Journal of Computational Mathematics
Vol.33, No.2, 2015, 209–226.
http://www.global-sci.org/jcm
doi:10.4208/jcm.1412-m2014-0041

A CASCADIC MULTIGRID ALGORITHM FOR COMPUTING THE FIEDLER VECTOR OF GRAPH LAPLACIANS[*]

John C. Urschel and Jinchao Xu
Department of Mathematics, Penn State University, Pennsylvania, USA
Email: jcurschel@gmail.com, xu@math.psu.edu
Xiaozhe Hu
Department of Mathematics, Tufts University, Medford, MA 02155
Email: xiaozhe.hu@tufts.edu
Ludmil T. Zikatanov
Department of Mathematics, Penn State University, Pennsylvania, USA
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
Email: ludmil@psu.edu

Abstract

In this paper, we develop a cascadic multigrid algorithm for fast computation of the Fiedler vector of a graph Laplacian, namely, the eigenvector corresponding to the second smallest eigenvalue. This vector has been found to have applications in fields such as graph partitioning and graph drawing. The algorithm is a purely algebraic approach based on a heavy edge coarsening scheme and pointwise smoothing for refinement. To gain theoretical insight, we also consider the related cascadic multigrid method in the geometric setting for elliptic eigenvalue problems and show its uniform convergence under certain assumptions. Numerical tests are presented for computing the Fiedler vector of several practical graphs, and numerical results show the efficiency and optimality of our proposed cascadic multigrid algorithm.

Mathematics subject classification: 65N55, 65N25.
Key words: Graph Laplacian; Cascadic Multigrid; Fiedler vector; Elliptic eigenvalue problems.

1. Introduction

Computation of the Fiedler vector of graph Laplacians has proven to be a relevant topic, and has found applications in areas such as graph partitioning and graph drawing [1]. There have been a number of techniques implemented for computation of the Fiedler vector, most notably by Barnard and Simon [2]. They implemented a multilevel coarsening strategy, using maximal independent sets and created a matching from them. For the refinement procedure, Rayleigh quotient iteration was used. We note that the term refinement refers to the smoothing process that occurs, and has a different meaning in the multigrid literature. Although at the time this was significantly faster than the standard recursive spectral bisection, it leaves room for improvement. The majority of the improvement has been in the form of coarsening algorithms. Better coarsening techniques, such as heavy edge matching (HEM), have been used more frequently, and have exhibited much shorter run times [3, 4].

[*] Received August 22, 2014 / Revised version received December 11, 2014 / Accepted December 22, 2014 / Published online March 13, 2015 /

For more general eigenproblems of symmetric positive definite matrices, techniques such as Jacobi-Davison [5] and the Locally Optimal Preconditioned Conjugate Gradient Method [6] (see also [7]) have been used and shown to give good approximations to eigenvalues and eigenvectors. These techniques can easily be extended to computing a Fiedler vector. Other eigensolvers are provided by setting an Algebraic MultiGrid (AMG) tuned specifically for graph Laplacians (see, e.g. Lean AMG [8]) as a preconditioner in the LOPCG Method.

In this paper, we introduce a new and fast coarsening algorithm, based on the conecpt of heavy edge matching, with a more aggressive coarsening procedure. For refinement, we implement a form of power iteration. For both our coarsening and refinement procedures we have created algorithms that are straightforward to implement. While heavy edge matching is complicated and tough to implement in high level programming languages, since it involves selecting an edge with heaviest weight between two unmatched vertices, heavy edge coarsening is significantly easier because we do not need to worry about whether a vertex has been aggregated or not. For the refinement procedure, power iteration does not require the inversion of a matrix, making its use much more straightforward than for Rayleigh quotient iteration, which requires some technique to approximately invert the matrix.

Based on these two improved components, we propose a cascadic multigrid (CMG) method to compute the Fiedler vector. The CMG method has been treated in the literature, most notably by Bornemann and Deuflhard [9,10], Braess, Deuflhard, and Lipnikov [11], and Shaidurov [12–14]. However, little has been done with respect to the elliptic eigenvalue problem. Our technique is a purely algebraic approach which only uses the given graph. Moreover, although the purely algebraic approach is technically difficult to analyze, we consider the CMG method for the elliptic eigenvalue problem in the geometric setting. Based on the standard smoothing property and approximation property, we show that the geometric CMG method converges uniformly for the model problem, which indirectly provides theoretical justification of the efficiency of the CMG method. This also shows the potential of our CMG method for solving other eigenvalue problems from different applications.

The remainder of the paper is organized as follows. In Section 2, we briefly review the Fiedler vector and introduce our cascadic multigrid method for computing the Fiedler vector of a graph Laplacian. The cascadic multigrid method for elliptic eigenvalue problems is proposed in Section 3 and its convergence analysis is also provided. Section 4 presents numerical experiments to support the theoretical results of CMG method for elliptic eigenvalue problems and demonstrate its efficiency for computing the Fiedler vector of some graph Laplacian problems from real applications. We conclude the paper in Section 5 by some general remarks on this work and proposed future work.

2. Cascadic MG Method for Computing the Fiedler Vector

We begin by formally introducing the concept of a graph Laplacian and Fiedler vector. We start with the concept of a graph. A weighted graph $G = (V, E, w)$ is said to be undirected if the edges have no orientation. A graph is a multigraph if $(i, i) \notin E$ for all $1 \leq i \leq |V|$ ($|V|$ is the number of vertices). For the remainder of this paper, we assume that all graphs are undirected and multigraphs.

We consider the task of representing a graph in matrix form. One of the most natural representations is through its Laplacian. The Laplacian of a graph is defined as follows:

Definition 2.1. *Let $G = (V, E, w)$ be a weighted graph. We define the Laplacian matrix of G,* *denoted $L(G) \in \mathbb{R}^{n\times n}$ (or just L for short), $n = |V|$, as follows:*

$$L(G)_{(i,j)} := \begin{cases} d_{v_i}, & for \quad i = j, \\ -w_{i,j}, & for \quad i \neq j, \end{cases}$$

where d_{v_i} is the degree of v_i, and $w_{i,j}$ is the weight of the edge connecting v_i and v_j.

The Laplacian $L(G)$ is self-adjoint, positive semi-definite, and diagonally dominant. In addition, the sum of any row (and also, any column) of L is zero. Therefore $\lambda = 0$ is an eigenvalue of L, with corresponding eigenvector $\mathbf{1} = (1, ..., 1)^T$. Let us order the eigenvalues of $L(G)$ as follows: $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$, and denote by $\varphi_1, \varphi_2, ..., \varphi_n$ the corresponding eigenvectors. We have already seen that $\varphi_1 = \alpha\mathbf{1}$. We now consider λ_2 and φ_2. This eigenvalue and eigenvector pair has special significance and, for this reason, are given special names.

Definition 2.2. *The algebraic connectivity of a graph G, denoted by $a(G)$, is defined to be the second smallest eigenvalue of the corresponding Laplacian matrix $L(G)$, with eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ and eigenvectors $\varphi_1, \varphi_2, ..., \varphi_n$. The eigenvector φ_2, corresponding to the eigenvalue $a(G)$, is called the Fiedler vector of G.*

The term Fiedler vector comes from the mathematician Miroslav Fiedler, who proved many results regarding the significance of this eigenvector. His work involving irreducible matrices and the Fiedler vector can be found in [15, 16].

We now introduce our cascadic MG (CMG) algorithm for computing the Fielder vector. Our CMG algorithm is a purely algebraic approach, and the multilevel structure is constructed from the graph directly. Therefore, similar to a standard algebraic MG (AMG) method, the new algorithm consists of three steps: a setup phase, a solving phase on the coarsest level, and a cascadic solving phase (also called refinement phase in our paper). The process works as follows:

- **Step 1:** Coarsen our graph G_0 iteratively to coarse graphs $G_1, G_2, ..., G_J$.

  Taking inspiration from AMG coarsening and graph matching, we introduce a technique we call heavy edge coarsening (HEC). At each level i, for the graph G_i with n_i vertices, this coarsening procedure produces aggregates G_i^m, $m = 1, 2, \cdots, n_{i+1}$ and restriction matrix $I_i^{i+1} \in \mathbb{R}^{n_{i+1}\times n_i}$ defined by

  $$(I_i^{i+1})_{pq} = 1, \quad \text{if } q \in G_i^p, \quad \text{and} \quad (I_i^{i+1})_{pq} = 0, \quad \text{if } q \notin G_i^p.$$

  The transpose of the restriction matrix is known as a prolongation. The coarser graph G_{i+1} is defined by designating the aggregates as the vertices of the coarse graph. Two aggregates are connected on this coarse graph if and only if there is an edge from G_i connecting a vertex from one aggregate and with a vertex from the other aggregate. This creates a multilevel structure of coarse Laplacians $L^0, L^1, ..., L^J$ where $L^{i+1} = I_i^{i+1} L^i (I_i^{i+1})^T$. In general, the choice of aggregates in the coarsening phase of a multilevel algorithm of this form tends to be the most expensive part of the procedure.

- **Step 2:** Solve for the Fiedler vector on the coarse graph G_J.

- **Step 3:** For $j = J$ to $j = 1$ we prolongate the Fiedler vector from the coarse graph G_j to the finer graph G_{j-1} and use the prolongated vector as an initial guess for a simple iterative procedure (such as power iteration). Such steps we call a "refinement" (or smoothing). We note that since we aim to approximate the Fiedler vector, we need to keep the iterates orthogonal to the constant vector.

Remark 2.1. In practice the coarse graph tends to be small in size (usually $|V| < 100$). The technique implemented on this level is not extremely relevant for single computations of the vector. However, for applications which may require this eigenvalue computation a large number of times (such as recursive spectral bisection, for large k), this becomes more of a relevant issue. The commonly used eigensolver on this coarse level, in the absence of a good intial guess, is the Lanczos algorithm. However, in our implementation we over-coarsen to $|V| < 25$ and use power iteration on a random vector, sampled from a Gaussian distribution.

Traditionally, in the MG method literature, **Step 2** and **Step 3** together are called the CMG method. However, in non-spectral methods for graph partitioning, this is not the case, and for this reason we maintain the three-step structure that is prevalent in the literature. We present the core of our cascadic eigensolver in Algorithm 2.1.

```
Algorithm 2.1 Multilevel Cascadic Eigensolver
  Input: graph Laplacian matrix $L^0 \in \mathbb{R}^{n_0 \times n_0}$
  Output: approximate Fiedler vector $\tilde{y}^{(0)}$
    Step 1: Setup Phase
    set $i = 0$
    while $n_i > 25$
      $I_i^{i+1} \leftarrow \mathrm{HEC}(L^i)$
      $L^{i+1} = I_i^{i+1} L^i (I_i^{i+1})^T$
      $i = i + 1$
    end while
      $J \leftarrow i$
    Step 2: Coarsest Level Solving Phase
      $\tilde{y}^{(J)} \leftarrow \mathrm{PI}(L^J, randn(n_J))$
    Step 3: Cascadic Refinement Phase
    for$j = J - 1$ to $0$ do
      $\hat{y}^{(j)} = (I_i^{i+1})^T \tilde{y}^{(j+1)}$
      $\tilde{y}^{(j)} \leftarrow \mathrm{PI}(L^j, \hat{y}^{(j)})$
     end for
```

Here, the subroutine HEC and PI are presented later in Algorithms 2.2 and 2.3, respectively. As mentioned before, because the size of the coarsest graph is very small, and power iteration is efficient, our focus is on the first and third steps. We will first introduce the heavy edge coarsening scheme we proposed for the setup phase, and then present our cascadic refinement scheme.

2.1. Heavy Edge Coarsening

We now consider the coarsening algorithm used for the setup phase. The goal for this step is to coarsen a graph quickly, while also maintaining some semblance of its structure. In practice, the coarsening procedure tends to dominate the run time of the multilevel eigensolver. To increase the efficiency of a coarsening algorithm one needs to make compromises between fast (with respect to computational time) and optimal (with respect to better representations of the graph on coarser graphs) coarsening techniques. We propose a new coarsening algorithm which combines ideas and algorithms described in the literature [1,3,4] and balances between reducing computational time and providing coarse graphs with good quality. In order to introduce our coarsening algorithm, we begin by considering matching as a coarsening technique. The formal concept of a matching is as follows:

Definition 2.3. *Let $G = (V, E)$. A matching is a subset $E^* \subset E$, such that no two elements of E^* are incident on the same vertex. A matching E^* is said to be a maximal matching if there does not exist an edge $e_{i,j} \in E \backslash E^*$ such that $E^* \cup \{e_{i,j}\}$ is still a matching.*

For our purposes, the matching computed at each level is always a maximal matching. A matching is computed at each level, and the edges in the matching are collapsed to form the coarser graph. We consider the class of matching algorithms concerned with finding the matching with the heaviest edge weight. A matching of heavy edges would make an ideal coarse graph for our multilevel eigenproblem. The reason for this is related to graph partitioning. This coarsening procedure creates a smaller edge cut on coarse levels for partitions, which results in smaller edge cuts for the finer graphs. Even though we are not refining partitions, this concept still applies, due to the close connection between the Fiedler vector and graph partitioning. To do a matching of heavy edges optimally is rather expensive because it would require searching for the heaviest weighed edge incident to two unmatched vertices at each step. In practice, the vertices are usually visited in a random order, and the heaviest weighed incident edge with an unmatched vertex is chosen. Such a technique produces a less optimal partition, but is much faster. We adopt a similar procedure in our coarsening algorithm.

However, we choose to perform a more aggressive coarsening procedure, rather than matching, because it reduces the number of levels in the multilevel scheme. In addition, when considering using heavy edge schemes, an aggressive coarsening procedure (see Algorithm 2.2) is significantly easier to implement than its matching counterpart because we consider mapping each vertex to a vertex incident with it with heaviest edge, rather than picking the heaviest edge with an unmatched vertex.

We visit the vertices in a random order. At each vertex we visit, we check if it has been mapped to some aggregate. If the vertex is unmapped, we map the vertex to the aggregate containing the adjacent vertex with the heaviest connecting edge. If the vertex already belongs to an aggregate, we skip it and continue to the next vertex. We finish when all vertices have been visited and belong to some aggregate. In general, this will not result in a matching. We call this technique heavy edge coarsening (HEC) and for the specific details regarding its implementation, we refer to Algorithm 2.2.

Remark 2.2. As an example, for a graph Laplacian corresponding to an anisotropic problem, one would end up with aggregates that contains vertices in lines pointing in the "strong" direction. This procedure would effectively only coarsen in the "strong" direction initially.

The HEC procedure proves to be a fast and efficient means of coarsening. The structure of the finer graph is well represented, making the refinement process of power iteration converge quickly. In addition, one of the biggest benefits of HEC is the relatively small number of coarse levels required. We will introduce this concept, in the form of a lemma.

Lemma 2.1. *Let $G_i = (V_i, E_i)$ be a connected graph with n_i vertices. Let n_{i+1}^{HEC} be the number of aggregates formed by heavy edge coarsening, and n_{i+1}^{M} be the number of aggregates formed by matching. Define the coarsening rate $k_{HEC}^i = n_{i+1}^{HEC}/n_i$ and $k_M^i = n_{i+1}^M/n_i$, respectively, Then we have $1/n_i \leq k_{HEC}^i \leq 0.5$ and $0.5 \leq k_M^i \leq 1$.*

Proof. From the definition of a matching, we have that n_{i+1}^M cannot be less than half of n_i. For the bounds on k_{HEC}^i, we note that for our HEC algorithm, every node in V_i is mapped to another node, or has been mapped to, which implies that each aggregation has at least two vertices, i.e. the average n_i/n_{i+1}^{HEC} is bigger than or equal to 2. Therefore, n_{i+1}^{HEC} is at most half of n_i. The lower bound results from taking a HEC procedure on a graph G_i such that G_{i+1} is a single node. □

Algorithm 2.2 Heavy Edge Coarsening (HEC)

```
Input: graph Laplacian matrix $L^i \in \mathbb{R}^{n_i \times n_i}$
Output: restriction matrix $I_i^{i+1}$
  $c \leftarrow 0$
  $p \leftarrow randperm(n_i)$
  $q \leftarrow zeros(n_i, 1)$
  for $i = 1$ to $n_i$ do
    if $q(p(i)) = 0$
      $m \leftarrow argmin(L(:, p(i)))$
      if $q(m) = 0$
        $c \leftarrow c + 1$
        $q(m) = c$
        $q(p(i)) = c$
      else
        $q(p(i)) = q(m)$
      end if
    end if
  end for
  $I_i^{i+1} \leftarrow zeros(c, n_i)$
  for $i = 1$ to $n_i$
    $I_i^{i+1}(q(i), i) = 1$
  end for
```

We have given a bound for the value of k_{HEC} (we drop the superscript i for simplicity). Given below in Table 2.1 are samples of what values k_{HEC} takes in practice for different graphs. As expected, the values taken in practice are significantly below the given bound of 0.5.

What remains to be explored is the properties of the restiction matrix I_i^{i+1}. The most important fact that we require is that the coarse matrix created by the restriction matrix is still a Laplacian matrix of the coarse graph. In addition, we want to inspect whether or not the constant eigenvector $\mathbf{1} = (1, ..., 1)^T$ is preserved under restrictions and prolongations. We also consider issues of orthogonal solutions with respect to the refinement procedure. Those

Table 2.1: Sample values of k_{HEC}.

Graph	Sample k_{HEC}^0 Value
144	0.1893
598a	0.2024
auto	0.1742

properties are summarized in the following proposition (see also [17, Theorem 3.6] for such results).

Proposition 2.1. *Let $I_i^{i+1} \in \mathbb{R}^{n_{i+1}\times n_i}$ be a restriction matrix defined by HEC. Then we have the following:*

1. $(I_i^{i+1})^T \mathbf{1}^{i+1} = \mathbf{1}^i$. *That is, the eigenvector* $\mathbf{1}$ *is preserved under refinement.*
2. *If L^i is a Laplacian matrix, then $L^{i+1} = I_i^{i+1} L^i (I_i^{i+1})^T$ is also a Lapacian matrix. In particular, $L^{i+1}\mathbf{1}^{i+1} = 0$.*
3. *Let $u \in \mathbf{1}^\perp = \{u|(u,\mathbf{1}) = 0\} \subset \mathbb{R}^{n_i}$. Then $I_i^{i+1}u \in \mathbf{1}^\perp \subset \mathbb{R}^{n_{i+1}}$. However, in general, $(I_{i-1}^i)^T u \notin \mathbf{1}^\perp \subset \mathbb{R}^{n_{i-1}}$.*

Proof. We begin with (1). This follows from the fact that each vertex in V_i is mapped to only one vertex in V_{i+1}. However, $I_i^{i+1}\mathbf{1}^i \neq \mathbf{1}^{i+1}$. This is expected, as the number of vertices in V_i mapped to a given vertex $v_j \in V_{i+1}$ varies.

To prove (2), we need to show that L^{i+1} is still symmetric, with positive diagonal and non-positive offdiagonal, with $L^{i+1}\mathbf{1}^{i+1} = 0$. We begin by decomposing L^i into its degree matrix D^i and adjacency matrix A^i. This gives us

$$L^{i+1} = I_i^{i+1} D^i (I_i^{i+1})^T - I_i^{i+1} A^i (I_i^{i+1})^T.$$

$I_i^{i+1} D^i (I_i^{i+1})^T$ is still a degree matrix, and $I_i^{i+1} A^i (I_i^{i+1})^T$ an adjacency matrix. We show that L^{i+1} is a Laplacian by taking

$$L^{i+1}\mathbf{1}^{i+1} = I_i^{i+1} L^i (I_i^{i+1})^T \mathbf{1}^{i+1} = I_i^{i+1} L^i \mathbf{1}^i = 0.$$

Part (3) of the Proposition can be shown as follows. Let $u \in \mathbf{1}^\perp \subset \mathbb{R}^{n_i}$. We have

$$(I_i^{i+1}u, \mathbf{1}^{i+1}) = (u, (I_i^{i+1})^T \mathbf{1}^{i+1}) = (u, \mathbf{1}^i) = 0.$$

Therefore, $I_i^{i+1}u \in \mathbf{1}^\perp \subset \mathbb{R}^{n_{i+1}}$. Looking at $(I_{i-1}^i)^T u$, we see

$$((I_{i-1}^i)^T u, \mathbf{1}^{i-1}) = (u, I_{i-1}^i \mathbf{1}^{i-1}) \neq (u, \mathbf{1}^i) = 0,$$

since $I_{i-1}^i \mathbf{1}^{i-1} \neq \mathbf{1}^i$. □

2.2. Refinement (Smoothing) Strategies

Given an approximate Fiedler vector $y^{(i+1)}$ on a coarse graph G_{i+1}, we aim to find an optimal manner to project this vector back to the finer graph G_i and refine it to an approximate Fiedler vector $y^{(i)}$ on G_i. We begin by considering the projection problem. The most natural way to project $y^{(i+1)}$ to G_i is to use the restriction matrix I_i^{i+1} obtained from coarsening, define our prolongation matrix to be $(I_i^{i+1})^T$, and let the initial approximation be $\tilde{y}^{(i)} = (I_i^{i+1})^T y^{(i+1)}$. However, we have to concern ourselves with orthogonality to the eigenvector $\mathbf{1}$. From Proposition 2.1, we have that $(\tilde{y}^{(i)}, \mathbf{1}^i) \neq 0$. Therefore, before we can perform any sort of eigenvalue refinement procedure, we require our inital vector to be in the subspace $\mathbf{1}^\perp = \{u|(u, \mathbf{1}) = 0\}$. This can be accomplished by one iteration of Gram-Schmidt. From here, the orthogonality will be approximately maintained, since $\mathbf{1}^\perp$ is L-invariant.

Given an approximation $\tilde{y}^{(i)}$, we can refine it in a number of ways. We consider power iteration as a refinement scheme in our CMG algorithm because of its simplicity. In this way we take advantage of the sparsity of our Laplacian.

Because the Fiedler vector corresponds to the second smallest eigenvalue of the graph Laplacian, we cannot apply the power iteration directly. Therefore, we compute a Gershgorin bound on the eigenvalues of a Laplacian L by considering $g = \|L\|_{\ell^1}$. From the Gershgorin circle Theorem and properties of the Laplacian, we have that all the eigenvalues of $gI - L$ are positive, with eigenvalues $g - \lambda_1, g - \lambda_2, ..., g - \lambda_n$. The eigenvectors obviously remain unchanged. In this way it suffices to perform power iteration on $gI - L$, coupled with an intial orthogonalization to $\mathbf{1}$. We note that $\mathbf{1}^\perp$ is also invariant under $gI - L$. This variant of power iteration is detailed in Algorithm 2.3.

We proceed by examining the convergence for power iteration. Let u^0 denote our initial guess, and u^k represent the normalized vector resulting from k iterations. For our algorithm, the stopping criterion is given by $(u^k, u^{k-1}) > 1 - \delta$, for some given tolerance δ. We note that this is equivalent to $||u^k - u^{k-1}||^2 < 2\delta$. We recall the following result, with respect to power iteration on an arbitrary symmetric matrix.

Theorem 2.1. *Let A be a symmetric matrix with eigenvalues $\lambda_1 > \lambda_2 \geq ... \geq \lambda_n \geq 0$ and corresponding eigenvectors $\varphi_1, \varphi_2, ..., \varphi_n$. Then power iteration, with intial guess u^0, $(u^0, \varphi_1) \neq 0$, has convergence rate given by*

$$\sin \angle(u^k, \varphi_1) < \left|\frac{\lambda_2}{\lambda_1}\right|^k \tan \angle(u^0, \varphi_1),$$

where $\angle(u, v)$ is the angle between the subspaces spanned by u and v.

A proof of this result can be found in [18]. We see that the number of iterations required depends on the eigenvalue gap, the quality of the initial guess in our multilevel structure, as well as the chosen tolerance. For general graphs it is hard to obtain better estimates for the power iteration portion of the cascadic algorithm. This stems mainly from the fact that the eigenvalues of a general graph does not follow any set spacing or structure, and that our aggregation procedure is random in nature, making an estimate of the quality of the initial approximation extremely tough in practice. This limits our ability to give rigorous theoretical results for our algorithm in general. In Section 3 we will give results for our cascadic eigenvalue algorithm for the case of graphs resulting from elliptic PDE discretizations, with geometric coarsening as the cascadic coarsening procedure and a fixed number of power iteration steps at each level. These simplifications remove the barriers that we currently face for analysis. However, we will give numerical justification that these results are robust to general graphs with HEC as the coarsening procedure.

```
**Algorithm 2.3** Power Iteration (PI)
  **Input**: graph Laplacian matrix $L \in \mathbb{R}^{n\times n}$, initial guess $\tilde{y}^0$
  **Output**: approximate Fiedler vector $\tilde{y}$
    $g = max_i \sum_{1\le j\le n} |l_{i,j}|$
    $B_g = gI - L$
    $u = \tilde{y}^0 - \frac{\mathbf{1}^T\tilde{y}^0}{n}\tilde{y}^0$
    $u \leftarrow \frac{u}{\|u\|}$
    $v \leftarrow zeros(n,1)$
    **while** $u^Tv < 1 - tol$
      $v \leftarrow u$
      $u = B_g v$
      $u \leftarrow \frac{u}{\|u\|}$
    **end while**
    $\tilde{y} = u$
```

3. Convergence Analysis of CMG for Elliptic Eigenvalue Problems

In Section 2, we introduced the CMG method for computing the Fiedler vector of a graph Laplacian. However, we used a purely algebraic coarsening strategy (see Section 2.1) to construct the hierarchical structure; hence, similar to the AMG method for the Poisson problem, the convergence analysis for a purely algebraic CMG method is difficult. In order to illustrate and theoretically justify the convergence of the proposed CMG method, we discuss the geometric CMG (GCMG) method for the elliptic eigenvalue problem. As a model which shares a great deal of properties with the graph Laplacian eigenproblem, we consider the following elliptic eigenvalue problem with Neumann boundary conditions,

$$-\Delta\varphi = \lambda\varphi, \quad \text{on } \Omega, \quad \frac{\partial\varphi}{\partial n} = 0, \quad \text{on } \partial\Omega \tag{3.1}$$

where $\Omega \in \mathbb{R}^d$ is a polygonal Lipschitz domain. We only consider the two- and three- dimensional case to illustrate the theoretical bounds that can be obtained for the cascadic multilevel algorithm. However, the GCMG method we discussed here can be naturally applied for higher dimentional cases. Using the standard Sobolev space $H^1(\Omega)$, we consider the weak formulation of (3.1) as follows: find $(\lambda,\varphi) \in \mathbb{R}\times H^1(\Omega)$ such that

$$a(\varphi, v) = \lambda(\varphi, v), \qquad \forall\ v \in H^1(\Omega), \tag{3.2}$$

where the bilinear form $a(u,v) = (\nabla u, \nabla v)$, and $(\cdot,\cdot)$ is the standard L^2 inner product. Here, the bounded symmetric bilinear form $a(\cdot,\cdot)$ is coercive on the quotient space $H^1(\Omega)$, and, therefore, induces an energy-norm as follows:

$$\|u\|_a^2 = a(u,u), \qquad \forall\ u \in H^1(\Omega)\backslash\mathbb{R}. \tag{3.3}$$

Moreover, we denote the L^2-norm by $\|\cdot\|$ as usual. Similar to the eigenvalues for the graph Laplacian, $\lambda = 0$ is also an eigenvalue of the eigenvalue problem (3.2), We can order the eigenvalues as follows: $0 = \lambda^{(1)} \le \lambda^{(2)} \le ...$ and denote by $\varphi^{(1)}, \varphi^{(2)}, ...$ the corresponding eigenfunctions. Again, we are interested in approximating the second smallest eigenvalue of

(3.2) and its corresponding eigenfunction space.

Given a nested family of quasi-uniform triangulations $\{\Gamma_j\}_{j=0}^J$, namely,

$$\frac{1}{c}2^{j-J} \leq h_j = \max_{T\in\Gamma_j} \mathrm{diam}(T) \leq c2^{j-J},$$

the spaces of linear finite elements are

$$V_j = \Big\{u \in C(\Omega) : u|_T \in P_1(T), \qquad \forall\, T \in \Gamma_j\Big\},$$

where $P_1(T)$ denotes the linear functions on the triangle T. We have

$$V_J \subset V_{J-1} \subset \cdots \subset V_0 \subset H^1(\Omega).$$

The finite element approximations of (3.2) on each level are as follows: find $(\lambda_j, \varphi_j) \in \mathbb{R} \times V_j$ such that

$$a(\varphi_j, v_j) = \lambda_j(\varphi_j, v_j), \qquad \forall\, v_j \in V_j. \tag{3.4}$$

We can order the eigenvalues as follows: $0 = \lambda_j^{(1)} \leq \lambda_j^{(2)} \leq \cdots \leq \lambda_j^{(N_j)}$ and denote by $\varphi^{(1)}, \varphi^{(2)}, ...\varphi^{(N_j)}$ the corresponding eigenfunctions. Again, we are interested in approximating the second smallest eigenpair on the finest level. Moreover, we can define an operator A_j by $a(u_j, v_j) = (A_j u_j, v_j)$, $\forall u_j, v_j \in V_j$.

We assume the elliptic eigenvalue problem has $H^{1+\alpha}$-regularity, i.e., the eigenvalue function $\varphi \in H^{1+\alpha}$ for some $0 < \alpha \leq 1$. Then we have the following error estimates regarding the standard finite element approximation of the elliptic eigenvalue problem, taken from the work of Babuška and Osborn [19].

Lemma 3.1. *Assume that* $(\lambda_h, \varphi_h) \in (\mathbb{R}\times V_h)$ *is a finite element approximation of* (3.2). *Then we have*

(i) $|\lambda - \lambda_h| \leq Ch^{2\alpha}$,

(ii) *there exists an eigenfunction* φ *corresponding to* λ*, such that*

$$\|\varphi - \varphi_h\|_a \leq Ch^{\alpha}, \tag{3.5}$$

where C *is a constant that does not depend on the mesh size.*

Now we introduce the Ritz projection on level j by $a(P_j u, v_j) = a(u, v_j)$, $\forall v_j \in V_j$. We assume the eigenvalue $\lambda^{(l)}$ we want to approximate has multiplicity k, i.e. $\lambda^{(l)} = \lambda^{(l+1)} = \cdots = \lambda^{(l+k-1)}$ and there are k corresponding eigenfunctions $\varphi^{(l)}, \varphi^{(l+1)}, \varphi^{(l+k-1)}$, then on level j, there are k approximate eigenpairs $(\lambda_j^{(l+i)}, \varphi_j^{(l+i)})$, $i = 0, \cdots k-1$, such that

$$\lambda_j^{(l)} \leq \lambda_j^{(l+1)} \leq \cdots \leq \lambda_j^{(l+k-1)}.$$

Let Q_j denote the L^2-projection onto $\mathrm{span}\{\varphi_j^{(l)}, \varphi_j^{(l+1)}, \cdots, \varphi_j^{(l+k-1)}\}$ and define $\Lambda_j := Q_j \circ P_j$. Then for an eigenfunction $\varphi^{(l+i)}$, $i = 0, 1, \cdots, k-1$, $\Lambda_j \varphi^{(l+i)} \in \mathrm{span}\{\varphi_j^{(l)}, \varphi_j^{(l+1)}, \cdots, \varphi_j^{(l+k-1)}\}$ is regarded as its approximation. The following best-approximation result of $\Lambda_j \varphi^{(l+i)}$ can be found in [20]. For the simplicity of the presentation, we omit the superscript $(l + i)$.

Lemma 3.2. *Assume that h_j is sufficiently small and the elliptic eigenvalue problem has $H^{1+\alpha}$-regularity, then for any eigenpair (λ, φ) with $\|\varphi\| = 1$, we have*

$$\|\varphi - \Lambda_j \varphi\|_a \le C h_j^{\alpha}, \tag{3.6}$$

where C is a constant that does not depend on the mesh size.

Next, we will present several results related to the approximation property of finite element approximate eigenfunctions between two successive levels j and $j+1$. For the sake of simplicity, we will use script h to denote level j and script H to denote level $j+1$. Moreover, we denote the mesh size h_j by h and h_{j+1} by H. Considering the eigenvalue problem on level $j+1$ as a finite element approximation of the eigenvalue problem on level j, we have the following lemma regarding the approximation in the energy norm.

Lemma 3.3. *Let $\{(\lambda_h^{(l+i)}, \varphi_h^{(l+i)})\}_{i=0}^{i=k-1}$ and $\{(\lambda_H^{(l+i)}, \varphi_H^{(l+i)})\}_{i=0}^{i=k-1}$ be approximate eigenpairs of the eigenvalue $\lambda^{(l)}$ with multiplicity k. For sufficiently small H and for any $w_h \in span\{\varphi_h^{(l)}, \varphi_h^{(l+1)}, \cdots, \varphi_h^{(l+k-1)}\}$ we have*

$$\|w_h - \Lambda_H w_h\|_a \le C H^{\alpha}, \tag{3.7}$$

Proof. Setting $w_h = \sum_{i=0}^{k-1} \beta_i \varphi_h^{(l+i)}$ we have,

$$\begin{aligned} &\|w_h - \Lambda_H w_h\|_a \\ &= \|\sum_{i=0}^{k-1} \beta_i \left(\varphi_h^{(l+i)} - \Lambda_H \varphi_h^{(l+i)}\right)\|_a \le \sum_{i=0}^{k-1} |\beta_i| \| \left(\varphi_h^{(l+i)} - \Lambda_H \varphi_h^{(l+i)}\right)\|_a \\ &\le C H^{\alpha}. \end{aligned}$$

This completes the proof. □

The next lemma provides an estimate on the error of approximation $(w_h - \Lambda_H w_h)$ in the L^2 norm. In the proof, we use a separation bound given in Boffi [21], namely, that for sufficiently small H the following estimate holds:

$$\frac{|\lambda_h^{(l)}|}{|\lambda_h^{(l)} - \lambda_H^{(i)}|} \le d_l < \infty, \quad \text{for all } i \ne l, l+1, ..., l+k-1. \tag{3.8}$$

The L^2 estimate is then as follows.

Lemma 3.4. *Let $\{(\lambda_h^{(l+i)}, \varphi_h^{(l+i)})\}_{i=0}^{i=k-1}$ and $\{(\lambda_H^{(l+i)}, \varphi_H^{(l+i)})\}_{i=0}^{i=k-1}$ be approximate eigenpairs of the eigenvalue $\lambda^{(l)}$ with multiplicity k. For sufficiently small H and for any $w_h \in span\{\varphi_h^{(l)}, \varphi_h^{(l+1)}, \cdots, \varphi_h^{(l+k-1)}\}$, we have*

$$\|(I - \Lambda_H) w_h\| \le C \|(I - P_H) w_h\|, \tag{3.9}$$

where C is a constant that does not depend on the mesh size.

Proof. Let us first set $w_h = \sum_{j=l}^{l+k-1} \beta_j \varphi_h^{(j)}$. Because $P_H w_h \in V_H$, we have $P_H w_h = \sum_{i=1}^{N_H} \alpha_i \varphi_H^{(i)}$ where $\alpha_i = (P_H w_h, \varphi_H^{(i)})$. Since by the definition of Q_H we have that $Q_H \varphi_H^{(l+i)} = \varphi_H^{(l+i)}$ for $i = 0, \ldots, k-1$, it is straightforward to calculate that

$$P_H w_h - \Lambda_H w_h = \sum_{i \ne l, l+1, \cdots, l+k-1} \alpha_i \varphi_H^{(i)}.$$

Next, using the relation

$$\lambda_H^{(i)}(P_H\varphi_h^{(j)},\varphi_H^{(i)}) = a(P_H\varphi_h^{(j)},\varphi_H^{(i)}) = a(\varphi_h^{(j)},\varphi_H^{(i)}) = \lambda_h^{(j)}(\varphi_h^{(j)},\varphi_H^{(i)}),$$

we obtain that

$$(\lambda_H^{(i)}-\lambda_h^{(j)})(P_H\varphi_h^{(j)},\varphi_H^{(i)}) = \lambda_h^{(j)}(\varphi_h^{(j)}-P_H\varphi_h^{(j)},\varphi_H^{(i)}).$$

Therefore,

$$\begin{aligned}\|P_Hw_h-\Lambda_Hw_h\|^2 = (P_Hw_h, P_Hw_h-\Lambda_Hw_h) &= \sum_{i\neq l,l+1,\cdots,l+k-1}(P_Hw_h,\varphi_H^{(i)})^2\\ &= \sum_{i\neq l,l+1,\cdots,l+k-1}\left(\sum_{j=l}^{l+k-1}\beta_j(P_H\varphi_h^{(j)},\varphi_H^{(i)})\right)^2\\ &= \sum_{i\neq l,l+1,\cdots,l+k-1}\left(\sum_{j=l}^{l+k-1}\beta_j\frac{\lambda_h^{(j)}}{\lambda_H^{(i)}-\lambda_h^{(j)}}(\varphi_h^{(j)}-P_H\varphi_h^{(j)},\varphi_H^{(i)})\right)^2\\ &\le d_l^2\left(\sum_{i\neq l,l+1,\cdots,l+k-1}(w_h-P_Hw_h,\varphi_H^{(i)})^2\right)\\ &= d_j^2\|w_h-P_Hw_h\|^2.\end{aligned}$$

Moreover, we have

$$\|w_h-\Lambda_Hw_h\| \le \|w_h-P_Hw_h\| + \|P_Hw_h-\Lambda_Hw_h\| \le (1+d_l)\|(I-P_H)w_h\|,$$

which leads to (3.9) with $C=1+d_l$. □

Based on Lemma 3.4 and the interpolation argument [22], we have the following *approximation property* for the eigenvalue problem.

Lemma 3.5. *Let* $\{(\lambda_h^{(l+i)},\varphi_h^{(l+i)})\}_{i=0}^{i=k-1}$ *and* $\{(\lambda_H^{(l+i)},\varphi_H^{(l+i)})\}_{i=0}^{i=k-1}$ *be approximate eigenpairs of the eigenvalue* $\lambda^{(l)}$ *with multiplicity* k*. Assuming that* H *is sufficiently small, for any* $w_h\in$ *span*$\{\varphi_h^{(l)},\varphi_h^{(l+1)},\cdots,\varphi_h^{(l+k-1)}\}$*, we have*

$$\|(I-\Lambda_H)w_h\|_{H^{1-\alpha}} \le CH^\alpha\|(I-\Lambda_H)w_h\|_a \tag{3.10}$$

where C *is a constant independent of the mesh size.*

Proof. From Lemma 3.4, we have

$$\begin{aligned}\|(I-\Lambda_H)w_h\| \le C\|(I-P_H)w_h\| &= C\|(I-P_H)[(I-P_H)w_h]\|\\ &\le CH\|(I-P_H)w_h\|_a \le CH\|(I-\Lambda_H)w_h\|_a,\end{aligned}$$

where the last inequality follows from noting that $\|(I-P_H)w_h\|_a=\inf_{v\in V_H}\|w_h-v\|_a$. By an interpolation argument, the desired result follows. □

Based on the nested spaces $V_J\subset V_{J-1}\subset\cdots\subset V_0$, the GCMG method for eigenvalue problems seeks to solve the eigenvalue problem exactly on the coarse grid V_J, and interpolate and smooth the approximation back to the fine grid V_0. In this section, we consider the GCMG method, and therefore, the geometric prolongation and restriction are used in our algorithm, and will be omitted as usual. Our cascadic Algorithm 2.1 can be framed as follows:

```
Algorithm 3.1 Geometric Cascadic Multigrid Method for Elliptic Eigenvalue Problem
  if $j = J$ (coarsest level)
      solve $a(\varphi_J, v_J) = \lambda(\varphi_J, v_J)$ exactly, and let $u_J := \varphi_J^{(l)}$
  else
      $u_j = (I - \omega_j A_j)^{k_j} u_{j+1}$, where $\omega_j = \|A_j\|_\infty^{-1}$. (with appropriate scaling)
      $\lambda_j = \frac{a(u_j,u_j)}{(u_j,u_j)}$
  end if
```

Remark 3.1. We present the algorithm for just computing one approximate eigenpair. However, we can easily extend the algorithm to compute several approximate eigenpairs by starting with k approximate eigenpairs on the coarest level and then, on each level, after smoothing each approximate eigenfunction, we can orthogonalize them and compute corresponding Rayleigh quotients.

This procedure is only performed once and results in the approximation $u_0 \in V_0$. Next, we consider the uniform convergence of the proposed GCMG method (Algorithm 3.1). Our analysis will follow the standard convergence analysis for the CMG method for elliptic partial differential equations. We will first present a two-level error estimate on two successive levels $j+1$ and j, and then generalize it to the multilevel case later. Again, we use h to denote $j+1$ and H to denote j for the sake of simplicity. We begin by recalling the following lemma.

Lemma 3.6. *For any $k \in \mathbb{Z}_+$, we have $\max_{t\in[0,1]} t(1-t)^k < \frac{1}{k+1}$.*

This is a simple result, and is used often in multigrid literature. Denoting by $S_h = I - \omega_h A_h$ the error propagation operator associated with the Richardson smoother, we have the following *smoothing property.*

Lemma 3.7. *Let $\omega = ||A_h||_\infty^{-1}$ and k be the number of smoothing steps. Then the following estimate holds*

$$\|S_h^k v_h\|_a \le C\frac{h^{-\alpha}}{k^{\alpha/2}}\|v_h\|_{H^{1-\alpha}}, \quad \forall\ v_h \in V_h\backslash\mathbb{R}. \tag{3.11}$$

Proof. Recall that, by the properties of a graph Laplacian, A_h is Hermitian and positive semi-definite, and, moreover, A_h is positive definite on the subspace $\{u \mid (u, \mathbf{1}) = 0\}$. Hence,

$$\begin{aligned}||S_h^\nu u||_a^2 &= \big((I-\omega A_h)^\nu u, (I-\omega A_h)^\nu u\big)_a \\ &= \big(A_h(I-\omega A_h)^\nu u, (I-\omega A_h)^\nu u\big) \\ &= \omega^{-1}\big(\omega A_h(I-\omega A_h)^{2\nu} u, u\big).\end{aligned}$$

Noting that the spectral radius $\rho(\omega A_h) \le 1$, $\omega^{-1} \eqsim h^{-2}$ and making use of Lemma 3.6, we obtain

$$||S_h^\nu u||_a^2 \lesssim h^{-2}\eta_0(2\nu)||u||^2.$$

This gives us

$$\|S_h^k v_h\|_a \le C\frac{h^{-1}}{k^{1/2}}\|v_h\|, \quad \forall\ v_h \in V_h\backslash\mathbb{R}.$$

Recalling that S_h is a contraction, and, hence, $\|S_h^k v_h\|_a \le C\|v_h\|_a$, for all $v_h \in V_h$, the desired result follows by an interpolation argument.

We are now able to show the uniform convergence of our GCMG Algorithm 3.1 under suitable conditions.

Lemma 3.8. *Let* $\{(\lambda_h^{(l+i)}, \varphi_h^{(l+i)})\}_{i=0}^{i=k-1}$ *and* $\{(\lambda_H^{(l+i)}, \varphi_H^{(l+i)})\}_{i=0}^{i=k-1}$ *be approximate eigenpairs of the eigenvalue* $\lambda^{(l)}$ *with multiplicity* k *and* u_h *be computed by Algorithm* 3.1*. Assuming that* H *is sufficiently small, there exist* $\varphi^h \in span\{\varphi_h^{(l)}, \varphi_h^{(l+1)}, \cdots, \varphi_h^{(l+k-1)}\}$ *and* $\varphi^H \in span\{\varphi_H^{(l)}, \varphi_H^{(l+1)},$
$\cdots, \varphi_H^{(l+k-1)}\}$ *such that the error of the two-level GCMG Algorithm* 3.1 *with the Richardson smoother for the eigenvector can be estimated by*

$$\|u_h - \varphi^h\|_a \le C\frac{h^\alpha}{k^{\alpha/2}} + \|u_H - \varphi^H\|_a, \tag{3.12}$$

where k *is the number of smoothing steps and* C *is a constant that does not depends on mesh size.*

Proof. Denote $e_H = u_H - \varphi^H$, we have $u_H = \varphi^H + e_H$. Let $\bar\varphi^h \in \text{span}\{\varphi_h^{(l)}, \varphi_h^{(l+1)}, \cdots, \varphi_h^{(l+k-1)}\}$ satisfy $\varphi^H = \Lambda_H \bar\varphi^h$, then we have

$$u_H = \bar\varphi^h + (\varphi^H - \bar\varphi^h) + e_H.$$

Let $\bar\varphi^h = \sum_{i=l}^{l+k-1} \beta_i \varphi_h^{(i)}$. We have

$$\begin{aligned} & u_h \\ &= S_h^k \bar\varphi^h + S_h^k(\varphi^H - \bar\varphi^h) + S_h^k e_H \\ &= \sum_{i=l}^{l+k-1} \beta_i \left(\frac{\omega_h^{-1} - \lambda_h^{(i)}}{\omega_h^{-1}}\right)^k \varphi_h^{(i)} + S_h^k(\varphi^H - \bar\varphi^h) + S_h^k e_H. \end{aligned}$$

Denote $\varphi^h := \sum_{i=l}^{l+k-1} \beta_i \left(\frac{\omega_h^{-1} - \lambda_h^{(i)}}{\omega_h^{-1}}\right)^k \varphi_h^{(i)} \in \text{span}\{\varphi_h^{(l)}, \varphi_h^{(l+1)}, \cdots, \varphi_h^{(l+k-1)}\}$, we have

$$e_h := u_h - \varphi^h = S_h^k(\varphi^H - \bar\varphi^h) + S_h^k e_H.$$

Therefore,

$$\begin{aligned} \|e_h\|_a &\le \|S_h^k(\varphi^H - \bar\varphi^h)\|_a + \|S_h^k e_H\|_a \\ &\le C\frac{h^{-\alpha}}{k^{\alpha/2}} \|\varphi^H - \bar\varphi^h\|_{H^{1-\alpha}} + \|e_H\|_a \qquad \text{(from Lemma 3.7)} \\ &\le C\frac{1}{k^{\alpha/2}} \|\Lambda_H \bar\varphi^h - \bar\varphi^h\|_a + \|e_H\|_a \qquad \text{(from Lemma 3.5)} \\ &\le C\frac{H^\alpha}{k^{\alpha/2}} + \|e_H\|_a \qquad \text{(from Lemma 3.2)}. \end{aligned}$$

Finally, (3.12) follows by noting that $2h/c \le H \le c2h$. □

By recursively applying the two-level result Lemma 3.8 on two successive levels $j+1$ and j, we can derive the error estimate of the multilevel GCMG. From now on, we use the script j

again to denote the index of the level. Because $2^j h_0/C \le h_j \le C2^j h_0$, we consider $k_j = \beta^j k_0$ for some fixed $\beta > 0$. We have the following error estimate.

Theorem 3.1. *Let $\{\lambda_0^{(l+i)}\varphi_0^{(l+i)}\}_{i=0}^{k-1}$ be approximate eigenpairs of the eigenvalue $\lambda^{(l)}$ with multiplicity k and u_0 be computed by Algorithm 3.1. Let the number of smoothing steps on level j be given by $k_j = \beta^j k_0$. If h_J is sufficiently small, then there exists $\varphi^0 \in span\{\varphi_0^{(l)}, \varphi_0^{(l+1)}, \cdots, \varphi_0^{(l=k-1)}\}$ such that the error of the GCMG method for the eigenvector can be estimated by*

$$\|u_0 - \varphi^0\|_a \le \begin{cases} C\frac{1}{1-(4/\beta)^{\alpha/2}}\frac{h_0^\alpha}{k_0^{\alpha/2}}, & \text{if } \beta > 4, \\ CJ\frac{h_0^\alpha}{k_0^{\alpha/2}}, & \text{if } \beta = 4. \end{cases} \tag{3.13}$$

and for the eigenvalue, by

$$|\lambda_0 - \lambda^0| \le \begin{cases} C(\frac{1}{1-(4/\beta)^{\alpha/2}})^2\frac{h_0^{2\alpha}}{k_0^{\alpha}}, & \text{if } \beta > 4, \\ CJ^2\frac{h_0^{2\alpha}}{k_0^{\alpha}}, & \text{if } \beta = 4, \end{cases} \tag{3.14}$$

where C denotes a constant that does not depend on the mesh size.

Proof. Using the two level result from Lemma 3.8,

$$\|u_{j+1} - \varphi^{j+1}\|_a \le C\frac{h_j^\alpha}{k_j^{\alpha/2}} + \|u_j - \varphi^j\|_a,$$

summing from $j = J-1$ to 0, and noting that $e_J = 0$, we have

$$\|u_0 - \varphi^0\|_a \le C\sum_{j=0}^{J-1}\frac{h_j^\alpha}{k_j^{\alpha/2}}.$$

Moreover, using the identity

$$\lambda_0 - \lambda^0 = \frac{a(u_0 - \varphi^0, u_0 - \varphi^0)}{(u_0, u_0)} - \lambda^0\frac{(u_0 - \varphi^0, u_0 - \varphi^0)}{(u_0, u_0)},$$

we have

$$|\lambda_0 - \lambda^0| \le C\left(\sum_{j=0}^{J-1}\frac{h_j^\alpha}{k_j^{\alpha/2}}\right)^2.$$

The estimates follow directly from the following estimation

$$\sum_{j=0}^{J-1}\frac{h_j^\alpha}{k_j^{\alpha/2}} \le C\frac{h_0^\alpha}{k_0^{\alpha/2}}\sum_{j=0}^{J-1}\left(\frac{4}{\beta}\right)^{\frac{j\alpha}{2}}.$$

This completes the proof of the theorem. □

What remains to be considered is the computational complexity. Assuming still that $k_j = \beta^j k_0$ for some fixed $\beta > 0$, we have the following corollary.

Corollary 3.1. *Let the number of smoothing steps on level j be given by $k_j = \beta^j k_0$, then the* *computational cost of the GCMG method is proportional to*

$$\sum_{j=1}^{J} k_j n_j \le \begin{cases} C\frac{1}{1-\beta/2^d} k_0 n_0, & \text{if } \beta < 2^d, \\ CJk_0 n_0, & \text{if } \beta = 2^d, \end{cases}$$

where d denotes the dimension and C denotes a constant that does not depend on the mesh size.

Proof. The result follows naturally from noting that $2^{dj}/c \le n_j \le c2^{dj}$ and observing that

$$\sum_{j=1}^{J} k_j n_j \le c k_0 n_0 \sum_{j=0}^{J-1} \big(\frac{\beta}{2^d}\big)^j$$

We see that if we set β to be $4 < \beta < 2^d$, our results regarding accuracy and complexity do not contradict. Therefore, we see that for $d = 3$ our algorithm is optimal, and is sub-optimal for $d = 2$.

4. Numerical Results

We now perform numerical tests on a variety of different graphs (listed in Table 4.1), taken from the University of Florida Sparse Matrix Collection [23]. All of our computations were performed on a MacBook Pro PC with a 2.9 GHz Intel Core i7 Processor with 8 GB RAM. All the algorithms are implemented in the FiedComp package[1)], written in MATLAB. We consider the performance of our eigensolver against the Locally Optimal Preconditioned Conjugate Gradient Method (LOPCG), with Lean Algebraic Multigrid (LAMG) as a preconditioner. The LOPCG Method is part of the MATLAB BLOPEX Package by Knyazev, and is described in [6, Algorithm 5.1]. The LAMG preconditioner is a MATLAB package by Livine, and is described in Livine and Brandt's paper [8]. We use a residual tolerance of .05 for the LOPCG, and an .1 tolerance for the LAMG preconditioner. For our Cascadic Eigensolver, we use the tolerance $(u^k, u^{k-1}) > 1 - 10^{-8}$. In Table 4.1 we report the run times in seconds for each graph, along with a measure of the error in the approximate eigenvector, given by $\|(L - \tilde{r}I)\tilde{y}\|$, where $\tilde{y}$ is the approximate eigenvector, and $\tilde{r}$ is the corresponding approximate eigenvalue. Note that our eigensolver consistently outperforms the Locally Optimal Preconditioned Conjugate Gradient Method with LAMG as a preconditioner.

We consider the number of steps of power iteration that we typically require for a given graph. We use the two dimensional Laplacian with $N = 10^3$ as an example. We implement our eigensolver, with our given tolerance and report the number of subgraphs we have, the size of each subgraph, and the number of iterations required on each level in Table 4.2.

We note that we observe a similar smoothing structure on each level to the condition $k_j = \beta^j k_0$ we assumed for the proof of Theorem 3.1. Also, we note that the coarsening appears to occur at roughly the same rate on each level, suggesting that although our heavy edge coarsening algorithm is random in nature, it typically maintains similar coarsening rates for a given graph structure. These two observations help give numerical evidence that the theoretical results from Section 3 are robust to general graphs with heavy edge coarsening as the restriction operator.

Finally, we give an example of the application of the GCMG algorithm to the two-dimensional Laplacian, to give some numerical results to support the theoretical bounds we obtained in Section 3. We choose $N = 1025$ and take $\beta = 4$, $k_0 = 1$. We give the error with respect to the difference in eigenvalue, taking $\tilde{r} = \tilde{y}^T L\tilde{y}$. Our results are given in Table 4.3.

[1)] `http://www.personal.psu.edu/jcu5018`

Table 4.1: Numerical Tests.

			LOPCG w/ LAMG		CMG Eigensolver	
Graph	Vertices	Edges	Run Time	Error	Run Time	Error
144	144649	1074393	6.254	5.0e-02	1.911	6.9e-03
598a	110971	741934	4.884	1.6e-02	1.414	6.8e-03
auto	448695	3314611	13.34	3.9e-02	6.050	9.2e-03
brack2	62631	366559	2.726	8.7e-03	0.780	8.3e-03
cs4	22499	87716	1.194	2.0e-02	0.271	1.1e-03
cti	16840	96464	1.236	4.5e-02	0.283	1.7e-03
delaunayn15	32768	196548	1.614	9.3e-03	0.389	4.8e-03
m14b	214765	3358036	9.210	2.4e-02	2.848	1.0e-02
PGPgiantc.	10680	48680	0.873	2.4e-02	0.201	5.8e-02
wing	62032	243088	2.232	1.2e-02	0.741	1.1e-03

Table 4.2: Graph Size and Number of Smoothing Steps by Level for 2D Laplacian, $N = 100$.

i	0	1	2	3	4	5	6
n_i	10000	3653	1195	384	137	46	14
k_i	3	7	10	17	26	44	-

Table 4.3: Errors on Sublevels for GCMG for 2D Laplacian, $N = 1025$.

i	0	1	2	3	4
$\lvert\lambda_i^{(2)} - \tilde{r}_0\rvert$	3.0369e-09	1.1189e-08	4.0447e-08	1.8336e-07	1.8068e-06
$\lvert\lambda_i^{(2)} - \tilde{r}_{k_i}\rvert$	3.0198e-09	1.0826e-08	3.4420e-08	8.1745e-08	8.2292e-08

5. Conclusion

In this paper, we have presented a fast algorithm for approximately computing the Fiedler vector of a graph Laplacian. We introduced a new coarsening procedure, called heavy edge coarsening. We note the speed with which the procedure coarsens, and the quality of coarse level graphs. The main contribution to the speed of the algorithm was a result of the implementation of the heavy edge coarsening procedure.

In addition to being a fast coarsening procedure, the heavy edge coarsening algorithm is also easier to implement than other techniques of a similar type, such as heavy edge matching and its variants (HEM and HEM*) [3, 4]. As a purely algebraic eigensolver, the combination of heavy edge coarsening and power iteration in a cascadic multigrid method provide a fast algorithm for finding the Fiedler vector of graph Laplacians. Numerical results show that our eigensolver is efficient and robust for different graphs.

Similar to the AMG method, the algebraic CMG eigensolver is difficult to analyze. Therefore, under a standard geometric setting, we consider the GCMG eigensolver and show that our cascadic eigensolver with power iteration as a smoother to be uniformly convergent for an elliptic eigenvalue problem discretized by standard linear finite element methods. In the three-dimensional case, it is optimal in terms of accuracy and computational complexity.

We believe that in future work convergence for the cascadic multigrid eigensolver could be shown in more general settings. In addition, the use of the heavy edge coarsening procedure for non-spectral methods is another avenue of research that could be explored in the future.

Acknowledgments. The research of Jinchao Xu is partially supported by NSF DMS-1217142 and NSFC 41230210. The research of Ludmil Zikatanov was supported in part by NSF DMS-1217142 and NSF DMS-1418843.

References

[1] Y. Koren, L. Carmel and D. Harel, Drawing huge graphs by algebraic multigrid optimization, *Multiscale Model. Simul.*, **1**:4 (2003), 645–673 (electronic).

[2] S.T. Barnard and H.D. Simon, Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, *Concurrency: Practice and Experience*, **6**:2 (1994), 101–117.

[3] G. Karypis and V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM J. Sci. Comput.*, **20**:1 (1998), 359–392 (electronic).

[4] G. Karypis and V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, *Journal of Parallel and Distributed Computing*, **48**:1 (1998), 96 – 129.

[5] G.L.G. Sleijpen and H.A.Van der Vorst, A Jacobi-Davidson iteration method for linear eigenvalue problems, *SIAM Rev.*, **42**:2 (2000), 267–293 (electronic).

[6] A.V. Knyazev, Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method, *SIAM J. Sci. Comput.*, **23**:2 (2001), 517–541 (electronic).

[7] I. Lashuk, M. Argentati, E. Ovtchinnikov and A. Knyazev, Preconditioned eigensolver LOBPCG in hypre and PETSc, Domain decomposition methods in science and engineering XVI, volume 55 of *Lect. Notes Comput. Sci. Eng.*, pages 635–642, Springer, Berlin, 2007.

[8] O.E. Livne and A. Brandt, Lean algebraic multigrid (LAMG): fast graph Laplacian linear solver, *SIAM J. Sci. Comput.*, **34**:4 (2012), B499–B522.

[9] F.A. Bornemann and P. Deuflhard, The cascadic multigrid method for elliptic problems, *Numer. Math.*, **75**:2 (1996), 135–152.

[10] F.A. Bornemann and P. Deuflhard, Cascadic multigrid methods, Domain decomposition methods in sciences and engineering (Beijing, 1995), pages 205–212, Wiley, Chichester, 1997.

[11] D. Braess, P. Deuflhard and K. Lipnikov, A subspace cascadic multigrid method for mortar elements, *Computing*, **69**:3 (2002), 205–225.

[12] V. Shaĭdurov, The convergence of the cascadic conjugate-gradient method under a deficient regularity, Problems and methods in mathematical physics (Chemnitz, 1993), volume 134 of *Teubner-Texte Math.*, pages 185–194, Teubner, Stuttgart, 1994.

[13] V.V. Shaidurov, Cascadic algorithm with nested subspaces in domains with curvilinear boundary, Advanced mathematics: computations and applications (Novosibirsk, 1995), pages 588–595, NCC Publ., Novosibirsk, 1995.

[14] V.V. Shaidurov, Some estimates of the rate of convergence for the cascadic conjugate-gradient method, *Comput. Math. Appl.*, **31**:4-5 (1996), 161–171, Selected topics in numerical methods (Miskolc, 1994).

[15] M. Fiedler, Algebraic connectivity of graphs, *Czechoslovak Math. J.*, **23(98)** (1973), 298–305.

[16] M. Fiedler, A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, *Czechoslovak Math. J.*, **25(100)**:4 (1975), 619–633.

[17] H. Kim, J. Xu and L. Zikatanov, A multigrid method based on graph matching for convection-diffusion equations, *Numer. Linear Algebra Appl.*, **10**:1-2 (2003), 181–195, Dedicated to the 60th birthday of Raytcho Lazarov.

[18] G.H. Golub and C.F. Van Loan, Matrix computations, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[19] I. Babuška and J.E. Osborn, Finite element-Galerkin approximation of the eigenvalues and eigenvectors of selfadjoint problems, *Math. Comp.*, **52**:186 (1989), 275–297.

[20] D. Gallistl, Adaptive Finite Element Computation of Eigenvalues, PhD thesis, der Humboldt-Universität zu Berlin, 2014.

[21] D. Boffi, Finite element approximation of eigenvalue problems, *Acta Numerica*, **19** (2010), 1–120.

[22] J. Berg and J. Lofstrom, Interpolation Spaces. An Introduction, Springer, 1976.

[23] T.A. Davis and Y. Hu, The University of Florida sparse matrix collection, *ACM Trans. Math. Software*, **38**:1 (2011), Art. 1, 25.