East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.210918.090519

Vol. **11**, No. 1, pp. 1-19
February 2021

An Algebraic Multigrid Method for Eigenvalue Problems and Its Numerical Tests

Ning Zhang[1], Xiaole Han[2], Yunhui He[3], Hehu Xie[4,5] and Chun'guang You[6,*]

[1]*Institute of Electrical Engineering, Chinese Academy of Sciences, No.6, Beiertiao, Zhongguancun, Haidian, Beijing 100190, China.*
[2]*Institute of Applied Physics and Computational Mathematics, Beijing 100094, China.*
[3]*Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NL A1C 5S7, Canada.*
[4]*LSEC, ICMSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.*
[5]*School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China.*
[6]*CAEP Software Center for High Performance Numerical Simulation, Beijing 100088, China.*

Received 21 September 2018; Accepted (in revised version) 9 May 2019.

Abstract. In order to solve eigenvalue problems, an algebraic multigrid method based on a multilevel correction scheme and the algebraic multigrid method for linear equations is developed. The algebraic multigrid method setup procedure is used for construction of an hierarchy and intergrid transfer operators. In this approach, large scale eigenvalue problems are solved by algebraic multigrid smoothing steps in the hierarchy and by low-dimensional eigenvalue problems. The efficacy and flexibility of the method is demonstrated by a number of test examples and the global convergence, which does not depend on the number of eigenvalues wanted, is obtained.

AMS subject classifications: 65N30, 65N25, 65L15, 65B99

Key words: Algebraic multigrid, multilevel correction, eigenvalue problem.

1. Introduction

Algebraic multigrid (AMG) method was introduced by Brandt *et al.* [2] while investigating multigrid algorithms for automatic algorithm design. However, its convergence has

*Corresponding author. *Email addresses:* zhangning@mail.iee.ac.cn (N. Zhang), youchg@lsec.cc.ac.cn (C. You), hanxiaole@lsec.cc.ac.cn (X. Han), yunhui.he@mun.ca (Y. He), hhxie@lsec.cc.ac.cn (H. Xie)

been proved only for symmetric positive definite M-matrices with weak diagonal dominance [27] and in few other cases not involving M-matrices [16, 24, 36]. One of essential problems in AMG methods is closely connected to the choice of coarse grid and intergrid transfer operators. The problem attracted considerable attention and in addition to the classical coarsening strategy proposed by Ruge and Stüben [27], other approaches such as aggregation and smooth aggregation methods [25, 31], compatible relaxation [5, 22], and interpolation [6] and energy-based strategies [4] have been exploited. Cleary *et al.* [10] carried out numerical experiments to study the robustness and scalability of the AMG method. Parallel and adaptive AMG methods have also been studied in [7, 12]. The simplicity of the AMG method leads to its application to various problems — cf. Refs. [1, 11, 23].

In this work, we deal with the computation of q eigenpairs (maybe not of the smallest magnitude) for the following generalised eigenvalue problem: Find $(\lambda^{(j)}, u^{(j)}) \in \mathbb{R}\times\mathbb{R}^N, j = 1, 2, \ldots, q$ such that $(u^{(j)})^T M u^{(k)} = \delta_{jk}, j, k = 1, 2, \ldots, q$ and

$$Au^{(j)} = \lambda^{(j)} M u^{(j)}, \quad j = 1, 2, \ldots, q, \tag{1.1}$$

where A is a real symmetric positive definite $N \times N$ matrix and M a real symmetric semi-positive $N \times N$ matrix. Note that generalised eigenvalue problems (1.1) arise in the discretisation of the elliptic partial differential equations of electromagnetics, quantum chemistry, material, acoustic data science, and so on. These applications usually require high resolution results and, consequently, suitable discretisations of large scale algebraic eigenvalue problems. Therefore, the construction of efficient eigensolvers with a nearly optimal computational complexity is very important.

It is natural to use AMG and MG methods in eigenvalue problems [3,8,13,14,28,35,37]. A good survey of various application of the AMG methods in eigenvalue problems is presented in [17]. In these methods, an AMG strategy is adopted as the only solver in inner iterations combined with special outer iterations, such as inverse power, shift-and-inverse, Rayleigh-quotient, and locally optimal block preconditioned conjugate gradients. But the application of the AMG method does not lead to a new eigensolver (outer iteration). Recently, a new multilevel correction method has been proposed to solve eigenvalue problems [18–21, 26, 32–34]. The method is based on a new understanding of Aubin-Nitsche technique in the finite element method [19]. In contrast to the methods considered in [17], where AMG is used only as a preconditioner of the stiffness matrix, the coarse space of the multigrid method is employed to improve the working subspace in the eigenvalue problem solving [15]. Therefore, in this multilevel correction scheme, the solution of eigenvalue problem on the finest level mesh can be reduced to solving a sequence of standard boundary value problems on multilevel meshes and eigenvalue problems on a low-dimensional space. Hence, the computational work and the memory required can be at an optimal level. The above discussion shows that the application of a multigrid method to a multilevel correction scheme can provide a new eigensolver.

Motivated by the AMG method for boundary value problems and the multilevel correction method, we develop a new AMG method for eigenvalue problems. It can compute various eigenpairs (which may be not of the smallest magnitude) and allows a free choice of the eigensolvers for the low dimensional eigenvalue problems considered. With simple

Gauss-Seidel relaxation as a smoothing process, the AMG still achieves robust convergence. The method also allows the addition of simple strategies, so that the problems discretised on completely unstructured grids can be efficiently solved. With no geometric background, this AMG method has a wide range of applications. Another aim of this paper is to investigate the efficiency of the AMG method for eigenvalue problems. We test various eigenvalue problems and numerical results show that the time consumption and iteration numbers are almost optimal. Besides, the convergence does not depend on the number of the eigenvalues computed.

The rest of this paper is organised as follows. In Section 2, we recall the classical AMG method, which is mainly used for the construction of a coarse-grid. An AMG algorithm for the eigenvalue problem is proposed and analysed in Section 3. The results of numerical tests presented in Section 4 show the efficiency of the algorithm. Some concluding remarks are given in the last section.

2. Algebraic Multigrid Hierarchy

Let us introduce the classical AMG method for solving ill-conditioned linear systems $Au = f$, which is similar to the geometric multigrid (GMG) method.

2.1. Standard coarsening and interpolation

Since there is no real geometric background, the main content is to determine a coarse-grid directly from the matrix $A = (a_{ij})$. By analogy, we define grid points Ω as the indices $\{1, 2, \ldots, N\}$ of $u = (u_1, u_2, \ldots, u_N)^T$, and choose a subset of Ω as the coarse grid points according to the undirected adjacency graph of the matrix A.

In order to derive a coarse level system, we split Ω into two disjoint subsets $\Omega = C \cup F$, where C contains the coarse grid points and F is the complement of C. Following [27, 29, 30], for a fixed $0 < \theta < 1$ (usually 0.25) we define the strong dependent set

$$S_i := \left\{ j \Big| -a_{ij} \geq \theta \max_{\substack{a_{i\ell}<0 \\ \ell \neq i}} |a_{i\ell}| \right\},$$

and the strong influence set $S_i^T := \{j | i \in S_j\}$. By $|S_i^T|$ we denote the number of elements in S_i^T. This characteristic shows how valuable as a coarse grid point the variable i is.

After the construction of the coarse variable set C and its complement F, we define interpolation from the coarse level with mesh size H to fine level with mesh size h. For any vector e^H in the coarse grid the interpolation (prolongation) operator to fine grid can be defined as follows:

$$\left(I_H^h e^H\right)_i = \begin{cases} e_i^H, & \text{if} \quad i \in C, \\ \sum\limits_{j \in P_i} \omega_{ij} e_j^H, & \text{if} \quad i \in F, \end{cases}$$

where P_i is a small set of interpolation points of $C \cap N_i$ and $N_i := \{j \in \Omega : j \neq i, a_{ij} \neq 0\}$ refers to the neighborhood of the point $i, i \in \Omega$.

In the simplest case $P_i = S_i \cap C$, the direct interpolation can be applied immediately [30]. More exactly, the interpolation weights are

$$\omega_{ij} = \begin{cases} -\alpha_i \dfrac{a_{ij}}{a_{ii}}, & \text{if} \quad a_{ij} < 0, \\ -\beta_i \dfrac{a_{ij}}{a_{ii}}, & \text{if} \quad a_{ij} > 0 \end{cases}$$

with

$$\alpha_i = \frac{\sum\limits_{j \in N_i, a_{ij} < 0} a_{ij}}{\sum\limits_{\ell \in P_i, a_{i\ell} < 0} a_{i\ell}}, \quad \beta_i = \frac{\sum\limits_{j \in N_i, a_{ij} > 0} a_{ij}}{\sum\limits_{\ell \in P_i, a_{i\ell} > 0} a_{i\ell}}.$$

The leading coefficients α_i and β_i are chosen so that for zero row sum matrices the method interpolates constants exactly. For more details and other interpolations, such as standard interpolation, the reader can consult Refs. [27, 29, 30].

Remark 2.1. For different types of matrices, different type of coarsening and interpolation strategies can be adopted.

2.2. Coarse problem

The AMG setup procedure leads to a hierarchy of vector spaces indexed by the set $k = 1, 2, \ldots, n$, where $k = 1$ and $k = n$ are, respectively, the finest and the coarsest levels. Denote the finest grid $\Omega_1 = \Omega$ and corresponding matrices $A_1 = A, M_1 = M$. Based on A_1, the AMG scheme builds up the prolongation and restriction operators I_{k+1}^k and $I_k^{k+1} := (I_{k+1}^k)^T, k = 1, 2, \ldots, n-1$, respectively. The coarse matrices are defined via the Galerkin projection as

$$A_{k+1} := I_k^{k+1} A_k I_{k+1}^k, \quad M_{k+1} := I_k^{k+1} M_k I_{k+1}^k \quad \text{for} \quad k = 1, 2, \ldots, n-1.$$

In what follows, $d_1, d_2, \ldots, d_n$ are the dimensions of the problems defined on the grids $\Omega_1, \Omega_2, \ldots, \Omega_n$, respectively.

3. AMG Algorithm for Eigenvalue Problems

In this section, we develop an AMG method for eigenvalue problems. Following the geometric case [33], we assume that $\{\lambda_k^{(j,\ell)}, u_k^{(j,\ell)}\}_{j=1}^q$ are approximations of the eigenpairs wanted. In order to improve the accuracy of such pairs, we design the AMG correction step presented in Algorithm 3.1.

According to the construction of (3.1), the result $\widetilde{u}_k^{(j,\ell+1)}$ can be viewed as performing a few inexact inverse power iterations on the given approximation $u_k^{(j,\ell)}$. Let V_n be the coarsest AMG space. The matrices $A_{n,k}^{(\ell+1)}$ and $M_{n,k}^{(\ell+1)}$ are exactly the Galerkin projections on the augmented subspace $V_n + \text{span}\{V_{k,\ell+1}\}$. Therefore, Algorithm 3.1 is always more accurate than the (block) inverse power method, which uses projection subspace $\text{span}\{V_{k,\ell+1}\}$.

Algorithm 3.1 AMG Correction Step.

1: **for** $j = 1, \ldots, q$ **do**

2: Use the AMG iterations to solve the linear equation

$$A_k \widehat{u}_k^{(j,\ell+1)} = \lambda_k^{(j,\ell)} M_k u_k^{(j,\ell)}. \tag{3.1}$$

Perform m AMG iteration steps with the initial value $u_k^{(j,\ell)}$ to obtain a new eigenfunction approximation

$$\widetilde{u}_k^{(j,\ell+1)} := \mathrm{AMG}\Big(k, \lambda_k^{(j,\ell)} u_k^{(j,\ell)}, u_k^{(j,\ell)}, m\Big),$$

where k denotes the working level Ω_k for the AMG iteration, $\lambda_k^{(j,\ell)} u_k^{(j,\ell)}$ leads to the right hand side term of the linear equation, $u_k^{(j,\ell)}$ is the initial guess and m the number of AMG cycles.

3: **end for**

4: Set

$$V_{k,\ell+1} := \Big[\widetilde{u}_k^{(1,\ell+1)}, \ldots, \widetilde{u}_k^{(q,\ell+1)}\Big]$$

and construct matrices $A_{n,k}^{(\ell+1)}$ and $M_{n,k}^{(\ell+1)}$ as follows

$$A_{n,k}^{(\ell+1)} = \begin{pmatrix} A_n & I_k^n A_k V_{k,\ell+1} \\ V_{k,\ell+1}^T A_k I_n^k & V_{k,\ell+1}^T A_k V_{k,\ell+1} \end{pmatrix}, \quad M_{n,k}^{(\ell+1)} = \begin{pmatrix} M_n & I_k^n M_k V_{k,\ell+1} \\ V_{k,\ell+1}^T M_k I_n^k & V_{k,\ell+1}^T M_k V_{k,\ell+1} \end{pmatrix}.$$

5: Solve the following eigenvalue problems: Find $\{\lambda_k^{(j,\ell+1)}, x_k^{(j,\ell+1)}\}_{j=1}^q$ such that

$$\begin{aligned} &\Big(x_k^{(j,\ell+1)}\Big)^T M_{n,k}^{(\ell+1)} x_k^{(j,\ell+1)} = 1, \\ &A_{n,k}^{(\ell+1)} x_k^{(j,\ell+1)} = \lambda_k^{(j,\ell+1)} M_{n,k}^{(\ell+1)} x_k^{(j,\ell+1)}, \quad j = 1, \ldots, q. \end{aligned}$$

6: Select the wanted eigenpairs $\{\lambda_k^{(j,\ell+1)}, x_k^{(j,\ell+1)}\}_{j=1}^q$ and do the following computation:

7: **for** $j = 1, \ldots, q$ **do**

8:

$$u_k^{(j,\ell+1)} = I_n^k x_k^{(j,\ell+1)}(1 : d_n) + V_{k,\ell+1} x_k^{(j,\ell+1)}(d_n + 1 : d_n + q).$$

9: **end for**

10: Summarise the above steps by defining

$$\Big\{\lambda_k^{(j,\ell+1)}, u_k^{(j,\ell+1)}\Big\}_{j=1}^q := \mathrm{AMGCorrection}\Big(n, k, \Big\{\lambda_k^{(j,\ell)}, u_k^{(j,\ell)}\Big\}_{j=1}^q\Big).$$

Moreover, the approximate low frequency information in V_n leads to a better convergence rate of Algorithm 3.1 than the usual (block) inverse power method — cf. [19–21, 33, 34].

Note that the efficacy and convergence of Algorithm 3.1 is studied in Section 4.

Remark 3.1. In order to obtain better results for (3.1), we can combine other efficient iterative strategies — e.g. shifting and polynomial filtering. More general correction subspaces — e.g. V_n + KrylovSpaces, can also be considered.

Using the previous considerations, we can construct an AMG method, which combines nested technique and the AMG correction step of Algorithm 3.1.

Algorithm 3.2 AMG Eigenvalue Solver.

1: For the n_1-th grid $\Omega_{n_1}, n_1 \leq n$, introduce the following low-dimensional eigenvalue problem: Find $\{\lambda_{n_1}^{(j)}, u_{n_1}^{(j)}\}_{j=1}^{q}$ such that

$$\begin{aligned} &\left(u_{n_1}^{(j)}\right)^T M_{n_1} u_{n_1}^{(j)} = 1, \\ &A_{n_1} u_{n_1}^{(j)} = \lambda_{n_1}^{(j)} M_{n_1} u_{n_1}^{(j)}. \end{aligned} \tag{3.2}$$

Solve the problems (3.2) to derive the eigenpairs $\{\lambda_{n_1}^{(j)}, u_{n_1}^{(j)}\}_{j=1}^{q}$ approximating the wanted eigenpairs.

2: For $k = n_1 - 1, \ldots, 1$, perform the following correction steps:

(a) Set $\lambda_k^{(j,0)} = \lambda_{k+1}^{(j)}$ and $u_k^{(j,0)} = I_{k+1}^k u_{k+1}^{(j)}$ for $j = 1, \ldots, q$.

(b) For $\ell = 0, \ldots, p_k - 1$ do the correction iteration

$$\left\{\lambda_k^{(j,\ell+1)}, u_k^{(j,\ell+1)}\right\}_{j=1}^{q} = \text{AMGCorrection}\left(n, k, \left\{\lambda_k^{(j,\ell)}, u_k^{(j,\ell)}\right\}_{j=1}^{q}\right).$$

(c) Set $\lambda_k^{(j)} = \lambda_k^{(j,p_k)}$ and $u_k^{(j)} = u_k^{(j,p_k)}$ for $j = 1, \ldots, q$.

As the result, we obtain eigenpair approximations $\{\lambda_1^{(j,\ell+1)}, u_1^{(j,\ell+1)}\}_{j=1}^{q}$ on the finest level grid Ω_1.

In contrast to the GMG method [33], we do not have exact prolongation and restriction operators. In practical computations, one chooses suitable iterations p_k in order to satisfy the accuracy requirements. In comparison to other AMG methods, the above method only requires the employment of smoothing iterations for standard elliptic linear equations and the AMG method can act as a black-box. The small scale eigenvalue problems used can be solved by any eigensolver, which can also act as a black-box in our method. Furthermore, unlike the classical eigensolvers such as Lanczos and Arnoldi, the memory required for the eigenpair solving is only about qN.

According to the analysis for GMG method [33, 34], the AMG method can have a good convergence rate if the coarse grid captures low-frequency information of the finest grid well.

4. Numerical Results

Let us illustrate the efficacy of the algorithm by six examples for algebraic eigenvalue problems arising in the linear finite element method for the Poisson eigenvalue problem on different domains. The AMG method is tested on structured and unstructured meshes. We use globally uniform convergence to show that the numerical method has the same convergence rate for various number of computed eigenpairs.

4.1. Default implementation settings

Some parameters have to be set to finish the AMG setup phase and the eigenvalue algorithm. Recall that we want to compute the first q eigenpairs of generalised eigenvalue problems. The main default settings and procedures are as follows.

- Strong dependent/influence set threshold: $\theta = 0.25$.
- Interpolation type: direct interpolation.
- Linear solver: 1 AMG V-cycle:
  - Pre-smoothing: 1 Gauss-Seidel iteration.
  - Post-smoothing: 1 Gauss-Seidel iteration.
- The initial eigensolver level: $n_1 = n$.
- Correction number at each level: $p_{n_1-1} = p_{n_1-2} = \cdots = p_2 = 1$, $p_1 \leq 20$.
- Direct eigensolver: Arnoldi process by the ARPACK library. The j-th eigenvalue on the finest level is denoted by λ_j^{dir}.
- q is the number of desired eigenvalues, ranging from 1 to 30.
- Total error at the finest level between the eigenvalue approximations by the AMG method and direct solver: $e_\ell = \sum_{j=1}^{q} |\lambda_j^\ell - \lambda_j^{\mathrm{dir}}|$ where $\lambda_j^\ell = \lambda_1^{(j,\ell)}$, $\ell = 1, 2, \ldots, p_1$.
- Total error tolerance: $\tau \leq 10^{-9}$.
- Average convergence ratio: ratio $= (e_{p_1}/e_1)^{1/(p_1-1)}$.
- The stiffness matrix A and mass matrices M are obtained by discretising the following problems respectively with the linear finite element method [9].
- Test machine: Intel Xeon E5-2620 2.00GHz, two 6-core dual thread CPUs, 72G memory.

Example 4.1 (Model eigenvalue problem)**.** The first problem is the most elementary eigenvalue problem — viz. find (λ, u) such that

$$\begin{aligned} -\Delta u &= \lambda u \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \\ \int_\Omega u^2 d\Omega &= 1, \end{aligned} \tag{4.1}$$

where $\Omega = (0,1) \times (0,1)$. The meshes are generated by uniform refinement starting with the mesh of size $h = 1$. At each level, the dimension of the corresponding grid respectively is 4198401, 2095105, 525309, 131581, 33021, 8569, 2109, 542 and there are 8 levels in total.

Table 1 shows the average convergence ratio, total errors, and iteration numbers for different desired amount of eigenvalues. Unlike the Krylov methods, where more than q eigenvalues are actually computed, Algorithm 3.2 has to determine the first q eigenvalues only. We also note that if λ_q and λ_{q+1} are close, then it is more difficult for the algorithm to find these first q eigenvalues. However, as follows from Table 1, the overall behaviour is similar — viz. for different number of computed eigenpairs, we obtain a globally uniform convergence ratio close to 0.11. In order to demonstrate the optimal computational complexity, Fig. 1 provides algebraic errors and CPU time (in seconds) for the first 13 eigenvalues. The left graph in Fig. 1 shows that the error of the eigenvalues linearly decreases with the number of iterations. The right graph in Fig. 1 shows perfect linear growth of CPU time and is faster than the direct method which needs 1737.56 seconds for the same error tolerance.

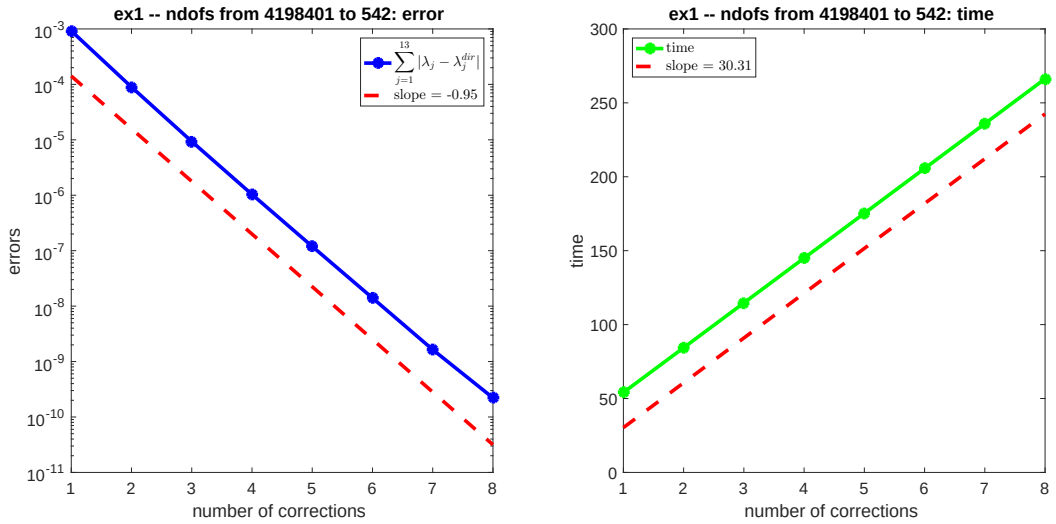

Figure 1: Example 4.1. AMG method for first 13 eigenvalues on uniform refinement mesh. Algebraic errors and CPU time.

Table 1: Example 4.1. Algebraic errors for unit square and uniform refinement mesh.

q	λ_q	λ_q/λ_{q+1}	ratio	iter	total error
1	19.739351152446577	0.399999830658111	0.110359	6	0.19e-09
2	49.348398772994052				
3	49.348426661655587	0.624999514802597	0.109683	7	0.12e-09
4	78.957543954641395	0.800000031375434	0.110713	7	0.21e-09
5	98.696926072478092				
6	98.696926072787591	0.769230457062699	0.111902	7	0.45e-09
7	128.306055963593963				
8	128.306290898228610	0.764706565466666	0.110547	7	0.69e-09
9	167.784999753374791				
10	167.785014924843836	0.944442983816993	0.112142	8	0.12e-09
11	177.654996436879685	0.900000105867284	0.115364	8	0.18e-09
12	197.394417265854258				
13	197.394417273111628	0.799999907230014	0.113346	8	0.22e-09
14	246.743050204326266				
15	246.743972243036808	0.961541879013450	0.112188	8	0.27e-09
16	256.612819086151660				
17	256.612819086431614	0.896550451610119	0.113451	8	0.37e-09
18	286.222396771513957				
19	286.222479299660563	0.906247979981473	0.112243	8	0.42e-09
20	315.832405282174534	0.941176611569284	0.111708	8	0.45e-09
21	335.571880346205262				
22	335.571880413890710	0.918920429907456	0.115726	8	0.69e-09
23	365.180563509384399				
24	365.180577155566198	0.924998520135014	0.110632	8	0.59e-09
25	394.790444748240247				
26	394.790444752357189	0.975610272573643	0.114287	8	0.87e-09
27	404.659991649028711				
28	404.662532642826989	0.911114427716026	0.113430	8	0.95e-09
29	444.140187371669356				
30	444.140438372226185	0.900001567378938	0.138346	9	0.60e-09

Example 4.2 (Poisson eigenvalue problem on L-shaped domain)**.** In this example, we test the performance of Algorithm 3.2 for the problem (4.1), but on an L-shaped domain $\Omega = [-1,1] \times [-1,1] \backslash (0,1) \times (-1,0)$. The meshes are generated by uniform refinement starting with the mesh of size $h = 1$. At each level, the dimension of the corresponding

grid respectively is 3149825, 1570817, 394236, 98812, 24828, 6518, 1596 and there are 7 levels in total.

Table 2 shows numerical results for different wanted eigenvalues determined by our AMG algorithm. We again see globally uniform convergence, but it is worth noting that at the coarsest level the dimension is increased to 1596. It is reasonable, since an L-shaped

Table 2: Example 4.2. Algebraic errors for L-shaped domain and uniform refinement mesh.

q	λ_q	λ_q/λ_{q+1}	ratio	iter	total error
1	9.639941155614530	0.634320854275592	0.105045	6	0.12e-09
2	15.197263483672709	0.769901102357829	0.104864	6	0.28e-09
3	19.739241101397241	0.668638429761318	0.105088	6	0.48e-09
4	29.521547405588866	0.925055808457296	0.104818	6	0.73e-09
5	31.913260946733118	0.769457056102703	0.107524	7	0.13e-09
6	41.475038397040180	0.922721052755031	0.107113	7	0.18e-09
7	44.948620466830491	0.910845766545385	0.106877	7	0.24e-09
8	49.348223505840721				
9	49.348260690989306	0.870179621581926	0.106558	7	0.35e-09
10	56.710430199775971	0.867439196741824	0.106295	7	0.41e-09
11	65.376836108842241	0.920039372342096	0.105972	7	0.48e-09
12	71.058737347746145	0.992811733747242	0.105879	7	0.54e-09
13	71.573224743772840	0.906477598649387	0.107138	7	0.66e-09
14	78.957521785881866	0.884144399022638	0.106518	7	0.73e-09
15	89.303876010710553	0.967460206330204	0.106860	7	0.87e-09
16	92.307544461658438	0.947893993499531	0.106015	7	0.95e-09
17	97.381716831929808	0.986674656579670	0.189571	10	0.28e-09
18	98.696886742288171				
19	98.696886759985802	0.971356701426527	0.108106	8	0.15e-09
20	101.607253663911848	0.904219670345751	0.107446	8	0.16e-09
21	112.370098767105745	0.972722811055895	0.110938	8	0.22e-09
22	115.521192152497733	0.900353570057789	0.112182	8	0.26e-09
23	128.306474249980454				
24	128.306787514516458	0.986060874784479	0.111082	8	0.29e-09
25	130.120554212801750	0.998942245349293	0.108884	8	0.27e-09
26	130.258335573047447	0.914382448573862	0.109380	8	0.29e-09
27	142.454982350337076	0.942648379890289	0.109414	8	0.32e-09
28	151.122078379763281	0.978332935190765	0.109511	8	0.34e-09
29	154.468967509814064	0.952336564402184	0.108736	8	0.35e-09
30	162.199975600831635	0.984711949246173	0.112557	8	0.48e-09

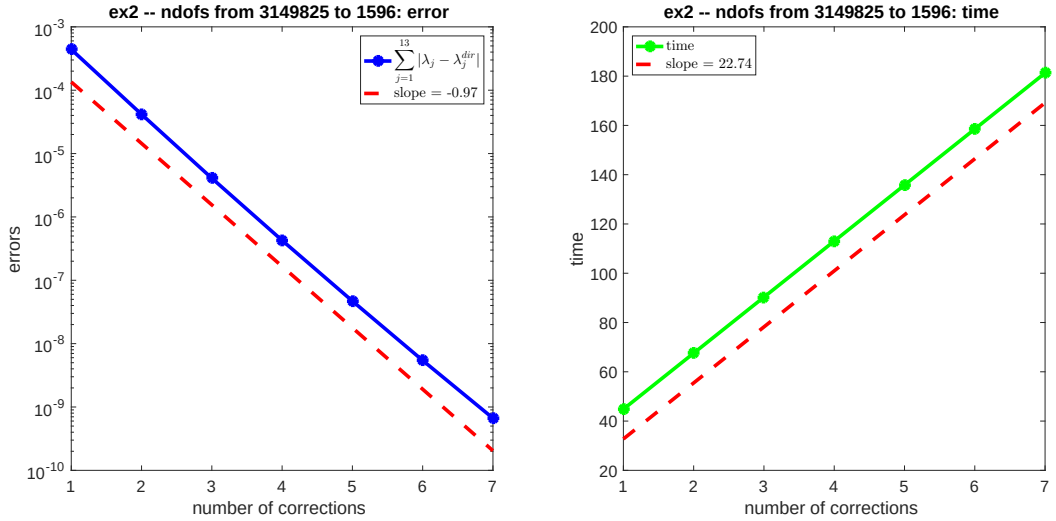

Figure 2: Example 4.2. AMG method for first 13 eigenvalues on uniform refinement mesh. Algebraic errors and CPU time.

domain is considered. Nevertheless, for the first 13 eigenvalues the behaviour of algebraic errors and CPU time is similar to Example 4.1. Again, the timings are close to scaling linearly with the number of iterations — cf. Fig. 2. Note that the direct method needs 936,94 seconds for finding the first 13 eigenpairs for the same error tolerance.

Example 4.3 (Poisson eigenvalue problem with discontinuous parameters I)**.** On the unit square $\Omega = (-1,1)\times(-1,1)$, we consider the following Poisson eigenvalue problem: Find (λ, u) such that

$$\begin{aligned} -\nabla\cdot(K\nabla u) &= \lambda u \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \\ \int_\Omega u^2 d\Omega &= 1, \end{aligned} \tag{4.2}$$

where the coefficient matrix K has discontinuous elements. In order to test our multigrid algorithm, in this and the next example we choose two different coefficient matrices K. In particular, here the matrix K is

$$\begin{aligned} K &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} && \text{in } (-1,0)\times(0,1)\cup(0,1)\times(-1,0), \\ K &= \begin{pmatrix} 1000 & 0 \\ 0 & 1000 \end{pmatrix} && \text{in } [0,1]\times[0,1], \\ K &= \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} && \text{in } [-1,0]\times[-1,0]. \end{aligned}$$

The meshes are generated by uniform refinement starting with the mesh of size $h = 1$. At each level, the dimension of the corresponding grid respectively is 4198401, 2095105, 525310, 131585, 33024, 8564, 2137 and there are 7 levels in total.

Table 3 shows numerical results for various wanted eigenvalues. In order to show globally uniform convergence ratio, the dimension on the coarsest level is increased to 2137. We also plot total errors and CPU time for the first 13 eigenvalues as the functions of the number of corrections in Fig. 3. It is clear that the decreasing of the total error is robust with respect to the number of corrections. Moreover, CPU time still grows linearly with along with the number of corrections.

Table 3: Example 4.3. Algebraic errors for unit square and uniform refinement mesh.

q	λ_q	λ_q/λ_{q+1}	ratio	iter	total error
1	0.019726703793271	0.399974375039268	0.097224	3	0.10e-09
2	0.049319919035649				
3	0.049320060970441	0.624839632041262	0.096406	3	0.66e-09
4	0.078932350704643	0.800221419880202	0.098629	4	0.11e-09
5	0.098638137824978				
6	0.098638194206567	0.768992275837269	0.097949	4	0.23e-09
7	0.128269421300976				
8	0.128270399246696	0.764957776222445	0.097399	4	0.38e-09
9	0.167682979680431				
10	0.167683133739996	0.944062075515168	0.096656	4	0.57e-09
11	0.177618758436507	0.900067811260723	0.096447	4	0.68e-09
12	0.197339307343651				
13	0.197339360277547	0.799924741492187	0.096014	4	0.91e-09
14	0.246697407945438				
15	0.246701139291675	0.961965508840176	0.098786	5	0.13e-09
16	0.256455285584114				
17	0.256455427715290	0.896251379863420	0.098347	5	0.16e-09
18	0.286142296098190				
19	0.286142698738707	0.906120868759646	0.097962	5	0.20e-09
20	0.315788664188252	0.941206046642901	0.097811	5	0.22e-09
21	0.335514912292169				
22	0.335514964638074	0.919331113066801	0.097356	5	0.26e-09
23	0.364955520235607				
24	0.364955778306734	0.924686470880238	0.096885	5	0.31e-09
25	0.394680564493737				
26	0.394680661391869	0.975464015890347	0.104709	5	0.50e-09
27	0.404608119789665				
28	0.404618330771774	0.911165080572709	0.096076	5	0.41e-09
29	0.444066985663512				
30	0.444068055608404	0.900409760718495	0.095797	5	0.47e-09

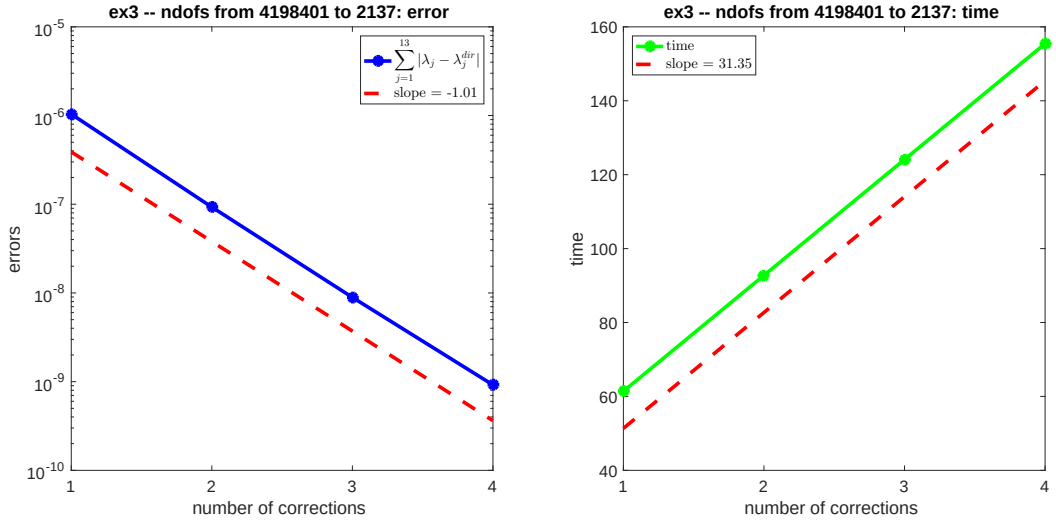

Figure 3: Example 4.3. AMG method for first 13 eigenvalues on uniform refinement mesh. Algebraic errors and CPU time.

Example 4.4 (Poisson eigenvalue problem with discontinuous parameters II)**.** We now consider the eigenvalue problem (4.2) with the coefficient matrix

$$K = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{in } (-1,0)\times(0,1)\cup(0,1)\times(-1,0),$$
$$K = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \quad \text{in } [0,1]\times[0,1]\cup[-1,0]\times[-1,0].$$

The mesh is generated by uniform refinement starting with the mesh of size $h = 1$.

Unlike the previous examples, we do not observe similar globally uniform convergence behaviour even if the coarsest level dimension is increased to 8515. In practical computing, we could take at least two simple strategies to improve the performance — viz. to increase the coarsest level dimension or to compute extra eigenpairs. Table 4 shows the results of numerical experiments for the first 14 eigenvalues with two improvement strategies. Note that the gap between 14-teen 15-teen eigenvalues is $\lambda_{14}/\lambda_{15} = 0.873102340090717$. Two AMG hierarchies are generated, where for each strategy the dimensions at every level respectively are 4198401, 2095105, 525311, 131589, 33028, 8515, 2165, 597, 176 and 4198401, 2095105, 525311, 131589, 33028, 8515, 2165, 597. Three extra eigenpairs — viz. the first 17 eigenvalues and the corresponding eigenvectors are computed for the comparison. The algebraic errors and CPU time for the first 14 eigenvalues in third situation, which takes the shortest time, are given in Fig. 4. Unlike Example 4.3, for discontinuous parameters the number of iterations required to achieve the same tolerance increases greatly and so does CPU time.

Remark 4.1. If the GMG method in [19, 33, 34] is applied to the problem (4.2) with the above coefficient matrix K defined in this example and the hierarchy levels are 1050625,

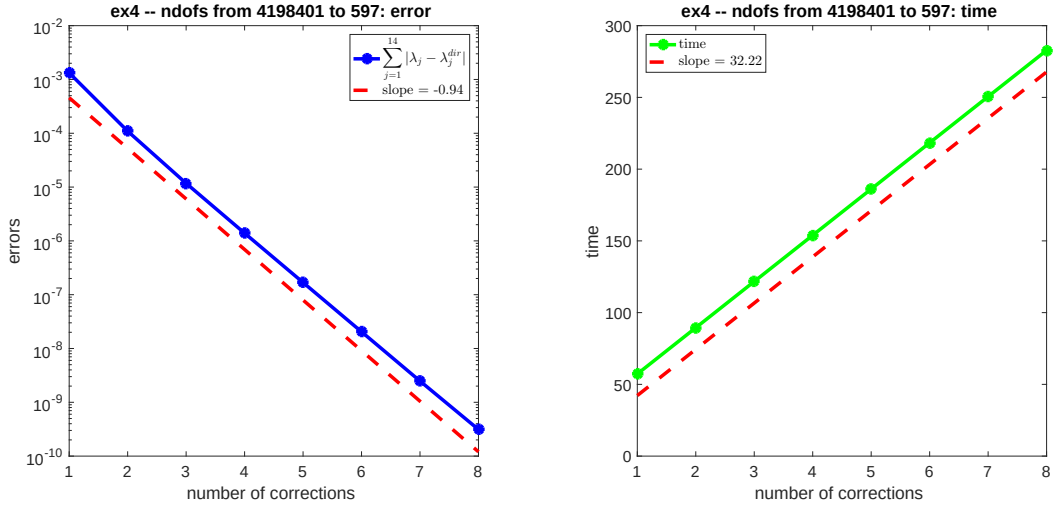

Figure 4: Example 4.4. AMG method for first 14 eigenvalues on uniform refinement mesh. Algebraic errors and CPU time.

263169, 66049, 16641, 4225, 1089, 289, we were not able to establish the convergence — cf. Table 5. The reason should come from the fact that the algebraic multigrid hierarchy has the adaptive coarsening property, according to the coefficient functions.

In order to show the generality of the AMG method, in the following examples the Delaunay scheme is applied to generate meshes with no hierarchical structures. Similar to Example 4.4, we test the method with two improvement strategies mentioned.

Table 4: Example 4.4. Algebraic errors for unit square and uniform refinement mesh.

q	actual q	d_n	ratio	iter	total error	time
14	14	176	0.252941	12	0.39e-09	406.1
14	17	176	0.132108	8	0.96e-09	336.9
14	14	597	0.113013	8	0.31e-09	282.7
14	17	597	0.113353	8	0.27e-09	359.9

Table 5: Example 4.4. Number of correction and corresponding algebraic errors for unit square and uniform refinement mesh for first 6 eigenvalues.

correction	$\sum_{j=1}^{6} \lvert\lambda_j - \lambda_j^{dir}\rvert$	ratio
1	2.526094e+01	-
2	2.565658e+01	1.015662
3	2.421893e+01	0.943965
4	2.499157e+01	1.031902
5	2.417530e+01	0.967338
6	2.556458e+01	1.057467

Example 4.5 (Example 4.1 with Delaunay mesh)**.** We now consider the model eigenvalue problem (4.1) on the unit square. Table 6 shows numerical results for computing the first 13 eigenvalues. Note that the gap between the 13-teen and 14-teen eigenvalues is $\lambda_{13}/\lambda_{14} = 0.799999552462451$. The dimensions of two AMG hierarchies at each level respectively are 4623349, 1630814, 614806, 222577, 76384, 27104, 9743, 3459, 1196 and 4623349, 1630814, 614806, 222577, 76384, 27104, 9743, 3459. The first 17 eigenvalues and the corresponding eigenvectors are also computed for comparison. Fig. 5 displays algebraic errors and CPU time for the first 13 eigenvalues with the later situation requiring the shortest time — cf. Table 6.

Table 6: Example 4.5. Results about the algebraic errors on unit square with Delaunay mesh.

q	actual q	d_n	ratio	iter	total error	time
13	13	1196	0.305067	18	0.90e-09	659.3
13	17	1196	0.158769	12	0.81e-09	604.0
13	13	3459	0.225724	15	0.44e-09	650.5
13	17	3459	0.151409	12	0.46e-09	723.2

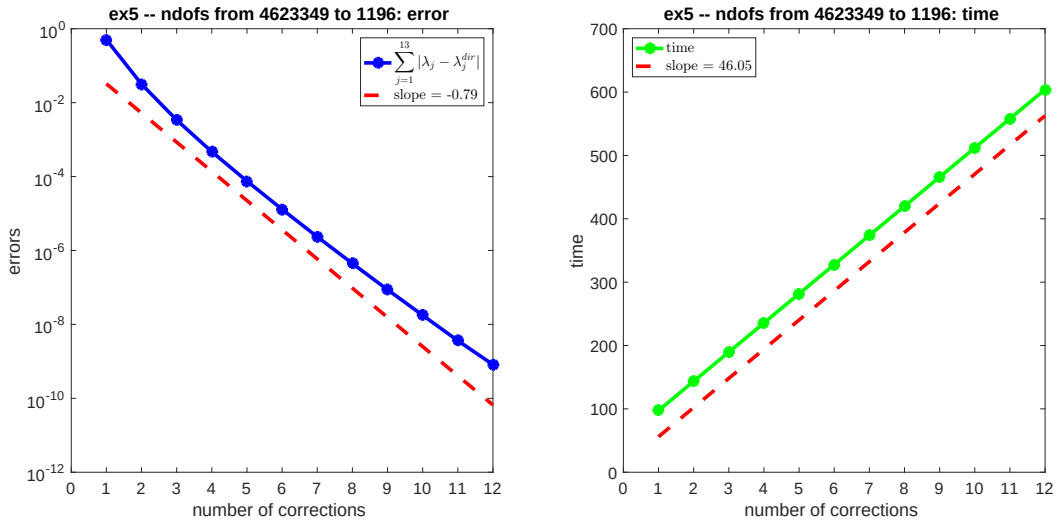

Figure 5: Example 4.5. AMG method for first 13 eigenvalues on the Delaunay mesh. Algebraic errors and CPU time.

Example 4.6 (Example 4.2 with Delaunay mesh)**.** We consider the eigenvalue problem (4.1) on the L-shape domain $\Omega = [-1,1]\times[-1,1]\backslash(0,1)\times(-1,0)$ with the mesh generated by Delaunay method. In Table 7, we present numerical results for computing the first 13 eigenvalues. Note that the gap between 13-teen and 14-teen eigenvalues is $\lambda_{13}/\lambda_{14} = 0.906477598649387$. The dimensions on each level respectively are 3468624, 1219797, 453107, 165008, 55740, 19564, 6958, 2458, 865 and 3468624, 1219797, 453107, 165008,

55740, 19564, 6958, 2458. Similar to the previous example, various extra eigenvalues from the first 14-teen to 17-teen eigenvalues, are computed. According to Table 7, the second situation takes the shortest time. Fig. 6 shows the convergence behaviour and CPU time.

Table 7: Example 4.6. Results about the algebraic errors on an L-shape domain with Delaunay mesh.

q	actual q	d_n	ratio	iter	total error	time
13	13	865	0.540191	20	0.20e-05	525.8
13	14	865	0.308052	18	0.42e-09	517.2
13	15	865	0.247463	15	0.66e-09	476.7
13	16	865	0.219253	14	0.53e-09	481.3
13	17	865	0.214510	14	0.40e-09	510.0
13	13	2458	0.468039	20	0.11e-06	597.9
13	14	2458	0.230357	14	0.99e-09	469.0
13	15	2458	0.182574	13	0.26e-09	469.2
13	16	2458	0.159574	12	0.31e-09	471.8
13	17	2458	0.155989	12	0.24e-09	507.2

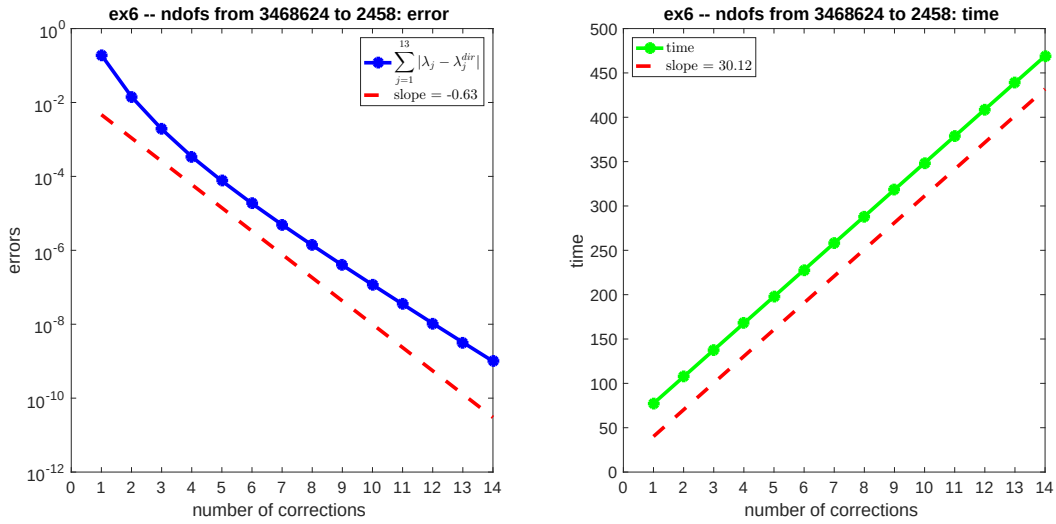

Figure 6: Example 4.6. AMG method for first 13 eigenvalues on the Delaunay mesh. Algebraic errors and CPU time.

5. Conclusions

An AMG method for algebraic eigenvalue problems arising from the discretisation of partial differential equations is developed. Paired with the multilevel correction method and the AMG method for linear equations, the resulting algorithm requires an almost optimal computational work and the least memory. The efficiency of the method is demon-

strated by six numerical examples. The first three examples show that under suitable conditions, the AMG method has globally uniform convergence. Another examples are included to discuss strategies for the convergence improvement if the conditions used are not satisfied. In particular, various interpolation methods (direct and standard), pre- and postsmoothing operators (CG,GS), linear solvers (AMG V-cycle,CG,PCG), and various parameters p_k and d_n are tested.

The development of robust and efficient AMG eigensolvers for eigenvalue problems involves testing of different coarsening and interpolation strategies for various types of matrices and using of other solution schemes. Furthermore, for large scale eigenvalue problems, the parallelisation methods should also be studied.

Acknowledgments

We would like to express our sincerely thanks to Prof. Chensong Zhang for providing the algebraic multigrid software.

This work was supported in part by the National Key Research and Development Program of China (2019YFA0709601), by the Science Challenge Project (No. TZ2016002), by the Beijing Natural Science Foundation (Z200003), by the National Natural Science Foundations of China (NSFC 11771434, 11801021, 91730302, 91630201), and by the National Center for Mathematics and Interdisciplinary Science, CAS.

References

[1] P. Bastian, M. Blatt and R. Scheichl, *Algebraic multigrid for discontinuous Galerkin discretizations of heterogeneous elliptic problems*, Numer. Linear Algebra Appl. **19(2)**, 367–388 (2012).

[2] A. Brandt, S. McCormick and J. Ruge, *Algebraic Multigrid (AMG) for Automatic Algorithm Design and Problem Solution*, Report Comp. Studies, Colorado State University (1982).

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

[4] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick and J. Ruge, *An energy-based AMG coarsening strategy*, Numer. Linear Algebra Appl. **13(2-3)**, 133–148 (2006).

[5] J. Brannick and R. Falgout, *Compatible relaxation and coarsening in algebraic multigrid*, SIAM J. Sci. Comp. **32(3)**, 1393–1416 (2010).

[6] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick and J. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comp. **22(5)**, 1570–1592 (2001).

[7] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick and J. Ruge, *Adaptive algebraic multigrid methods*, SIAM J. Sci. Comp. **4**, 1261–1286 (2006).

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

[9] P. G. Ciarlet, *The Finite Element Method for Elliptic Problem*, North-Holland (1978).

[10] A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, G. Miranda and J. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comp. **21(5)**, 1886–1908 (2000).

[11] H. De Sterck, T. Manteuffel, S. McCormick, K. Miller, J. Ruge and G. Sanders, *Algebraic multigrid for Markov chains*, SIAM J. Sci. Comp. **32(2)**, 544–562 (2010).
[12] R. Falgout, V. Henson, J. Jones and U. Yang, *Boomer AMG: A Parallel Implementation of Algebraic Multigrid*, Techn. Report UCRL-MI-133583, Lawrence Livermore National Laboratory (1999).
[13] B. Gong, J. Han, J. Sun and Z. Zhang, *A shifted-inverse adaptive multigrid method for the elastic eigenvalue problem*, Commun. Comput. Phys. (2019).
[14] W. Hackbusch, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multi-grid method*, SIAM J. Numer. Anal. **16(2)**, 201–215 (1979).
[15] Y. He, H. Xie, M. Yue and C. You, N. Zhang, *Energy error estimates of subspace method and multigrid method for eigenvalue problems*, Arxiv, 1705.03038 (2017).
[16] W. Huang, *Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance*, Appl. Math. Comput. **46(2)**, 145–164 (1991).
[17] A.V. Knyazev and K. Nrymeyr, *Efficient solution of symmetric eigenvalue problems using multigrid preconditioners in the locally optimal block conjugate gradient method*, Electron. Trans. Numer. Anal. **15**, 38–55 (2003).
[18] R. Kužel and P. Vaněk, *Exact interpolation scheme with approximation vector used as a column of the prolongator*, Numer. Linear Algebra Appl. **22(6)**, 950–964 (2015).
[19] Q. Lin and H. Xie, *An observation on Aubin-Nitsche lemma and its applications*, Math. Pract. Theory **41(17)**, 247–258 (2011).
[20] Q. Lin and H. Xie, *A multilevel correction type of adaptive finite element method for Steklov eigenvalue problems*, in: Proceedings of the International Conference: Applications of Mathematics 2012, J. Brandts, J. Chleboun, S. Korotov, K. Segeth, J.Šístek and T. Vejchodský (Eds), pp. 134–143, 2012.
[21] Q. Lin and H. Xie, *A multi-level correction scheme for eigenvalue problems*, Math. Comp. **84**, 71–88 (2015).
[22] O. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl. **11(2-3)**, 205–227 (2004).
[23] I. Livshits, *Algebraic Multigrid Algorithm for The Indefinite Helmholtz Equations*, Delft University of Technology (2011).
[24] S. MacLachlan and L. Olson, *Theoretical bounds for algebraic multigrid performance: Review and analysis*, Numer. Linear Algebra Appl. **21**, 194–220 (2014).
[25] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal. **37(6)**, 123–146 (2010).
[26] I. Pultarová, *Convergence theory of exact interpolation scheme for computing several eigenvectors*, Numer. Linear Algebra Appl. **23(2)**, 373–390 (2016).
[27] J. Ruge and K. Stüben, *Algebraic Multigrid*, Multigrid Methods. Frontiers in Applied Mathematics **3**, S.F. McCormick (Ed), pp. 73–130, SIAM (1987).
[28] V.V. Shaidurov, *Multigrid Methods for Finite Element*, Kluwer Academic Publics (1995).
[29] K. Stüben, *A review of algebraic multigrid*, J. Comput. Appl. Math. **128(1)**, 281–309 (2001).
[30] K. Stüben, *An introduction to algebra multigrid*, Academic Press 413–532 (2001).
[31] P. Vaněk, J. Mandel and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing **56(3)**, 179–196 (1996).
[32] P. Vaněk and I. Pultarová, *Convergence theory for the exact interpolation scheme with approximation vector as the first column of the prolongator and Rayleigh quotient iteration nonlinear smoother*, Appl. Math. **62(1)**, 49–73 (2017).
[33] H. Xie, *A multigrid method for eigenvalue problem*, J. Comput. Phys. **274**, 550–561 (2014).
[34] H. Xie, *A type of multilevel method for the Steklov eigenvalue problem*, IMA J. Numer. Anal. **34**,

592–608 (2014).
[35] H. Xie, M. Xie, X. Yin and M. Yue, *Computable error estimates for a nonsymmetric eigenvalue problem*, East Asian J. Appl. Math. **7**, 583–602 (2017).
[36] J. Xu and L. Zikatanov, *Algebraic multigrid methods*, Acta Numer. **26**, 591–721 (2017).
[37] Q. Zhai, H. Xie, R. Zhang and Z. Zhang, *The weak Galerkin method for elliptic eigenvalue problems*, Commun. Comput. Phys. **26**, 160–191 (2019).