	$0.13009 \cdot 10^{-5}$	$0.32103 \cdot 10^{-3}$	31.10	$0.11669 \cdot 10^{-5}$	$0.17195 \cdot 10^{-3}$	57.79
	10	$0.15593 \cdot 10^{-5}$	$0.19889 \cdot 10^{-3}$	51.08	$0.51628 \cdot 10^{-6}$	$0.10705 \cdot 10^{-3}$	96.00

Table 3.8. Accuracy of the computed eigenpair $\{\mu, \mathbf{v}\}$ as a function of ν and q, $p = 1$, elasticity problem

The results of the corresponding experiments for $N = 16$ are given in Table 3.8. Here the FMG-EV method is used with $\varepsilon_{coarse} = 10^{-4}$ and $\varepsilon_0 = 10^{-2}$. Analizing these results one can see that in the case $q = 1$ and $\nu = 3$ we compute a sufficiently accurate eigenvalue approximation with a reasonable computational cost.

Moreover, in Figure 3.9 we compare the method with these parameters, denoted by "real", to the method with $q = 1$, $\nu = 3$, $\varepsilon_{coarse} = 10^{-16}$ and $\varepsilon_0 = 10^{-16}$, denoted by "ideal", from which one can see that our choice is close to optimal from the accuracy point of view and more attractive from the timeing point of view. Hence, we see that the all conclusion, which are made for Laplace operator, are also valied in a more general case – for linear elasticity problems.

4 The FMG-EV(p) method

Now there are several possibilities to define a multilevel method for computing the first p eigenvalues and their eigenvectors of A. There are two different approaches how to proceed eigenvectors through the multilevel process simultaneously or sequentially. Both of them have their advantages and disadvantages in terms accuracy of the eigenpairs computed, handling the orthogonalization conditions, the computational complexity and the amount of storage requirements.

4.1 The basic method

Following [4] we use a combination of those approaches. Indeed we compute all eigenvectors simultaneously through outer nested iteration process followed by the orthogonalization and the Ritz projection [23] to improve the eigenvector approximations on the currently finest level, but we proceed vector by vector through the inner nonlinear multilevel method.

There are few hard points in this approach. First of all, there is no exact correspondence between eigenvectors on different levels, i.e., the ith eigenvalue and its eigenvector on the coarse level can be a *good* approximation to jth eigenpair on the fine level with $i \neq j$. It is not a problem if j is also less than p, and hence, during the following Ritz projection we find both of desired eigenvectors on the fine level. However, if j is great than p, then the further efforts to improve this approximation in the direction of ith eigenvector is vain since FAS works good only in a small neighbourhood of the solution [24, 25].

Thus, on the coarse level we can find a fixed number of approximation vectors, which are a good approximation to desired eigenvectors. This number depend on the order of the coarse level problem, and hence, can be defined as cn_L, where c is a coefficient less than 1. Now if $p > cn_L$, then we find the coarse level approximations only for first cn_L eigenvectors and start our nested iteration process with this smaller number of vectors. After transfering and processing these approximated eigenvectors on the next finer level we compute the deficient coarse level approximations using the relaxation steps followed by orthogonalization with respect to previous ones, and if p is still great than cn_{L-1}, then we repeat this process on the next finer level and so on. Note that the coarsest level used in the inner multilevel method for ith eigenvector is one on each $\mathbf{v}_i^{(k)}$ first appeared. The rest components of ths method is similar to the FMG-EV method for the first eigenvalue.

```
$\{\mu_i^{(0)}, \mathbf{v}_i^{(0)}\}_{i=1}^p =$   **FMG-EV**$(p, A^{(L)}, \ldots, A^{(0)})$:
        $i = 0,\ k = L,\ \mu_{i-1}^{(k)} = 0$
    1   $i = i + 1$
        Compute the coarse level approximation $\{\mu_i^{(k)}, \mathbf{v}_i^{(k)}\}$,
        which is orthogonal to computed ones $\{\mu_j^{(k)}, \mathbf{v}_j^{(k)}\}_{j=1}^{i-1}$
        **if**  $((i < cn_k)$ **and** $(i < p))$    go to 1
        $\{\mu_j^{(k)}, \mathbf{v}_j^{(k)}\}_{j=1}^{i} =$ **Ritz** $(A^{(0)}, \{\mathbf{v}_j^{(k)}\}_{j=1}^{i})$
    2   **if**  $(k = 0)$    **stop**
        $k = k - 1$
        **for**  $s = 1, \ldots, i$
            $\mathbf{v}_{s,0}^{(k)} = P_{k+1}^{k} \mathbf{v}_s^{(k+1)}$
            $\mu_{s,0}^{(k)} = \mu_s^{(k+1)}$
            **for**  $it = 1, \ldots, q$    (*inner iteration*)
                $\{\mu_{s,it}^{(k)}, \mathbf{v}_{s,it}^{(k)}\} =$ **FAS-EV** $(A^{(k)}, \mu_{s,it-1}^{(k)}, \mathbf{v}_{s,it-1}^{(k)}, \{\mathbf{v}_j^{(k)}\}_{j=1}^{s-1})$
        $\{\mu_j^{(k)}, \mathbf{v}_j^{(k)}\}_{j=1}^{i} =$ **Ritz** $(A^{(0)}, \{\mathbf{v}_{j,q}^{(k)}\}_{j=1}^{i})$
        **if**  $(i < p)$   **then**   go to 1
                            **else**   go to 2
```

Here **FAS-EV** $(A, \mu_i, \mathbf{v}_i, \{\mathbf{v}_j\}_{j=1}^{i-1})$ denotes the inner V-cycle multigrid method applied for solving the eigenproblem $A\mathbf{u}_i = \lambda_i \mathbf{u}_i$ with an orthogonalization condition to $\{\mathbf{v}_j\}_{j=1}^{i-1}$ using $\{\mu_i, \mathbf{v}_i\}$ as initial guess, and **Ritz** $(A, \{\mathbf{v}_i\}_{i=1}^{n})$ computes new eigenvalue and eigenvector approximations in the subspace spanned onto $\mathbf{v}_1, \ldots, \mathbf{v}_n$ by the Ritz method.

4.2 The inner multigrid solver

Since we apply the inner nonlinear multilevel method sequentially to each approximated eigenvector, then the main algorithm for the inner multilevel method presented in Section 3.2 does not changed. Nevertheless, in addition to it we have to orthogonalize the current eigenvector approximation to previously computed ones. It caused the corresponding modifications in the relaxation step and the coarse level solver.

```
$\{\mu_i^{(l)}, \mathbf{v}_i^{(l)}\} =$ FAS-EV $(A^{(l)}, \mu_{i,0}^{(l)}, \mathbf{v}_{i,0}^{(l)}, \{\mathbf{v}_j^{(k)}\}_{j=1}^{i-1})$:
    $\mathbf{b}_i^{(l)} = \mathbf{0}$
    for $k = l, \dots, L-1$
        for $it = 1, \dots, \nu_1$    (presmoothing)
            $\{\mu_{i,it}^{(k)}, \mathbf{v}_{i,it}^{(k)}\} =$ Relax $(A^{(k)}, \mu_{i,it-1}^{(k)}, \mathbf{v}_{i,it-1}^{(k)}, \{\mathbf{v}_j^{(k)}\}_{j=1}^{i-1}, \mathbf{b}_i^{(k)})$
        $\{\mu_i^{(k)}, \mathbf{v}_i^{(k)}\} = \{\mu_{i,\nu_1}^{(k)}, \mathbf{v}_{i,\nu_1}^{(k)}\}$
        $\mathbf{b}_i^{(k+1)} = R_k^{k+1}\mathbf{b}_i^{(k)} + \left(A^{(k+1)}R_k^{k+1} - R_k^{k+1}A^{(k)}\right)\mathbf{v}_i^{(k)}$
        $\mathbf{v}_{i,0}^{(k+1)} = R_k^{k+1}\mathbf{v}_i^{(k)}$
        $\mu_{i,0}^{(k+1)} = \mu_i^{(k)}$
    Solve the coarse level problem $A^{(L)}\mathbf{v}_i^{(L)} = \mu_i^{(L)}\mathbf{v}_i^{(L)} + \mathbf{b}_i^{(L)}$
    with $(\mathbf{v}_i^{(L)}, R_l^L\mathbf{v}_j^{(l)}) = (R_{L-1}^L\mathbf{v}_i^{(L)}, R_l^L\mathbf{v}_j^{(l)}), j = 1, \dots, i-1$
    for $k = L-1, \dots, l$
        $\mathbf{v}_{i,0}^{(k)} = \mathbf{v}_i^{(k)} + P_{k+1}^k(\mathbf{v}_i^{(k+1)} - R_k^{k+1}\mathbf{v}_i^{(k)})$
        $\mu_{i,0}^{(k)} = \mu_i^{(k+1)}$
        for $it = 1, \dots, \nu_2$    (postsmoothing)
            $\{\mu_{i,it}^{(k)}, \mathbf{v}_{i,it}^{(k)}\} =$ Relax $(A^{(k)}, \mu_{i,it-1}^{(k)}, \mathbf{v}_{i,it-1}^{(k)}, \{\mathbf{v}_j^{(k)}\}_{j=1}^{i-1}, \mathbf{b}_i^{(k)})$
        $\{\mu_i^{(k)}, \mathbf{v}_i^{(k)}\} = \{\mu_{i,\nu_2}^{(k)}, \mathbf{v}_{i,\nu_2}^{(k)}\}$
```

Here **Relax** $(A, \mu_i, \mathbf{v}_i, \{\mathbf{v}_j\}_{j=1}^{i-1}, \mathbf{b})$ denotes a nonlinear iteration step applied for solving the nonlinear problem $(A - \hat{\mu}_i I)\mathbf{u}_i = \mathbf{b}$ with an orthogonalization condition to $\{\mathbf{v}_j\}_{j=1}^{i-1}$ using $\{\mu_i, \mathbf{v}_i\}$ i as initial eigenpair approximation.

4.3 Nonlinear relaxation step

As it was mentioned above we have to handle the separateness of computed eigenvector approximation during the inner nonlinear multilevel process. Repeating the same arguments from Section 3.2 we obtain the following sequence of problems

$$\begin{array}{lll} A^{(k)}\mathbf{w}_i^{(k)} = \hat{\mu}_i^{(k)}\mathbf{w}_i^{(k)} + \mathbf{b}_i^{(k)}, & \mathbf{b}_i^{(k)} = R_{k-1}^k\mathbf{b}_i^{(k-1)} + \left(A^{(k)}R_{k-1}^k - R_{k-1}^kA^{(k-1)}\right)\mathbf{v}_i^{(k-1)}, \\ \phi_k(\mathbf{w}_i^{(k)}) = \sigma_k, & \mathbf{b}_i^{(l)} = 0, \qquad k = l, l+1, \dots, L, \quad i = 1, \dots, p. \end{array} \tag{19}$$

Let $\mathbf{u}_j^{(l)}$ is the solution on level l, then $R_l^{l+1}\mathbf{u}_j^{(l)}$ is the solution on next level, and hence, if we would like to find $\mathbf{u}_i^{(l)}$, $i \neq j$, we have to orthogonalize the current eigenvector approximation $\mathbf{v}_i^{(l+1)}$ to $R_l^{l+1}\mathbf{u}_j^{(l)}$. Unfortunately, we do not known the exact solution $\mathbf{u}_j^{(l)}$ (it is our original problem!), but we have $\mathbf{v}_j^{(l)}$, $j < i$, which is a good approximation to $\mathbf{u}_j^{(l)}$. Thus, from a practical point of view $R_l^{l+1}\mathbf{v}_j^{(l)}$ is an approximation to $R_l^{l+1}\mathbf{u}_j^{(l)}$, which one can use to keep orthogonalization on the level $l+1$. Now similar to (13) and (14) we define the following orthonormalization condition

$$\phi_k(\mathbf{v}_i^{(k)}) = (\mathbf{v}_i^{(k)}, R_l^k\mathbf{v}_j^{(l)}), \quad \sigma_k = (R_{k-1}^k\mathbf{v}_i^{(k)}, R_l^k\mathbf{v}_j^{(l)}) \quad j = 1, \dots, i. \tag{20}$$

Nevertheless, we have to note that the condition (20) does not guarantee a uniqueness of the solution (19), since we work with approximations rather than exact solutions. In our case it may happens that the sequence of test vectors $\{R_l^k \mathbf{v}_j^{(l)}\}_{j=1}^{i-1}$ used for the orthogonalization can be linear dependent, and hence, on the level k we will compute $\mathbf{v}_i^{(k)}$, which is a good approximation to $\mathbf{v}_j^{(l)}, j < i$, and which we already known, instead the desired approximation to $\mathbf{v}_i^{(l)}$. It is confirmed by numerical results.

```
$\{\mu_{i,it}^{(k)}, \mathbf{v}_{i,it}^{(k)}\} = $ Relax $(A^{(k)}, \mu_{i,it-1}^{(k)}, \mathbf{v}_{i,it-1}^{(k)}, \{\mathbf{v}_s^{(k)}\}_{s=1}^{i-1}, \mathbf{b}^{(k)})$
    $\mathbf{w}_{i,0}^{(k)} = $ GSI $(A^{(k)} - \mu_{i,it-1}^{(k)} I_k, \mathbf{v}_{i,it-1}^{(k)}, \mathbf{b}^{(k)})$
    for $s = 1, \ldots, i-1$    (orthogonalization)
        $\alpha_s = \dfrac{(R_l^k \mathbf{v}_s^{(l)}, \mathbf{w}_{i,s-1}^{(k)}) - (R_l^k \mathbf{v}_s^{(l)}, R_{k-1}^k \mathbf{v}_i^{(k-1)})}{(R_l^k \mathbf{v}_s^{(l)}, R_l^k \mathbf{v}_s^{(l)})}$
        $\mathbf{w}_{i,s}^{(k)} = \mathbf{w}_{i,s-1}^{(k)} - \alpha_s \cdot R_l^k \mathbf{v}_s^{(l)}$
    $\mathbf{v}_{i,it}^{(k)} = \dfrac{(R_{k-1}^k \mathbf{v}_i^{(k-1)}, R_l^k \mathbf{v}_i^{(l)})}{(\mathbf{w}_{i,i-1}^{(k)}, R_l^k \mathbf{v}_i^{(l)})} \mathbf{w}_{i,i-1}^{(k)}$
    $\mu_{i,it}^{(k)} = \dfrac{(A^{(k)} \mathbf{v}_{i,it}^{(k)} - \mathbf{b}^{(k)}, \mathbf{v}_{i,it}^{(k)})}{(\mathbf{v}_{i,it}^{(k)}, \mathbf{v}_{i,it}^{(k)})}$

Here GSI $(A, \mathbf{y}, \mathbf{b})$ denotes the Gauss-Seidel iteration step applied
for solving the linear system $A\mathbf{x} = \mathbf{b}$ usingi $\mathbf{y}$ as initial guess.
```

4.4 Coarse level solvers

The algorithm for the coarse level solver in inner multigrid method presented in Section 3.4 does not changed since it is based on the definition of the relaxation step, which is already modified. On the other hand, the coarse level solver for computing initial coarse level approximations is changed. It is caused by the orthogonalization condition. To do this we use the standard Gramma-Schmidt orthogonalization process, and hence, we obtain

```
$\{\mu_i^{(L)}, \mathbf{v}_i^{(L)}\} =$ **OuterCoarseLevelSolver** $(A^{(L)}, \{\mathbf{v}_s^{(k)}\}_{s=1}^{i-1})$
    $\mu_{i,0}^{(L)} = 0$, $\mathbf{v}_{i,0}^{(L)} = random\ vector$, $it = 0$
    $\mathbf{r}_0^{(L)} = A^{(L)}\mathbf{v}_{i,0}^{(L)} - \mu_{i,0}^{(L)}\mathbf{v}_{i,0}^{(L)}$
    **while** $( ( \|r_{it}^{(L)}\| \,/\, \|\mathbf{r}_0^{(L)}\| \geq \varepsilon_{coarse} )$ **and** $( it \leq \nu_{coarse} ) )$ **do**
        $\mathbf{w}_{it,0}^{(L)} =$ **GSI** $(A^{(L)} - \mu_{i,it-1}^{(L)} I_L, \mathbf{v}_{i,it-1}^{(L)}, \mathbf{b}^{(L)})$
        **for** $s = 1, \ldots, i-1$    $(orthogonalization)$
            $\alpha_s = \frac{(\mathbf{v}_s^{(L)}, \mathbf{w}_{it,s-1}^{(L)})}{(\mathbf{v}_s^{(L)}, \mathbf{v}_s^{(L)})}$
            $\mathbf{w}_{it,s}^{(L)} = \mathbf{w}_{it,s-1}^{(L)} - \alpha_s \cdot \mathbf{v}_s^{(L)}$
        $\mathbf{v}_{i,it}^{(L)} = \mathbf{w}_{it,i-1}^{(L)} / \|\mathbf{w}_{it,i-1}^{(L)}\|$
        $\mu_{i,it}^{(L)} = (A^{(L)}\mathbf{v}_{i,it}^{(L)}, \mathbf{v}_{i,it}^{(L)}) / (\mathbf{v}_{i,it}^{(L)}, \mathbf{v}_{i,it}^{(L)})$
        $\mathbf{r}_{it}^{(L)} = A^{(L)}\mathbf{v}_{i,it}^{(L)} - \mu_{i,it}^{(L)}\mathbf{v}_{i,it}^{(L)}$
        $it = it + 1$
    $\mu_i^{(L)} = \mu_{i,it}^{(L)}$, $\mathbf{v}_i^{(L)} = \mathbf{v}_{i,it}^{(L)}$
```

4.5 Ritz method

It is well-known that Ritz projection computes the best set of approximated eigenvectors from a subspace to eigenvectors of the original matrix A [23]. Here we use them to find all eigenvectors from the subspace spanned onto computed approximations $\mathbf{v}_1^{(k)}, \ldots, \mathbf{v}_p^{(k)}$. As a result we get a number of orthonormal approximated eigenvectors and their eigenvalues. Here we present only the basic steps of the method, which are required for definition the computational costs in the next subsection.

```
$\{\mu_i, \mathbf{v}_i\}_{i=1}^p =$ **Ritz** $(A, \hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_p)$
    1  Orthonormalize the vectors $\{\hat{\mathbf{v}}_i\}_{i=1}^p$ and form $n$-by-$p$ matrix
       $W = [\mathbf{w}_1^{(k)}, \ldots, \mathbf{w}_p^{(k)}]$ from new orthonormal vectors $\{\mathbf{w}_i^{(k)}\}_{i=1}^p$
    2  Compute matrix-vector products $A\mathbf{w}_i^{(k)}$, $i = 1, \ldots, p$
    3  Compute scalar products $h_{ij} = (\mathbf{w}_j^{(k)}, A\mathbf{w}_i^{(k)})$ and
       form $p$-by-$p$ matrix $H = \{h_{ij}\} = W^T A W$
    4  Compute all eigenpairs of $H$: $H\mathbf{u}_i = \mu_i \mathbf{u}_i$, $i = 1, \ldots, p$
    5  Compute all Ritz vectors $\mathbf{v}_i = W\mathbf{u}_i$, $i = 1, \ldots, p$.
```

4.6 Computational complexity

It is evidence that the whole computational complexity of the FMG-EV(p) method grows at least linear with respect to the number of desired eigenpairs since we proceed vector by vector through the inner nonlinear multilevel method. However, the main difference between methods for computing of one and few eigenpairs is keeping of their separateness during computational process. It is handled by a Gram-Schmidt orthogonalization method and a Ritz projection method for the outer process and by a specific orthogonalization method during relaxation steps for the inner process. The later is change the formulae for the whole computational complexity of inner multilevel process applied to ith eigenvector approximation as follows

$$\hat{W}_{V,i}^{(l)} \le \sum_{k=l}^{L-1} \Big[(\nu_1+\nu_2)\Big\{(3C_A+9)n_k + (i-1)\Big[(1-\rho)^{-1}C_P n_l + 7n_k\Big]\Big\}$$
$$+(2C_A+6C_P+4)n_k\Big] + C_0 n_0$$

$$\le [(\nu_1+\nu_2)\{3C_A+9+(i-1)((L-l)C_P+7)\}+2C_A+6C_P+4](1-\rho)^{-1}n_l + C_0 n_0.$$

Next summing by i we obtain

$$\hat{W}_V^{(l)} \le \sum_{i=1}^{p} \hat{W}_{V,i}^{(l)} \le p(\nu_1+\nu_2)(3C_A+9)(1-\rho)^{-1}n_l + \frac{1}{2}p(p-1)((L-l)C_P+7)(1-\rho)^{-1}n_l$$
$$+p(2C_A+6C_P+4)(1-\rho)^{-1}n_l + pC_0 n_0.$$

On the other hand, the Ritz projection method on the level l requires the following computational cost

$$W_{Ritz}^{(l)} = p(p+1)n_l + pC_A n_l + \frac{1}{2}p(p+1)n_l + O(p^3) + p^2 n_l$$
$$= \frac{1}{2}p(5p+3)n_l + pC_A n_l + O(p^3),$$

where each term of the sum corresponds to above defined actions of the Ritz method, respectively.

Now taking into account the prolongation of p vectors from level n_{l+1} to n_l and the computation of their Ritz projection on the level l we obtain the computational complexity of the FAML-EV method for the transfer from level $l+1$ to level l

$$\hat{W}^{(l)} \le q\hat{W}_V^{(l)} + pC_P n_l + W_{Ritz}^{(l)}$$

$$\le C(\nu_1,\nu_2,C_A,C_P)\left(qp^2(L-l)(1-\rho)^{-1}n_l + qpn_0\right) + O(p^2 n_l) + O(p^3).$$

Finally, summing by levels we obtain that the total computational complexity denoted by $\hat{W}$ is proportional to the number of nodes on the fine level, the number of level used and the number of desired eigenvectors, i.e.,

$$\hat{W} = \sum_{l=0}^{L} \hat{W}^{(l)} \le C(\nu_1,\nu_2,C_A,C_P)\left(qp^2 \tfrac{L}{\ln \rho^{-1}}(1-\rho)^{-2}n_0 + qpLn_0\right) + O(p^2 L n_0) + O(Lp^3)$$
$$= C(q,\nu_1,\nu_2,\rho,C_A,C_P)p^2 L n_0 = O(p^2 L n_0).$$

where $C(\mu,\nu_1,\nu_2,\rho,C_A,C_P)$ is a constant depending on the user-defined parameters only. Hence, the total computational costs of the FMG-EV method is nearly optimal.

4.7 Numerical results

First we use the above mentioned test problem (17) with the exact solution (18) to check the quality of the FMG-EV(p) method. Similar to the case for the first eigenpair the following values of parameters has been choosen $\varepsilon_{coarse} = 10^{-31}$, $\varepsilon_0 = 10^{-16}$, $\nu_{coarse} = 1000$ and $\nu_0 = 100$. The results for the first five eigenvalues and their eigenvectors on $(32 \times 32 \times 32)$-grid are given in Tables 4.1–4.2.

q	ν	$\lvert\lambda_1-\mu_1\rvert$	$\lvert\lambda_2-\mu_2\rvert$	$\lvert\lambda_3-\mu_3\rvert$	$\lvert\lambda_4-\mu_4\rvert$	$\lvert\lambda_5-\mu_5\rvert$
Lvls = 5, NFine = 29791, NCoarse = 1, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.83528 \cdot 10^{-2}$	$0.94833 \cdot 10^{-3}$	$0.94864 \cdot 10^{-3}$	$0.94896 \cdot 10^{-3}$	$0.83950 \cdot 10^{-1}$
	2	$0.69550 \cdot 10^{-6}$	$0.31825 \cdot 10^{-5}$	$0.31826 \cdot 10^{-5}$	$0.31827 \cdot 10^{-5}$	$0.43920 \cdot 10^{-1}$
	3	$0.16185 \cdot 10^{-9}$	$0.89322 \cdot 10^{-9}$	$0.89323 \cdot 10^{-9}$	$0.89323 \cdot 10^{-9}$	$0.21761 \cdot 10^{-1}$
	4	$0.34395 \cdot 10^{-13}$	$0.33402 \cdot 10^{-12}$	$0.33402 \cdot 10^{-12}$	$0.33403 \cdot 10^{-12}$	$0.19062 \cdot 10^{-1}$
	5	$0.86509 \cdot 10^{-17}$	$0.88854 \cdot 10^{-16}$	$0.88855 \cdot 10^{-16}$	$0.88862 \cdot 10^{-16}$	$0.18896 \cdot 10^{-1}$
	10	$0.20247 \cdot 10^{-29}$	$0.66829 \cdot 10^{-31}$	$0.26770 \cdot 10^{-31}$	$0.19259 \cdot 10^{-31}$	$0.18890 \cdot 10^{-1}$
2	1	$0.66492 \cdot 10^{-6}$	$0.28152 \cdot 10^{-5}$	$0.28162 \cdot 10^{-5}$	$0.28166 \cdot 10^{-5}$	$0.19049 \cdot 10^{-2}$
	2	$0.36474 \cdot 10^{-12}$	$0.55080 \cdot 10^{-11}$	$0.29096 \cdot 10^{-12}$	$0.30506 \cdot 10^{-12}$	$0.18892 \cdot 10^{-1}$
	3	$0.11061 \cdot 10^{-20}$	$0.48379 \cdot 10^{-15}$	$0.44919 \cdot 10^{-15}$	$0.11955 \cdot 10^{-19}$	$0.18891 \cdot 10^{-1}$
	4	$0.75144 \cdot 10^{-28}$	$0.38784 \cdot 10^{-18}$	$0.12579 \cdot 10^{-19}$	$0.11994 \cdot 10^{-26}$	$0.18890 \cdot 10^{-1}$
	5	$0.15214 \cdot 10^{-31}$	$0.15702 \cdot 10^{-21}$	$0.33236 \cdot 10^{-22}$	$0.10977 \cdot 10^{-31}$	$0.18890 \cdot 10^{-1}$
	10	$0.24459 \cdot 10^{-31}$	$0.19837 \cdot 10^{-31}$	$0.50074 \cdot 10^{-32}$	$0.32548 \cdot 10^{-31}$	$0.18890 \cdot 10^{-1}$
Lvls = 4, NFine = 29791, NCoarse = 27, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.29206 \cdot 10^{-3}$	$0.97645 \cdot 10^{-3}$	$0.97648 \cdot 10^{-3}$	$0.97726 \cdot 10^{-3}$	0.1805769470
	2	$0.78025 \cdot 10^{-6}$	$0.31911 \cdot 10^{-5}$	$0.31911 \cdot 10^{-5}$	$0.31911 \cdot 10^{-5}$	0.1722472399
	3	$0.15763 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.82751 \cdot 10^{-9}$	0.1719186764
	4	$0.42301 \cdot 10^{-13}$	$0.32098 \cdot 10^{-12}$	$0.32098 \cdot 10^{-12}$	$0.32099 \cdot 10^{-12}$	0.1718761739
	5	$0.86888 \cdot 10^{-17}$	$0.85300 \cdot 10^{-16}$	$0.85300 \cdot 10^{-16}$	$0.85307 \cdot 10^{-16}$	0.1718697271
	10	$0.94370 \cdot 10^{-32}$	$0.75111 \cdot 10^{-32}$	$0.19066 \cdot 10^{-31}$	$0.21955 \cdot 10^{-31}$	0.1718685606
2	1	$0.72755 \cdot 10^{-6}$	$0.28289 \cdot 10^{-5}$	$0.28290 \cdot 10^{-5}$	$0.28291 \cdot 10^{-5}$	0.1719674484
	2	$0.39828 \cdot 10^{-13}$	$0.29559 \cdot 10^{-12}$	$0.29563 \cdot 10^{-12}$	$0.29565 \cdot 10^{-12}$	0.1718698729
	3	$0.10626 \cdot 10^{-20}$	$0.10586 \cdot 10^{-19}$	$0.12209 \cdot 10^{-19}$	$0.16090 \cdot 10^{-19}$	0.1718685910
	4	$0.55921 \cdot 10^{-28}$	$0.10966 \cdot 10^{-26}$	$0.20140 \cdot 10^{-25}$	$0.25174 \cdot 10^{-24}$	0.1718685612
	5	$0.19451 \cdot 10^{-31}$	$0.62785 \cdot 10^{-31}$	$0.37420 \cdot 10^{-30}$	$0.14590 \cdot 10^{-29}$	0.1718685605
	10	$0.16370 \cdot 10^{-31}$	$0.15312 \cdot 10^{-21}$	$0.41675 \cdot 10^{-21}$	$0.18266 \cdot 10^{-23}$	0.1718685605
Lvls = 3, NFine = 29791, NCoarse = 343, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.29204 \cdot 10^{-3}$	$0.97639 \cdot 10^{-3}$	$0.97655 \cdot 10^{-3}$	$0.97723 \cdot 10^{-3}$	$0.18430 \cdot 10^{-2}$
	2	$0.78025 \cdot 10^{-6}$	$0.31911 \cdot 10^{-5}$	$0.31911 \cdot 10^{-5}$	$0.31912 \cdot 10^{-5}$	$0.71561 \cdot 10^{-5}$
	3	$0.15763 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.82751 \cdot 10^{-9}$	$0.82751 \cdot 10^{-9}$	$0.23087 \cdot 10^{-8}$
	4	$0.42301 \cdot 10^{-13}$	$0.32098 \cdot 10^{-12}$	$0.32098 \cdot 10^{-12}$	$0.32099 \cdot 10^{-12}$	$0.11690 \cdot 10^{-11}$
	5	$0.86888 \cdot 10^{-17}$	$0.85299 \cdot 10^{-16}$	$0.85299 \cdot 10^{-16}$	$0.85300 \cdot 10^{-16}$	$0.38678 \cdot 10^{-15}$
	10	$0.11362 \cdot 10^{-31}$	$0.82814 \cdot 10^{-32}$	$0.71259 \cdot 10^{-31}$	$0.81659 \cdot 10^{-31}$	$0.10400 \cdot 10^{-31}$
2	1	$0.72755 \cdot 10^{-6}$	$0.28289 \cdot 10^{-5}$	$0.28290 \cdot 10^{-5}$	$0.28291 \cdot 10^{-5}$	$0.61068 \cdot 10^{-5}$
	2	$0.39828 \cdot 10^{-13}$	$0.29558 \cdot 10^{-12}$	$0.29560 \cdot 10^{-12}$	$0.29561 \cdot 10^{-12}$	$0.10645 \cdot 10^{-11}$
	3	$0.10626 \cdot 10^{-20}$	$0.10593 \cdot 10^{-19}$	$0.11592 \cdot 10^{-19}$	$0.16149 \cdot 10^{-19}$	$0.52750 \cdot 10^{-19}$
	4	$0.55899 \cdot 10^{-28}$	$0.10728 \cdot 10^{-26}$	$0.50317 \cdot 10^{-25}$	$0.25403 \cdot 10^{-24}$	$0.85392 \cdot 10^{-26}$
	5	$0.10977 \cdot 10^{-31}$	$0.30814 \cdot 10^{-31}$	$0.16555 \cdot 10^{-29}$	$0.22887 \cdot 10^{-29}$	$0.25422 \cdot 10^{-31}$
	10	$0.14829 \cdot 10^{-31}$	$0.41214 \cdot 10^{-31}$	$0.10014 \cdot 10^{-31}$	$0.13481 \cdot 10^{-31}$	$0.10168 \cdot 10^{-30}$
Lvls = 2, NFine = 29791, NCoarse = 3375, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.28836 \cdot 10^{-3}$	$0.96578 \cdot 10^{-3}$	$0.96578 \cdot 10^{-3}$	$0.96578 \cdot 10^{-3}$	$0.18266 \cdot 10^{-2}$
	2	$0.78025 \cdot 10^{-6}$	$0.31911 \cdot 10^{-5}$	$0.31911 \cdot 10^{-5}$	$0.31911 \cdot 10^{-5}$	$0.71560 \cdot 10^{-5}$
	3	$0.15763 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.82750 \cdot 10^{-9}$	$0.23087 \cdot 10^{-8}$
	4	$0.42301 \cdot 10^{-13}$	$0.32098 \cdot 10^{-12}$	$0.32098 \cdot 10^{-12}$	$0.32098 \cdot 10^{-12}$	$0.11690 \cdot 10^{-11}$
	5	$0.86888 \cdot 10^{-17}$	$0.85299 \cdot 10^{-16}$	$0.85299 \cdot 10^{-16}$	$0.85299 \cdot 10^{-16}$	$0.38678 \cdot 10^{-15}$
	10	$0.44296 \cdot 10^{-32}$	$0.18681 \cdot 10^{-31}$	$0.18103 \cdot 10^{-31}$	$0.20029 \cdot 10^{-31}$	$0.80889 \cdot 10^{-32}$
2	1	$0.72754 \cdot 10^{-6}$	$0.28289 \cdot 10^{-5}$	$0.28289 \cdot 10^{-5}$	$0.28289 \cdot 10^{-5}$	$0.61063 \cdot 10^{-5}$
	2	$0.39827 \cdot 10^{-13}$	$0.29557 \cdot 10^{-12}$	$0.29557 \cdot 10^{-12}$	$0.29557 \cdot 10^{-12}$	$0.10644 \cdot 10^{-11}$
	3	$0.10626 \cdot 10^{-20}$	$0.10585 \cdot 10^{-19}$	$0.10585 \cdot 10^{-19}$	$0.10585 \cdot 10^{-19}$	$0.52701 \cdot 10^{-19}$
	4	$0.55911 \cdot 10^{-28}$	$0.10640 \cdot 10^{-26}$	$0.10641 \cdot 10^{-26}$	$0.10641 \cdot 10^{-26}$	$0.84829 \cdot 10^{-26}$
	5	$0.90518 \cdot 10^{-32}$	$0.19644 \cdot 10^{-31}$	$0.12903 \cdot 10^{-31}$	$0.24459 \cdot 10^{-31}$	$0.16948 \cdot 10^{-31}$
	10	$0.24844 \cdot 10^{-31}$	$0.47570 \cdot 10^{-31}$	$0.20029 \cdot 10^{-31}$	$0.25614 \cdot 10^{-31}$	$0.49688 \cdot 10^{-31}$

Table 4.1. Accuracy of computed eigenvalues μ_i as a function of ν and q, $p = 5$, Laplace problem

q	ν	$\|\mathbf{Av}_1-\mu_1\mathbf{v}_1\|$	$\|\mathbf{Av}_2-\mu_2\mathbf{v}_2\|$	$\|\mathbf{Av}_3-\mu_3\mathbf{v}_3\|$	$\|\mathbf{Av}_4-\mu_4\mathbf{v}_4\|$	$\|\mathbf{Av}_5-\mu_5\mathbf{v}_5\|$
Lvls = 5, NFine = 29791, NCoarse = 1, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.29266 \cdot 10^{-1}$	$0.71637 \cdot 10^{-1}$	$0.71649 \cdot 10^{-1}$	$0.71658 \cdot 10^{-1}$	$0.83950 \cdot 10^{-1}$
	2	$0.22693 \cdot 10^{-2}$	$0.45634 \cdot 10^{-2}$	$0.45634 \cdot 10^{-2}$	$0.45635 \cdot 10^{-2}$	$0.65110 \cdot 10^{-1}$
	3	$0.32745 \cdot 10^{-4}$	$0.76292 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.23570 \cdot 10^{-1}$
	4	$0.53360 \cdot 10^{-6}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.57555 \cdot 10^{-2}$
	5	$0.76464 \cdot 10^{-8}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.98312 \cdot 10^{-3}$
	10	$0.45221 \cdot 10^{-15}$	$0.45557 \cdot 10^{-16}$	$0.80728 \cdot 10^{-16}$	$0.38392 \cdot 10^{-16}$	$0.52163 \cdot 10^{-5}$
2	1	$0.21825 \cdot 10^{-2}$	$0.42567 \cdot 10^{-2}$	$0.42571 \cdot 10^{-2}$	$0.42573 \cdot 10^{-2}$	$0.94627 \cdot 10^{-2}$
	2	$0.52210 \cdot 10^{-6}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.40164 \cdot 10^{-3}$
	3	$0.85727 \cdot 10^{-10}$	$0.28196 \cdot 10^{-9}$	$0.27972 \cdot 10^{-9}$	$0.27937 \cdot 10^{-9}$	$0.50707 \cdot 10^{-4}$
	4	$0.19581 \cdot 10^{-13}$	$0.31330 \cdot 10^{-12}$	$0.10004 \cdot 10^{-12}$	$0.87396 \cdot 10^{-13}$	$0.50515 \cdot 10^{-5}$
	5	$0.14869 \cdot 10^{-16}$	$0.45261 \cdot 10^{-15}$	$0.60726 \cdot 10^{-16}$	$0.24530 \cdot 10^{-16}$	$0.69688 \cdot 10^{-6}$
	10	$0.23540 \cdot 10^{-26}$	$0.21185 \cdot 10^{-19}$	$0.17954 \cdot 10^{-19}$	$0.20190 \cdot 10^{-19}$	$0.46533 \cdot 10^{-9}$
Lvls = 4, NFine = 29791, NCoarse = 27, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.39956 \cdot 10^{-1}$	$0.72774 \cdot 10^{-1}$	$0.72776 \cdot 10^{-1}$	$0.97726 \cdot 10^{-1}$	0.1867057169
	2	$0.22781 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.19440 \cdot 10^{-1}$
	3	$0.32745 \cdot 10^{-4}$	$0.76292 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.31491 \cdot 10^{-2}$
	4	$0.53360 \cdot 10^{-6}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.12095 \cdot 10^{-2}$
	5	$0.76464 \cdot 10^{-8}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.47354 \cdot 10^{-3}$
	10	$0.44894 \cdot 10^{-17}$	$0.73339 \cdot 10^{-16}$	$0.43861 \cdot 10^{-16}$	$0.76619 \cdot 10^{-16}$	$0.43801 \cdot 10^{-5}$
2	1	$0.21843 \cdot 10^{-2}$	$0.42567 \cdot 10^{-2}$	$0.42571 \cdot 10^{-2}$	$0.42572 \cdot 10^{-2}$	$0.15604 \cdot 10^{-1}$
	2	$0.52210 \cdot 10^{-6}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.48879 \cdot 10^{-3}$
	3	$0.85727 \cdot 10^{-10}$	$0.28322 \cdot 10^{-9}$	$0.28063 \cdot 10^{-9}$	$0.27935 \cdot 10^{-9}$	$0.76028 \cdot 10^{-4}$
	4	$0.19535 \cdot 10^{-13}$	$0.33968 \cdot 10^{-12}$	$0.12704 \cdot 10^{-12}$	$0.87305 \cdot 10^{-13}$	$0.11685 \cdot 10^{-4}$
	5	$0.39948 \cdot 10^{-17}$	$0.38703 \cdot 10^{-15}$	$0.18923 \cdot 10^{-15}$	$0.30853 \cdot 10^{-16}$	$0.17964 \cdot 10^{-5}$
	10	$0.64462 \cdot 10^{-31}$	$0.28575 \cdot 10^{-16}$	$0.18990 \cdot 10^{-16}$	$0.54995 \cdot 10^{-17}$	$0.15441 \cdot 10^{-9}$
Lvls = 3, NFine = 29791, NCoarse = 343, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.39687 \cdot 10^{-1}$	$0.71641 \cdot 10^{-1}$	$0.71644 \cdot 10^{-1}$	$0.71656 \cdot 10^{-1}$	$0.97039 \cdot 10^{-1}$
	2	$0.22757 \cdot 10^{-2}$	$0.45634 \cdot 10^{-2}$	$0.45634 \cdot 10^{-2}$	$0.45635 \cdot 10^{-2}$	$0.67757 \cdot 10^{-2}$
	3	$0.32745 \cdot 10^{-4}$	$0.76292 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.76293 \cdot 10^{-4}$	$0.12983 \cdot 10^{-3}$
	4	$0.53360 \cdot 10^{-6}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.14788 \cdot 10^{-5}$	$0.28500 \cdot 10^{-5}$
	5	$0.76464 \cdot 10^{-8}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.24110 \cdot 10^{-7}$	$0.51928 \cdot 10^{-7}$
	10	$0.44894 \cdot 10^{-17}$	$0.62019 \cdot 10^{-16}$	$0.33795 \cdot 10^{-16}$	$0.28857 \cdot 10^{-16}$	$0.10870 \cdot 10^{-15}$
2	1	$0.21843 \cdot 10^{-2}$	$0.42567 \cdot 10^{-2}$	$0.42570 \cdot 10^{-2}$	$0.42572 \cdot 10^{-2}$	$0.61919 \cdot 10^{-2}$
	2	$0.52210 \cdot 10^{-6}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.14352 \cdot 10^{-5}$	$0.27464 \cdot 10^{-5}$
	3	$0.85727 \cdot 10^{-10}$	$0.28461 \cdot 10^{-9}$	$0.28173 \cdot 10^{-9}$	$0.27938 \cdot 10^{-9}$	$0.64873 \cdot 10^{-9}$
	4	$0.19535 \cdot 10^{-13}$	$0.31547 \cdot 10^{-13}$	$0.11360 \cdot 10^{-12}$	$0.87354 \cdot 10^{-12}$	$0.25379 \cdot 10^{-12}$
	5	$0.39948 \cdot 10^{-17}$	$0.62357 \cdot 10^{-15}$	$0.21039 \cdot 10^{-15}$	$0.35207 \cdot 10^{-16}$	$0.11873 \cdot 10^{-15}$
	10	$0.66504 \cdot 10^{-31}$	$0.32661 \cdot 10^{-22}$	$0.79563 \cdot 10^{-26}$	$0.19303 \cdot 10^{-22}$	$0.25208 \cdot 10^{-25}$
Lvls = 2, NFine = 29791, NCoarse = 3375, $\nu_0 = 100$, $\nu_{coarse} = 1000$						
1	1	$0.39700 \cdot 10^{-1}$	$0.72378 \cdot 10^{-1}$	$0.72378 \cdot 10^{-1}$	$0.72378 \cdot 10^{-1}$	$0.99144 \cdot 10^{-1}$
	2	$0.22781 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.45793 \cdot 10^{-2}$	$0.68194 \cdot 10^{-2}$
	3	$0.32312 \cdot 10^{-4}$	$0.73423 \cdot 10^{-4}$	$0.73423 \cdot 10^{-4}$	$0.73423 \cdot 10^{-4}$	$0.12174 \cdot 10^{-3}$
	4	$0.53015 \cdot 10^{-6}$	$0.14496 \cdot 10^{-5}$	$0.14496 \cdot 10^{-5}$	$0.14496 \cdot 10^{-5}$	$0.27490 \cdot 10^{-5}$
	5	$0.75984 \cdot 10^{-8}$	$0.23622 \cdot 10^{-7}$	$0.23622 \cdot 10^{-7}$	$0.23622 \cdot 10^{-7}$	$0.49987 \cdot 10^{-7}$
	10	$0.44587 \cdot 10^{-17}$	$0.28209 \cdot 10^{-16}$	$0.28209 \cdot 10^{-16}$	$0.28209 \cdot 10^{-16}$	$0.10411 \cdot 10^{-15}$
2	1	$0.21859 \cdot 10^{-2}$	$0.42678 \cdot 10^{-2}$	$0.42678 \cdot 10^{-2}$	$0.42678 \cdot 10^{-2}$	$0.62206 \cdot 10^{-2}$
	2	$0.51935 \cdot 10^{-6}$	$0.14153 \cdot 10^{-5}$	$0.14153 \cdot 10^{-5}$	$0.14153 \cdot 10^{-5}$	$0.26837 \cdot 10^{-5}$
	3	$0.84032 \cdot 10^{-10}$	$0.26284 \cdot 10^{-9}$	$0.26284 \cdot 10^{-9}$	$0.26284 \cdot 10^{-9}$	$0.58309 \cdot 10^{-9}$
	4	$0.19277 \cdot 10^{-13}$	$0.83157 \cdot 10^{-13}$	$0.83157 \cdot 10^{-13}$	$0.83157 \cdot 10^{-13}$	$0.23354 \cdot 10^{-12}$
	5	$0.39426 \cdot 10^{-17}$	$0.22772 \cdot 10^{-16}$	$0.22772 \cdot 10^{-16}$	$0.22772 \cdot 10^{-16}$	$0.79539 \cdot 10^{-16}$
	10	$0.75781 \cdot 10^{-31}$	$0.11026 \cdot 10^{-27}$	$0.53513 \cdot 10^{-28}$	$0.25093 \cdot 10^{-28}$	$0.38020 \cdot 10^{-17}$

Table 4.2. Accuracy of computed eigenpairs $\{\mu_i, \mathbf{v}_i\}$ as a function of ν and q, $p = 5$, Laplace problem

Based on the results we have to conclude that the FMG-EV(p) method have the similar behaviour as the FMG-EV method except few remarks, when the coarse level problems are solved exactly.

First of all, in two- and three-level cases we have surprisely a good convergence for the last eigenvalue λ_5 with respect to disconvergence for four- and five-level cases. It can be intuitively explained by the coarse level problem is not sufficiently reach to generate a good approximation to these eigenspaces.

Moreover, compare the results from Tables 4.1 and 4.2 for the first eigenpair $\{\lambda_1, \mathbf{v}_1\}$ with ones from Table 3.2 one can see that when more eigenvectors, whether or not they converge, are included in the process, all approximations are improved.

q	ν	$\lvert\lambda_1-\mu_1\rvert$	$\lvert\lambda_2-\mu_2\rvert$	$\lvert\lambda_3-\mu_3\rvert$	$\lvert\lambda_4-\mu_4\rvert$	$\lvert\lambda_5-\mu_5\rvert$
Lvls = 4, NFine = 10125, NCoarse = 3						
1	1	$0.17121 \cdot 10^{-2}$	$0.18504 \cdot 10^{-2}$	$0.23162 \cdot 10^{-2}$	$0.21263 \cdot 10^{-2}$	$0.24166 \cdot 10^{-2}$
	2	$0.87427 \cdot 10^{-4}$	$0.18120 \cdot 10^{-3}$	$0.18459 \cdot 10^{-3}$	$0.14426 \cdot 10^{-3}$	$0.16316 \cdot 10^{-3}$
	3	$0.59710 \cdot 10^{-5}$	$0.15262 \cdot 10^{-4}$	$0.23290 \cdot 10^{-4}$	$0.16597 \cdot 10^{-4}$	$0.20785 \cdot 10^{-4}$
	4	$0.64233 \cdot 10^{-6}$	$0.18220 \cdot 10^{-5}$	$0.27387 \cdot 10^{-5}$	$0.22954 \cdot 10^{-5}$	$0.26639 \cdot 10^{-5}$
	5	$0.62436 \cdot 10^{-7}$	$0.27586 \cdot 10^{-6}$	$0.33646 \cdot 10^{-6}$	$0.29719 \cdot 10^{-6}$	$0.81938 \cdot 10^{-6}$
	10	$0.12715 \cdot 10^{-8}$	$0.16685 \cdot 10^{-8}$	$0.54289 \cdot 10^{-8}$	$0.13347 \cdot 10^{-6}$	$0.34387 \cdot 10^{-6}$
2	1	$0.66509 \cdot 10^{-4}$	$0.13746 \cdot 10^{-3}$	$0.14220 \cdot 10^{-3}$	$0.11944 \cdot 10^{-3}$	$0.13899 \cdot 10^{-3}$
	2	$0.56575 \cdot 10^{-6}$	$0.17661 \cdot 10^{-5}$	$0.25094 \cdot 10^{-5}$	$0.16908 \cdot 10^{-5}$	$0.20703 \cdot 10^{-5}$
	3	$0.52939 \cdot 10^{-8}$	$0.36231 \cdot 10^{-7}$	$0.47925 \cdot 10^{-7}$	$0.13661 \cdot 10^{-6}$	$0.40596 \cdot 10^{-6}$
	4	$0.52667 \cdot 10^{-9}$	$0.87679 \cdot 10^{-9}$	$0.23866 \cdot 10^{-8}$	$0.69039 \cdot 10^{-7}$	$0.18090 \cdot 10^{-6}$
	5	$0.92139 \cdot 10^{-10}$	$0.16693 \cdot 10^{-9}$	$0.27081 \cdot 10^{-8}$	$0.17478 \cdot 10^{-7}$	$0.11907 \cdot 10^{-6}$
	10	$0.11360 \cdot 10^{-9}$	$0.11232 \cdot 10^{-9}$	$0.42050 \cdot 10^{-10}$	$0.65845 \cdot 10^{-8}$	$0.18503 \cdot 10^{-7}$
Lvls = 3, NFine = 10125, NCoarse = 81						
1	1	$0.17126 \cdot 10^{-2}$	$0.18520 \cdot 10^{-2}$	$0.23164 \cdot 10^{-2}$	$0.21153 \cdot 10^{-2}$	$0.22651 \cdot 10^{-2}$
	2	$0.86727 \cdot 10^{-4}$	$0.18121 \cdot 10^{-3}$	$0.18583 \cdot 10^{-3}$	$0.15580 \cdot 10^{-3}$	$0.16274 \cdot 10^{-3}$
	3	$0.59921 \cdot 10^{-5}$	$0.15343 \cdot 10^{-4}$	$0.23375 \cdot 10^{-4}$	$0.12978 \cdot 10^{-4}$	$0.16919 \cdot 10^{-4}$
	4	$0.63646 \cdot 10^{-6}$	$0.18893 \cdot 10^{-5}$	$0.27678 \cdot 10^{-5}$	$0.22854 \cdot 10^{-5}$	$0.28454 \cdot 10^{-5}$
	5	$0.70249 \cdot 10^{-7}$	$0.27376 \cdot 10^{-6}$	$0.34354 \cdot 10^{-6}$	$0.38226 \cdot 10^{-6}$	$0.91077 \cdot 10^{-6}$
	10	$0.13113 \cdot 10^{-8}$	$0.20920 \cdot 10^{-8}$	$0.71929 \cdot 10^{-8}$	$0.16827 \cdot 10^{-6}$	$0.23884 \cdot 10^{-6}$
2	1	$0.66521 \cdot 10^{-4}$	$0.13756 \cdot 10^{-3}$	$0.14223 \cdot 10^{-3}$	$0.11527 \cdot 10^{-3}$	$0.13875 \cdot 10^{-3}$
	2	$0.56377 \cdot 10^{-6}$	$0.17248 \cdot 10^{-5}$	$0.25207 \cdot 10^{-5}$	$0.15277 \cdot 10^{-5}$	$0.22036 \cdot 10^{-5}$
	3	$0.68877 \cdot 10^{-8}$	$0.36520 \cdot 10^{-7}$	$0.81525 \cdot 10^{-7}$	$0.82430 \cdot 10^{-7}$	$0.36574 \cdot 10^{-6}$
	4	$0.78704 \cdot 10^{-9}$	$0.22278 \cdot 10^{-8}$	$0.15379 \cdot 10^{-7}$	$0.86195 \cdot 10^{-7}$	$0.16180 \cdot 10^{-6}$
	5	$0.83009 \cdot 10^{-10}$	$0.23116 \cdot 10^{-9}$	$0.94830 \cdot 10^{-8}$	$0.62724 \cdot 10^{-7}$	$0.87238 \cdot 10^{-7}$
	10	$0.11371 \cdot 10^{-9}$	$0.11276 \cdot 10^{-9}$	$0.36793 \cdot 10^{-9}$	$0.69298 \cdot 10^{-8}$	$0.12069 \cdot 10^{-7}$
Lvls = 2, NFine = 10125, NCoarse = 1029						
1	1	$0.16598 \cdot 10^{-2}$	$0.17743 \cdot 10^{-2}$	$0.22529 \cdot 10^{-2}$	$0.20921 \cdot 10^{-2}$	$0.22736 \cdot 10^{-2}$
	2	$0.89283 \cdot 10^{-4}$	$0.17879 \cdot 10^{-3}$	$0.18547 \cdot 10^{-3}$	$0.14831 \cdot 10^{-3}$	$0.16975 \cdot 10^{-3}$
	3	$0.68256 \cdot 10^{-5}$	$0.15841 \cdot 10^{-4}$	$0.23290 \cdot 10^{-4}$	$0.15277 \cdot 10^{-4}$	$0.19748 \cdot 10^{-4}$
	4	$0.10038 \cdot 10^{-5}$	$0.21974 \cdot 10^{-5}$	$0.32588 \cdot 10^{-5}$	$0.17504 \cdot 10^{-5}$	$0.24626 \cdot 10^{-5}$
	5	$0.16654 \cdot 10^{-6}$	$0.33157 \cdot 10^{-6}$	$0.22759 \cdot 10^{-5}$	$0.19340 \cdot 10^{-6}$	$0.65046 \cdot 10^{-6}$
	10	$0.13450 \cdot 10^{-8}$	$0.56302 \cdot 10^{-8}$	$0.20701 \cdot 10^{-5}$	$0.18521 \cdot 10^{-7}$	$0.10047 \cdot 10^{-6}$
2	1	$0.66248 \cdot 10^{-4}$	$0.13791 \cdot 10^{-3}$	$0.14184 \cdot 10^{-3}$	$0.11067 \cdot 10^{-3}$	$0.15442 \cdot 10^{-3}$
	2	$0.56097 \cdot 10^{-6}$	$0.18273 \cdot 10^{-5}$	$0.56997 \cdot 10^{-5}$	$0.16318 \cdot 10^{-5}$	$0.25670 \cdot 10^{-5}$
	3	$0.25852 \cdot 10^{-7}$	$0.37555 \cdot 10^{-7}$	$0.17063 \cdot 10^{-5}$	$0.26033 \cdot 10^{-7}$	$0.30782 \cdot 10^{-7}$
	4	$0.13821 \cdot 10^{-8}$	$0.25116 \cdot 10^{-8}$	$0.94665 \cdot 10^{-6}$	$0.46884 \cdot 10^{-8}$	$0.39074 \cdot 10^{-7}$
	5	$0.14959 \cdot 10^{-9}$	$0.46594 \cdot 10^{-8}$	$0.15020 \cdot 10^{-5}$	$0.28829 \cdot 10^{-7}$	$0.83986 \cdot 10^{-7}$
	10	$0.11348 \cdot 10^{-9}$	$0.28841 \cdot 10^{-10}$	$0.40120 \cdot 10^{-6}$	$0.18245 \cdot 10^{-8}$	$0.48712 \cdot 10^{-8}$

Table 4.6. Accuracy of computed eigenvalues μ_i as a function of ν and q, $p = 5$, elasticity problem

Similar to the FMG-EV method for the first eigenvalue one can see that the accuracy for all approximated eigenvalues is improved, when the number of relaxation steps ν and the number of inner iterations q are increasing. Taking into account the whole comutational complexity, which is also very fast growing when ν and q are increasing, the optimal choice of the iterative parameters, in practice, seems to be $q = 1$ and $\nu = 2$ or close to these values. Here we have to note that the later result is *identical* to the result for the first eigenvalue.

Now we have to define the optimal values for ε_{coarse} and ε_0. To do this we made a series of runs for different combinations of ε_{coarse} and ε_0 under fixed values of $q = 1$ and $\nu = 2$. The corresponding results for different number of levels used are presented in Figures 4.3.1–4.3.4.

q	ν	$\|\mathbf{A}\mathbf{v}_1 - \mu_1\mathbf{v}_1\|$	$\|\mathbf{A}\mathbf{v}_2 - \mu_2\mathbf{v}_2\|$	$\|\mathbf{A}\mathbf{v}_3 - \mu_3\mathbf{v}_3\|$	$\|\mathbf{A}\mathbf{v}_4 - \mu_4\mathbf{v}_4\|$	$\|\mathbf{A}\mathbf{v}_5 - \mu_5\mathbf{v}_5\|$
Lvls = 4, NFine = 10125, NCoarse = 3						
1	1	$0.35633 \cdot 10^{-1}$	$0.35849 \cdot 10^{-1}$	$0.48551 \cdot 10^{-1}$	$0.40670 \cdot 10^{-1}$	$0.46340 \cdot 10^{-1}$
	2	$0.69620 \cdot 10^{-2}$	$0.11551 \cdot 10^{-1}$	$0.10216 \cdot 10^{-1}$	$0.96870 \cdot 10^{-2}$	$0.96702 \cdot 10^{-2}$
	3	$0.16195 \cdot 10^{-2}$	$0.23696 \cdot 10^{-2}$	$0.31153 \cdot 10^{-2}$	$0.25545 \cdot 10^{-2}$	$0.28774 \cdot 10^{-2}$
	4	$0.54673 \cdot 10^{-3}$	$0.73684 \cdot 10^{-3}$	$0.10033 \cdot 10^{-2}$	$0.81480 \cdot 10^{-3}$	$0.88640 \cdot 10^{-3}$
	5	$0.15609 \cdot 10^{-3}$	$0.28865 \cdot 10^{-3}$	$0.33740 \cdot 10^{-3}$	$0.27417 \cdot 10^{-3}$	$0.36506 \cdot 10^{-3}$
	10	$0.12719 \cdot 10^{-4}$	$0.14326 \cdot 10^{-4}$	$0.17823 \cdot 10^{-4}$	$0.84203 \cdot 10^{-4}$	$0.13971 \cdot 10^{-3}$
2	1	$0.61016 \cdot 10^{-2}$	$0.89436 \cdot 10^{-2}$	$0.98460 \cdot 10^{-2}$	$0.83752 \cdot 10^{-2}$	$0.91747 \cdot 10^{-2}$
	2	$0.51795 \cdot 10^{-3}$	$0.72641 \cdot 10^{-3}$	$0.96730 \cdot 10^{-3}$	$0.71808 \cdot 10^{-3}$	$0.83226 \cdot 10^{-3}$
	3	$0.42454 \cdot 10^{-4}$	$0.10305 \cdot 10^{-3}$	$0.11552 \cdot 10^{-3}$	$0.11879 \cdot 10^{-3}$	$0.17028 \cdot 10^{-3}$
	4	$0.86842 \cdot 10^{-5}$	$0.15120 \cdot 10^{-4}$	$0.19860 \cdot 10^{-4}$	$0.59069 \cdot 10^{-4}$	$0.10028 \cdot 10^{-3}$
	5	$0.53047 \cdot 10^{-5}$	$0.55883 \cdot 10^{-5}$	$0.10770 \cdot 10^{-4}$	$0.31108 \cdot 10^{-4}$	$0.77286 \cdot 10^{-4}$
	10	$0.87844 \cdot 10^{-6}$	$0.14166 \cdot 10^{-5}$	$0.19605 \cdot 10^{-5}$	$0.16655 \cdot 10^{-4}$	$0.31198 \cdot 10^{-4}$
Lvls = 3, NFine = 10125, NCoarse = 81						
1	1	$0.35638 \cdot 10^{-1}$	$0.35845 \cdot 10^{-1}$	$0.48545 \cdot 10^{-1}$	$0.39561 \cdot 10^{-1}$	$0.45533 \cdot 10^{-1}$
	2	$0.69323 \cdot 10^{-2}$	$0.11575 \cdot 10^{-1}$	$0.10219 \cdot 10^{-1}$	$0.97138 \cdot 10^{-2}$	$0.98802 \cdot 10^{-2}$
	3	$0.16205 \cdot 10^{-2}$	$0.23698 \cdot 10^{-2}$	$0.31158 \cdot 10^{-2}$	$0.22178 \cdot 10^{-2}$	$0.26073 \cdot 10^{-2}$
	4	$0.54086 \cdot 10^{-3}$	$0.74379 \cdot 10^{-3}$	$0.10048 \cdot 10^{-2}$	$0.88062 \cdot 10^{-3}$	$0.81750 \cdot 10^{-3}$
	5	$0.15830 \cdot 10^{-3}$	$0.28768 \cdot 10^{-3}$	$0.33659 \cdot 10^{-3}$	$0.28437 \cdot 10^{-3}$	$0.34140 \cdot 10^{-3}$
	10	$0.12587 \cdot 10^{-4}$	$0.16523 \cdot 10^{-4}$	$0.17521 \cdot 10^{-4}$	$0.98736 \cdot 10^{-4}$	$0.10736 \cdot 10^{-3}$
2	1	$0.61022 \cdot 10^{-2}$	$0.89445 \cdot 10^{-2}$	$0.98459 \cdot 10^{-2}$	$0.82619 \cdot 10^{-2}$	$0.91492 \cdot 10^{-2}$
	2	$0.52500 \cdot 10^{-3}$	$0.71841 \cdot 10^{-3}$	$0.96738 \cdot 10^{-3}$	$0.66363 \cdot 10^{-3}$	$0.82692 \cdot 10^{-3}$
	3	$0.46287 \cdot 10^{-4}$	$0.10465 \cdot 10^{-3}$	$0.11551 \cdot 10^{-3}$	$0.99123 \cdot 10^{-4}$	$0.16709 \cdot 10^{-3}$
	4	$0.13839 \cdot 10^{-4}$	$0.19808 \cdot 10^{-4}$	$0.21173 \cdot 10^{-4}$	$0.66386 \cdot 10^{-4}$	$0.94793 \cdot 10^{-4}$
	5	$0.52278 \cdot 10^{-5}$	$0.62587 \cdot 10^{-5}$	$0.16354 \cdot 10^{-4}$	$0.57611 \cdot 10^{-4}$	$0.66844 \cdot 10^{-4}$
	10	$0.70692 \cdot 10^{-6}$	$0.79362 \cdot 10^{-6}$	$0.41566 \cdot 10^{-5}$	$0.19018 \cdot 10^{-4}$	$0.23745 \cdot 10^{-4}$
Lvls = 2, NFine = 10125, NCoarse = 1029						
1	1	$0.35119 \cdot 10^{-1}$	$0.35115 \cdot 10^{-1}$	$0.47902 \cdot 10^{-1}$	$0.39215 \cdot 10^{-1}$	$0.45660 \cdot 10^{-1}$
	2	$0.70299 \cdot 10^{-2}$	$0.11518 \cdot 10^{-1}$	$0.10113 \cdot 10^{-1}$	$0.96555 \cdot 10^{-2}$	$0.10124 \cdot 10^{-1}$
	3	$0.16783 \cdot 10^{-2}$	$0.23134 \cdot 10^{-2}$	$0.31116 \cdot 10^{-2}$	$0.25111 \cdot 10^{-2}$	$0.28154 \cdot 10^{-2}$
	4	$0.58759 \cdot 10^{-3}$	$0.79815 \cdot 10^{-3}$	$0.94454 \cdot 10^{-3}$	$0.69161 \cdot 10^{-3}$	$0.91486 \cdot 10^{-3}$
	5	$0.21834 \cdot 10^{-3}$	$0.33317 \cdot 10^{-3}$	$0.33348 \cdot 10^{-3}$	$0.26339 \cdot 10^{-3}$	$0.35363 \cdot 10^{-3}$
	10	$0.13607 \cdot 10^{-4}$	$0.25534 \cdot 10^{-4}$	$0.22342 \cdot 10^{-3}$	$0.30763 \cdot 10^{-4}$	$0.72578 \cdot 10^{-4}$
2	1	$0.60697 \cdot 10^{-2}$	$0.89510 \cdot 10^{-2}$	$0.98347 \cdot 10^{-2}$	$0.81740 \cdot 10^{-2}$	$0.96936 \cdot 10^{-2}$
	2	$0.52357 \cdot 10^{-3}$	$0.74812 \cdot 10^{-3}$	$0.98926 \cdot 10^{-3}$	$0.79192 \cdot 10^{-3}$	$0.90675 \cdot 10^{-3}$
	3	$0.88952 \cdot 10^{-4}$	$0.10400 \cdot 10^{-3}$	$0.21830 \cdot 10^{-3}$	$0.89080 \cdot 10^{-4}$	$0.86215 \cdot 10^{-4}$
	4	$0.15535 \cdot 10^{-4}$	$0.18265 \cdot 10^{-4}$	$0.14572 \cdot 10^{-3}$	$0.19477 \cdot 10^{-4}$	$0.45988 \cdot 10^{-4}$
	5	$0.58039 \cdot 10^{-5}$	$0.21082 \cdot 10^{-4}$	$0.18240 \cdot 10^{-3}$	$0.40298 \cdot 10^{-4}$	$0.63035 \cdot 10^{-4}$
	10	$0.59914 \cdot 10^{-6}$	$0.40553 \cdot 10^{-5}$	$0.96543 \cdot 10^{-4}$	$0.74399 \cdot 10^{-5}$	$0.14618 \cdot 10^{-4}$

Table 4.7. Accuracy of computed eigenpairs $\{\mu_i, \mathbf{v}_i\}$ as a function of ν and q, $p = 5$, elasticity problem

Studying those Figures one can find that the pair of $\varepsilon_{coarse} = 10^{-4}$ and $\varepsilon_0 = 10^{-2}$ is still otimal from different points of view. Indeed, in this case we got a good accuracy for the desired eigenvalues, i.e., the error between exact and computed eigenvalues is nearly 10^{-6} for first fourth eigenvalues (at least), and also to obtain this accuracy we pay a reasonable computational costs.

Moreover, the present experiments shows that the FMG-EV(p) method is of capable to deal with a multiply eigenvalue. Indeed, from (18) we have

$$\lambda^h_{1,1,2} = \lambda^h_{1,2,1} = \lambda^h_{2,1,1} = 4\left[2\sin^2\left(\frac{\pi}{2N}\right) + \sin^2\left(\frac{\pi}{N}\right)\right],$$

and how one can see from the numerous numerical results presented here we find this eigenvalue and its eigensubspace without problems.

Additional to the case for one eigenvalue we have to show that the total computational complexity grows as $O(p^2)$. It is evidence from Figure 4.5, where "linear" and "theoretical" lines denotes the linear and quadratic function on p, respectively. How it is readily seen, the real computational complexity grwos as a function between linear and quadratic ones.

Finally, we want to mention a problem with the eigenvector orthogonalization for the inner nonlinear method. Indeed, when we try to solve the coarse level problem for the second, third or other eigenvectors during the FAS method with some given accuracy, we always perform the maximal number of iterations, i.e. we could not reach the desired accuracy. It seems that it is not a strong restriction on the suggested method since the method is still convergent. However, the proposed technique with corresponding block-type modifications for the inner nonlinear method can be done. The further investigation will be directed in this way.

The final group of experiments was done for the three dimensional linear elasticity problem with Dirichlet boundary conditions in the cube domain. Note that in this example we have multiply eigenvalues: $\lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 = \lambda_5$. The results for the FMG-EV(p) method with $\varepsilon_0 = 10^{-4}$ and $\nu_{coarse} = 100$, are presented in Tables 4.6 and 4.7. Based on the present results one can made the similar conlusions as for the Laplace operator. Indeed, we obtain

- the accuracy is always better when q and ν increase;
- the accuracy does not depend on the number of levels used;
- the time grows linearly with respect to n_L (exept two-level case), q and ν, and quadraticaly with respect to p;
- there are a limit accuracy for each pair q and ν;
- it is more important to find a good initial coarse level approximation, then try to improve it later by inner multilevel process;
- the method is capable to deal with multiply eigenvalues;
- when more eigenvectors, whether or not they converge, are included in the process, all approximations are improved.

Finally, we have to note that the present method can be used as a good competitor to the direct eigensolver when the small or medium number of eigenvalues required, i.e., $p < \sqrt{n_0}$, but if we would like to finding a series of eigenvectors with $p > \sqrt{n_0}$, then the Lanzos method with LU-factorization is better or, the same, less time consuming.

5 Conclusions

As a final conclusion, we have found that the FMG-EV(p) method, applied for computing the smallest eigenvalue and its eigenvector, leads to an iterative eigensolver with an optimal order of the computational complexity and an uniform convergence behaviour. However, there is a problem to keep orthogonalization between approximated eigenvectors during the inner multilevel process. It seems that it is not a strong restriction on the suggested method since the proposed technique with corresponding block-type modifications can be adopted for the problem of computing the number of smallest eigenvalues and their eigenvectors. As it was mentioned in first sections another open questions are the method for computing the sequence of matrices $A^{(k)}$ and also the proof of the convergence rate of the method. The future investigation will be directed in these ways.

Acknowledgments

The author would like to thank Dr. K. Stuben and C.-A. Thole from the Fraunhofer Institute for Algorithms and Scientific Computing, Sankt Augustin, Germany, for initiation of the present study and fruitful discussion of the first numerical results.

References

[1] 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.

[2] J. H. Bramble, J. E. Pasciak and A. V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math. 6(2) (1996), pp. 159–189.

[3] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] A. Brandt, S. McCormick and J. Ruge, *Multigrid methods for differential eigenproblems*, SIAM J. Sci. Stat. Comput., 4(2) (1983), pp. 244–260.

[5] A. Brandt, S. McCormick and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, In: "Sparsity and Its Applications" (D.J.Evans, ed.), Cambridge Univ. Press, Cambridge, 1984.

[6] V. E. Bulgakov, M. V. Belyi and K. M. Mathisen, *Multilevel aggregation method for solving large-scale generalized eigenvalue problems in structural dynamics*, Int. J. Numer. Methods Eng., 40(3) (1997), pp. 453–471.

[7] Z. Cai, J. Mandel and S. McCormick, *Multigrid methods for nearly singular linear equations and eigenvalue problems*, SIAM J. Numer. Anal., 34(1) (1997), pp. 178–200.

[8] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin - Heidelberg - New York, 1985.

[9] T. Hwang and I. D. Parsons, *A Multi-grid Method for the Generalized Symmetric Eigenvalue Problem: Part I–Algorithm and Implementation*, Int. J. Numer. Methods Eng., 35(8) (1992), pp. 1663–1676.

[10] T. Hwang and I. D. Parsons, *A Multi-grid Method for the Generalized Symmetric Eigenvalue Problem: Part II–Performance Evaluation*, Int. J. Numer. Methods Eng., 35(8) (1992), pp. 1677–1696.

[11] T. Hwang and I. D. Parsons, *Multigrid solution procedures for structural dynamics eigenvalue problems*, Comput. Mech. 10(3/4) (1992), pp. 247–262.

[12] T. Hwang and I. D. Parsons, *Parallel implementation and performance of multigrid algorithms for solving eigenvalue problems*, Comput. Struct. 50(3) (1994), pp. 325–336.

[13] A.V. Knyazev and A.L. Skorokhodov, *The preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problem*, SIAM J. Numerical Analysis, 31 (1994), pp. 1226–1239.

[14] A.V. Knyazev, *New estimates for Ritz vectors*, Math. Comp. 66 (1997), pp. 985–995.

[15] A. V. Knyazev, *Preconditioned eigensolvers – an oxymoron?*, Electronic Transaction on Numerical Analysis, Volume 7, 1998, pp. 104–123.

[16] A.V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, SIAM Journal on Scientific Computing, 23 (2001), pp. 517–541.

[17] A.V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration, III: A short and sharp convergence estimate for generalized eigenvalue problems*, April 2001. Accepted for Linear Algebra Appl.

[18] A.V. Knyazev and K. Neymeyr,*Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block preconditioned gradient method*, July 2001. Accepted for ETNA.

[19] A. I. Lur'e, *Theory of elasticity*, Moscow, Nauka, 1970.

[20] K. Neymeyr, *A geometric theory for preconditioned inverse iteration, I: Extrema of the Rayleigh quotient*, Linear Algebra Appl. 332 (2001), pp. 61–85.

[21] K. Neymeyr, *A geometric theory for preconditioned inverse iteration, II:Convergence estimates* , Linear Algebra Appl. 332 (2001), pp. 67–104.

[22] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp., 71 (2002), pp. 197–216.

[23] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Inc., 1980.

[24] A. Reusken, *Convergence of the multigrid full approximation scheme for a class of elliptic mildly nonlinear boundary value problems*, Numer. Math. 52 (1988), pp. 251–277.

[25] A. Reusken, *Convergence of the multilevel full approximation scheme including the V-cycle*, Numer. Math. 53 (1988), pp. 663–686.

[26] U. Trottenberg, C. Oosterlee and A. Schüller, *Multigrid*, Academic Press, 2001.

[27] P. Wesseling,*An introduction to multigrid methods*, John Wiley & Sons Ltd., 1992.

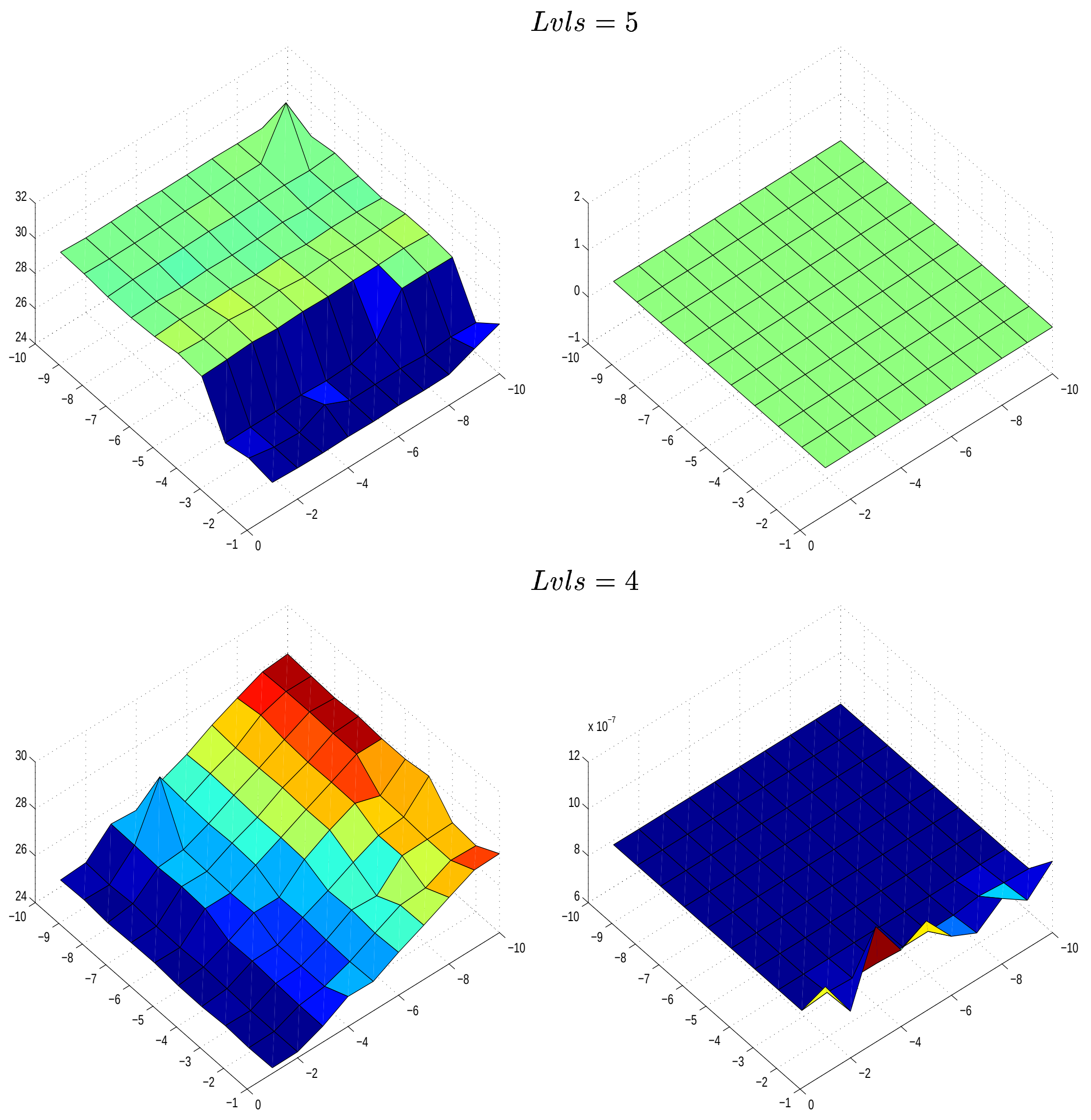

Fig. 3.3.1. The time of the whole iterative process and the accuracy of the first eigenvalue with respect to the inner stopping criteria ε_0 and ε_{coarse}, Laplace problem

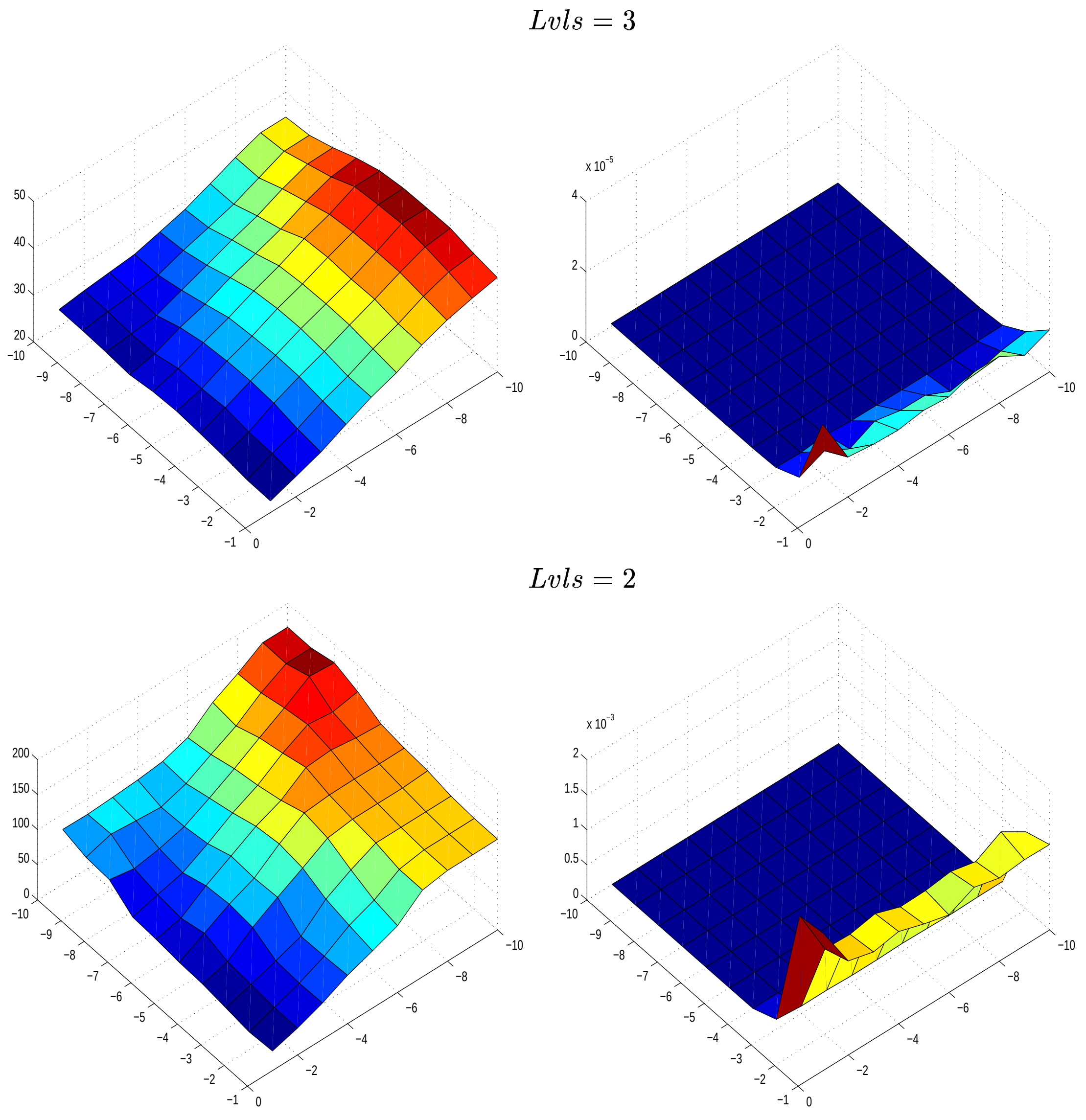

Fig. 3.3.2. The time of the whole iterative process and the accuracy of the first eigenvalue with respect to the inner stopping criteria ε_0 and ε_{coarse}, Laplace problem

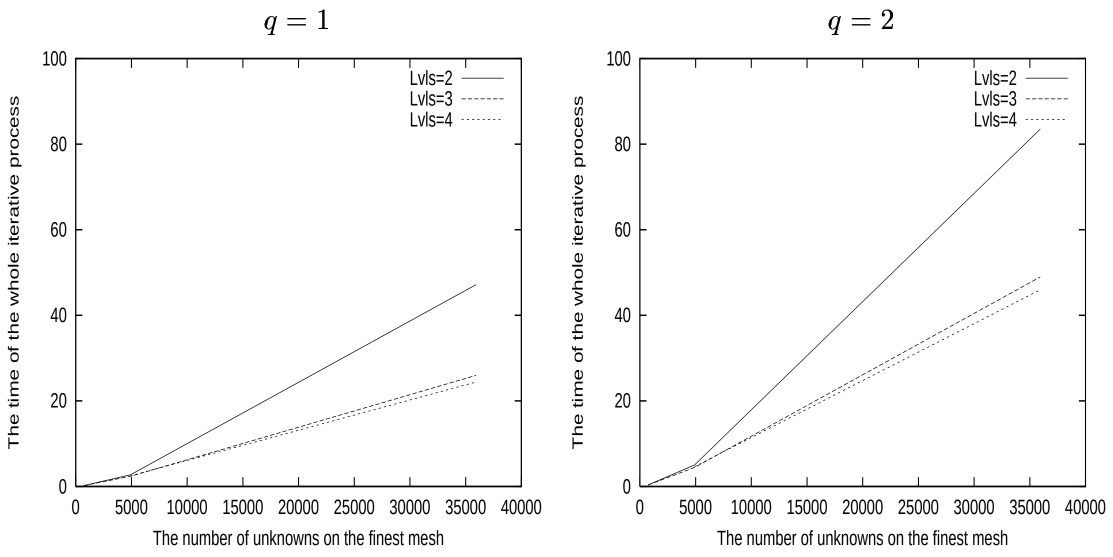

Fig. 3.4. The time of the whole iterative process for finding the first eigenpair with respect to the number of unknowuns on the finest level, Laplace problem

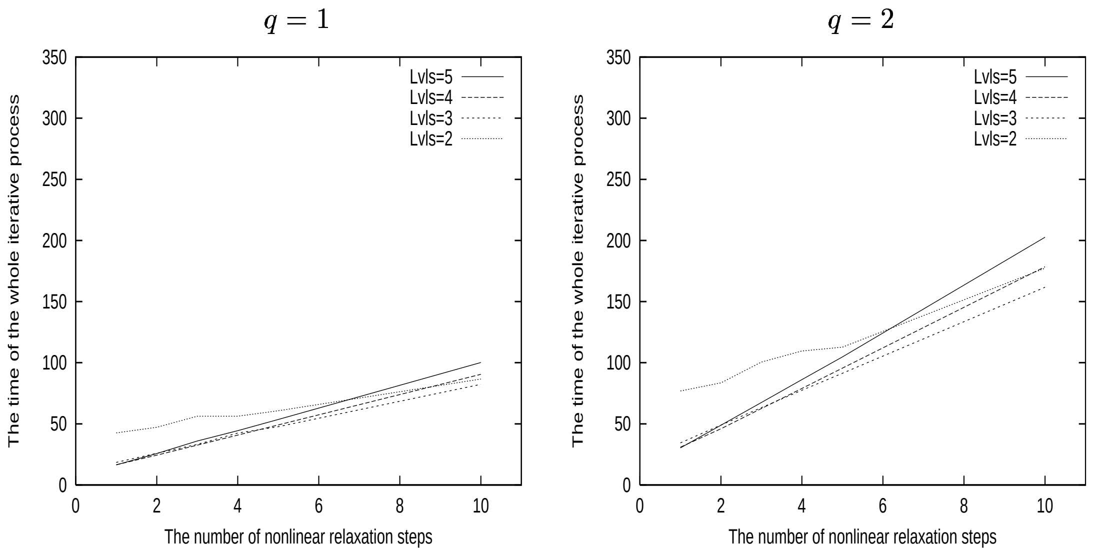

Fig. 3.5. The time of the whole iterative process for finding the first eigenpair with respect to the number of pre- and post- smoothing steps used, Laplace problem

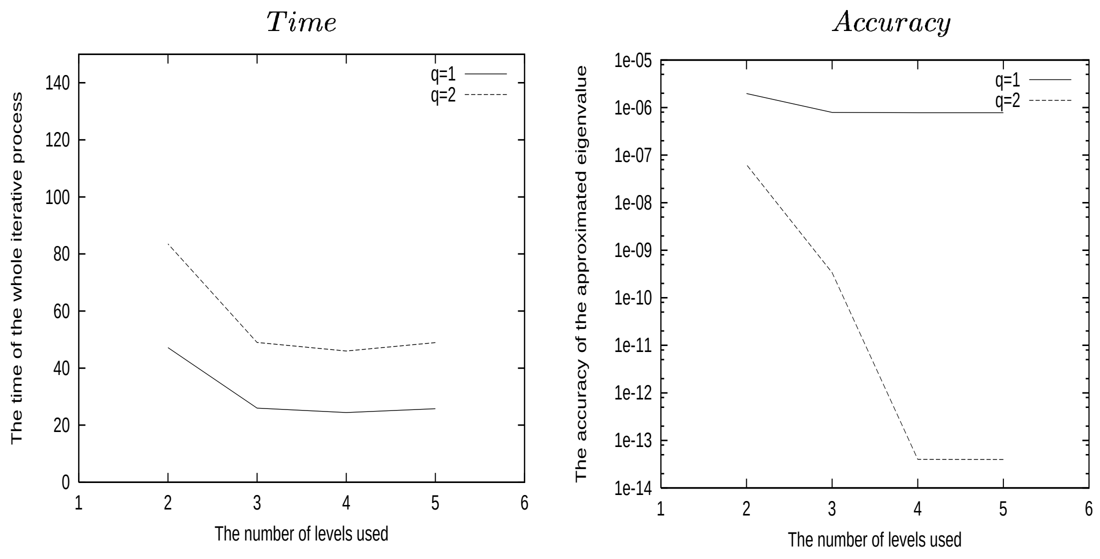

Fig. 3.6. The time of the whole iterative process and the accuracy for the first eigenvalue with respect to the number of level used, Laplace problem

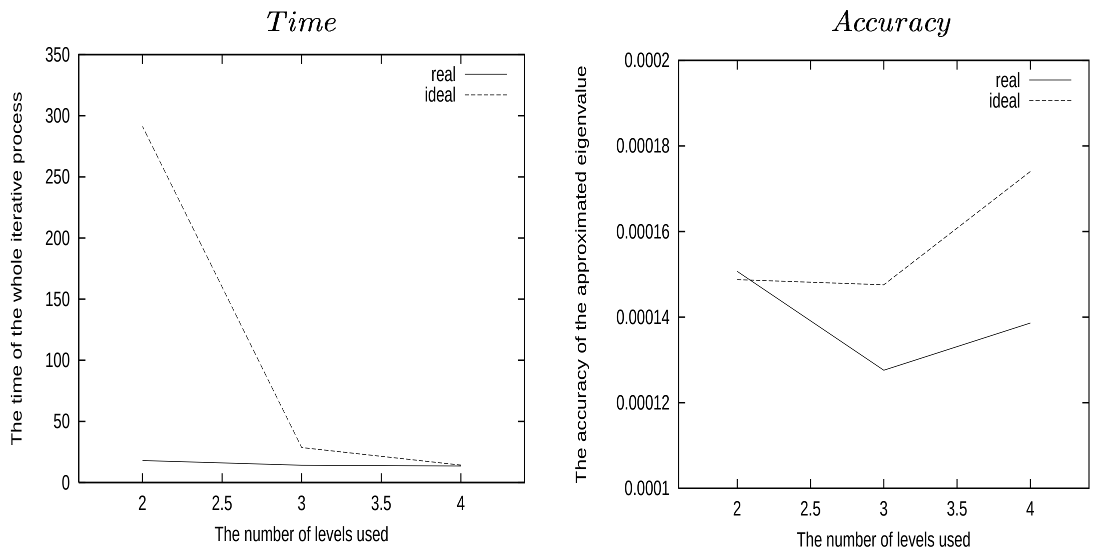

Fig. 3.9. The time of the whole iterative process and the accuracy for the first eigenvalue with respect to the number of level used, Laplace problem

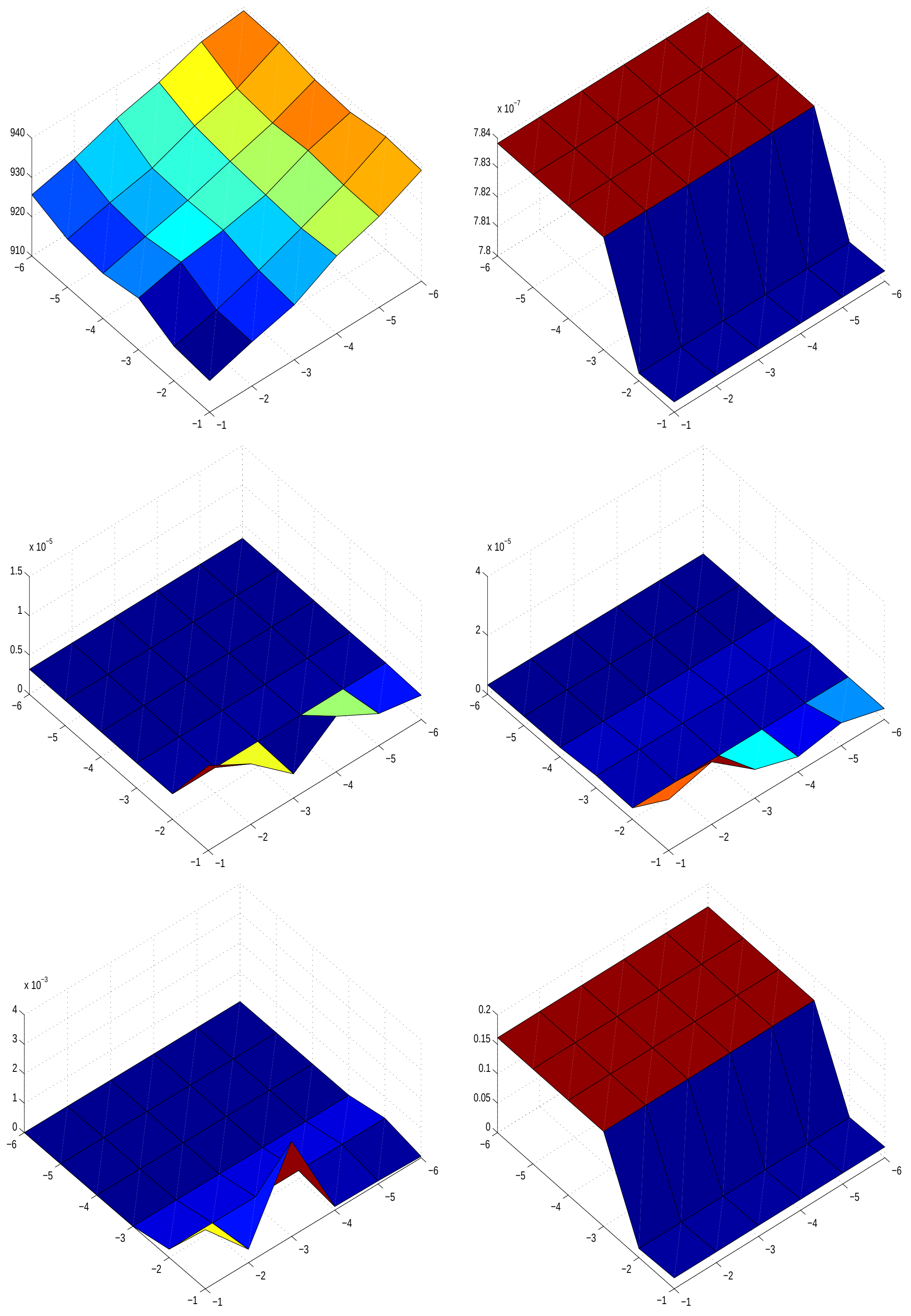

Fig. 4.3.1. The time of the whole iterative process and the accuracy for five smallest eigenvalues **vs.** the inner stopping criteria ε_0 and ε_{coarse}, $Llvs = 5$, Laplace problem

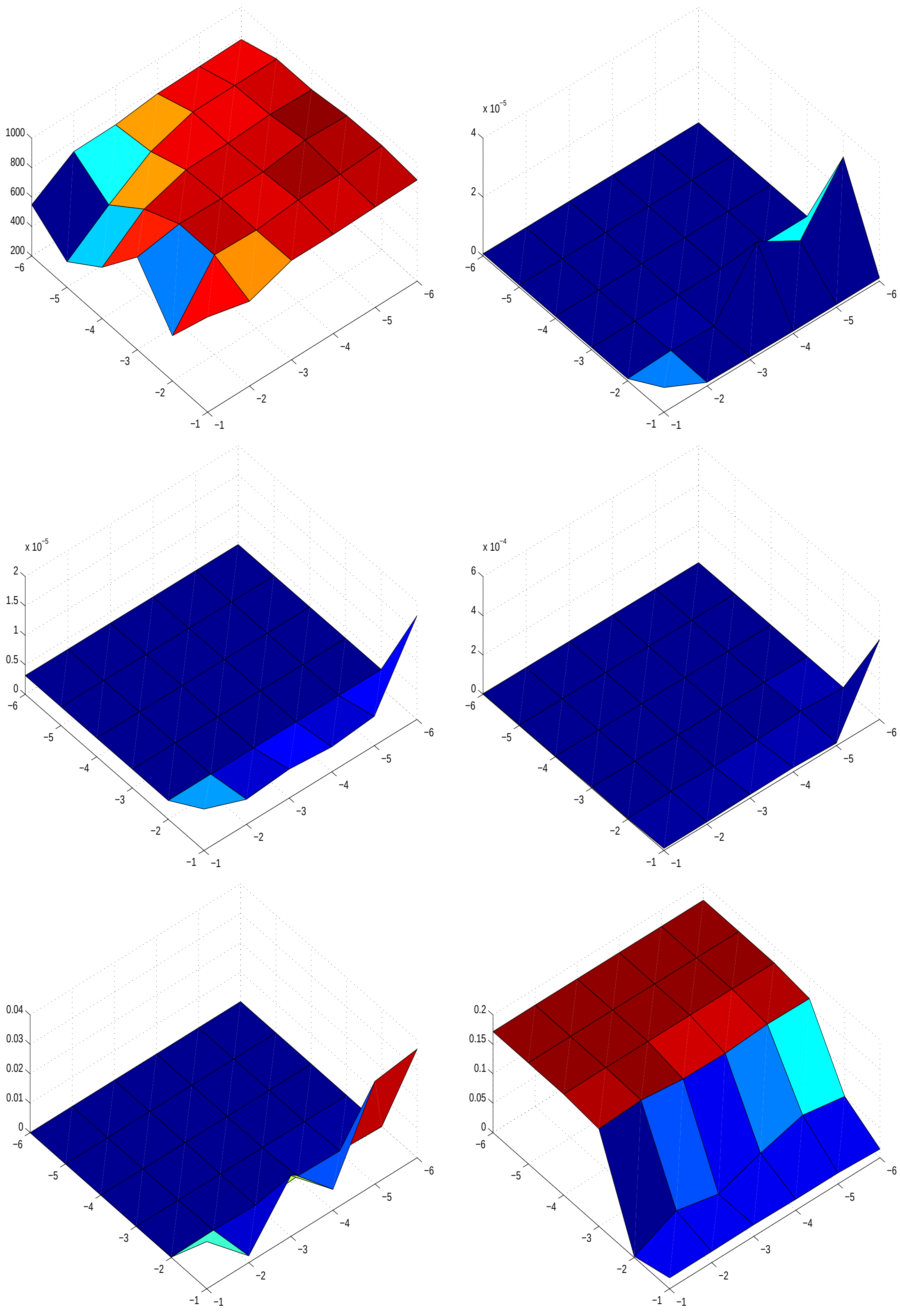

Fig. 4.3.2. The time of the whole iterative process and the accuracy for five smallest eigenvalues **vs.** the inner stopping criteria ε_0 and ε_{coarse}, $Llvs = 4$, Laplace problem

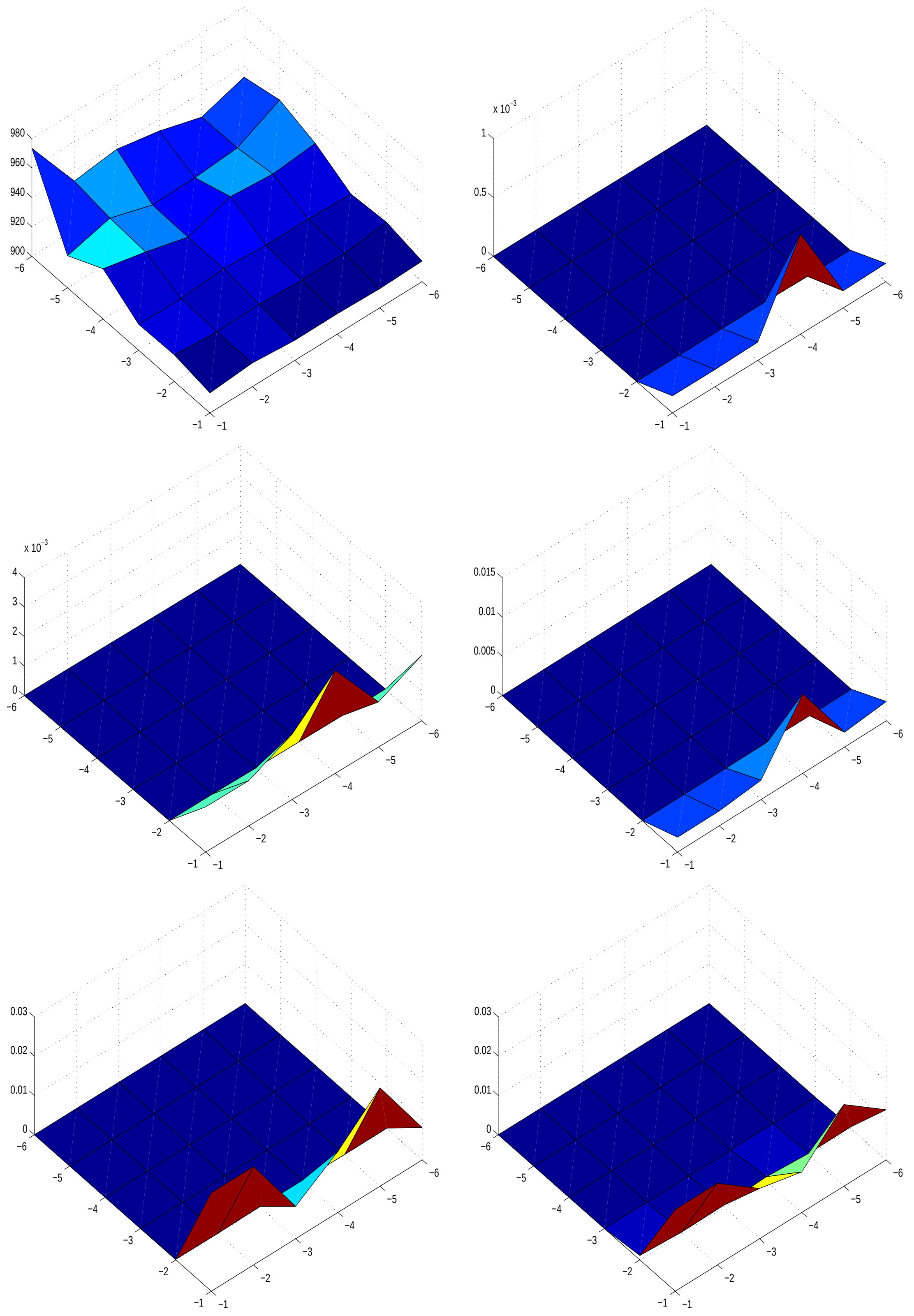

Fig. 4.3.3. The time of the whole iterative process and the accuracy for five smallest eigenvalues **vs.** the inner stopping criteria ε_0 and ε_{coarse}, $Llvs = 3$, Laplace problem

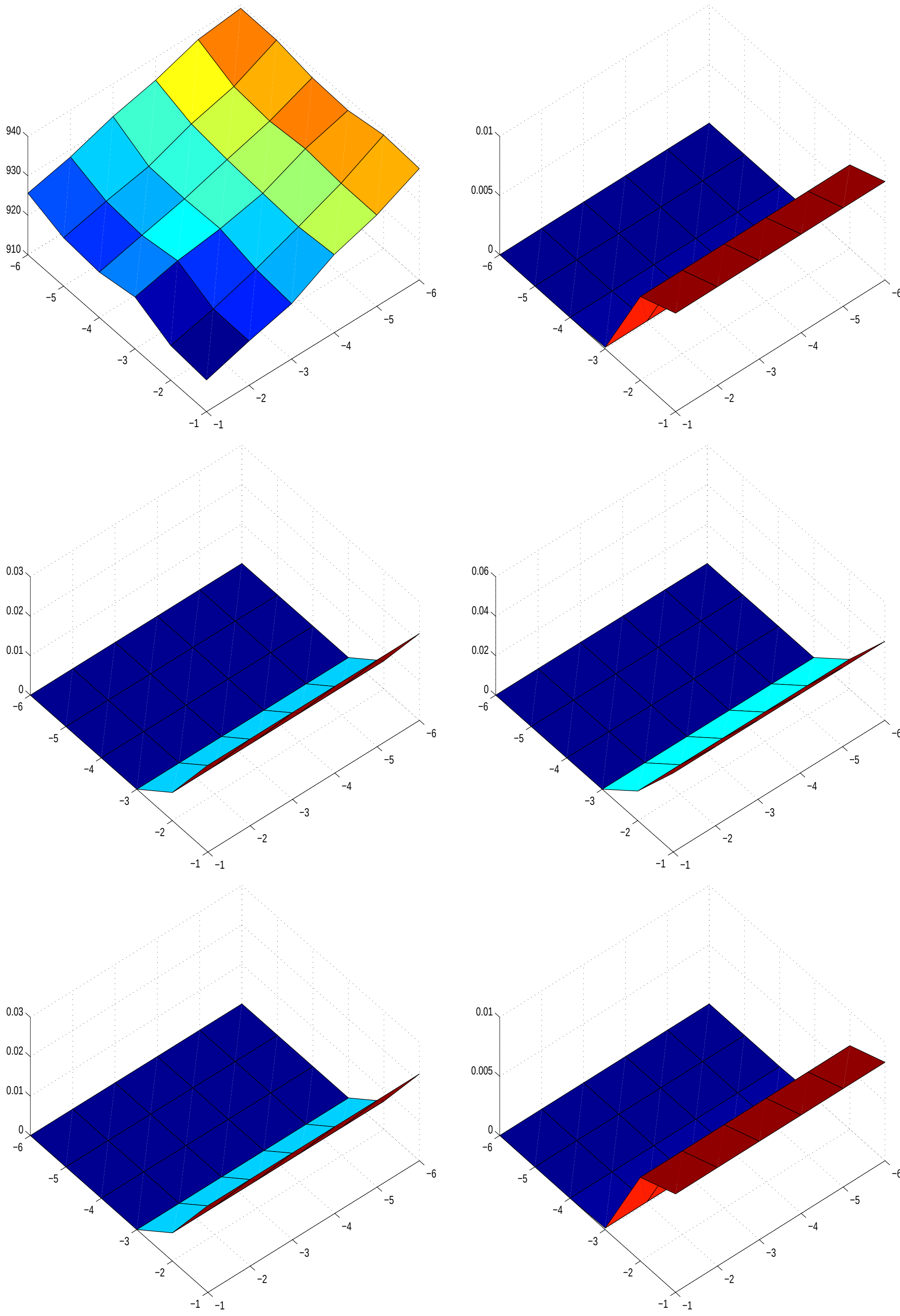

Fig. 4.3.4. The time of the whole iterative process and the accuracy for five smallest eigenvalues **vs.** the inner stopping criteria ε_0 and ε_{coarse}, $Llvs = 2$, Laplace problem

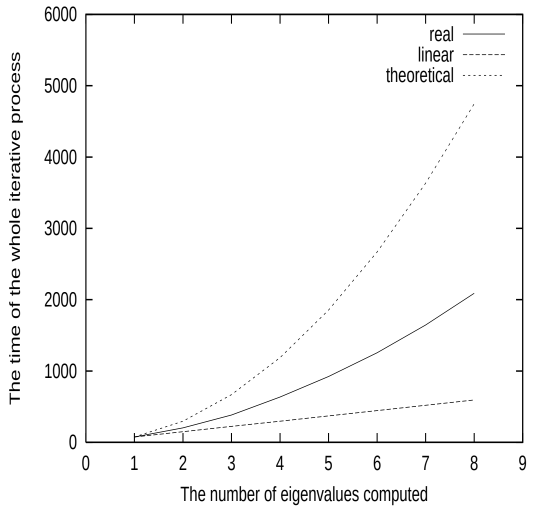

Fig. 4.5. The time of the whole iterative process with respect to the number of eigenvectors computed, Laplace problem