SIAM J. SCI. COMPUT.
Vol. 43, No. 5, pp. S727–S742

A POSTERIORI ERROR ESTIMATES FOR MULTILEVEL METHODS FOR GRAPH LAPLACIANS*

XIAOZHE HU†, KAIYI WU†, AND LUDMIL T. ZIKATANOV‡

Abstract. In this paper, we study a posteriori error estimators which aid multilevel iterative solvers for linear systems of graph Laplacians. In earlier works such estimates were computed by solving a perturbed global optimization problem, which could be computationally expensive. We propose a novel strategy to compute these estimates by constructing a Helmholtz decomposition on the graph based on a spanning tree and the corresponding cycle space. To compute the error estimator, we solve a linear system efficiently on the spanning tree and then a least-squares problem on the cycle space. As we show, such an estimator has a nearly linear computational complexity for sparse graphs under certain assumptions. Numerical experiments are presented to demonstrate the efficacy of the proposed method.

Key words. graph Laplacian, a posteriori error estimates, cycle space, spanning tree, Helmholtz decomposition

AMS subject classifications. 65N55, 65F10

DOI. 10.1137/20M1349618

1. Introduction.

Graphs are frequently employed to model networks in social science, energy, and biological applications [6, 18, 29]. In many cases, these applications require the solution of large-scale linear systems of equations given by the graph Laplacian matrix [20, 38, 43, 54, 61]. Such linear systems have also been the key to developing link-based ranking algorithms for web-searching queries [39, 50, 60] or recommendation systems [25, 41]. Exploiting the properties of graph Laplacians to achieve dimension reduction in high-dimensional data representation, researchers have produced fruitful results in image classification [26, 65], representation learning [27], and clustering [3, 47]. In the numerical solutions of PDEs, the stiffness matrices arising from the finite-element or finite-difference method also take the form of graph Laplacians as discussed in [64]. Therefore, it is important to develop efficient and robust methods for solving graph Laplacian systems.

To solve large-scale graph Laplacian linear systems, direct methods suffer from their expensive computational costs [33]. Iterative methods, such as the algebraic multigrid (AMG) methods originated in [9], are often applied to solve the linear systems (see also [63] and the references therein for a recent survey on the AMG methods). In practice, the AMG method achieves optimal computational complexity for many applications, including solving the linear systems with weighted graph Laplacians [5, 11, 22, 33, 35, 40, 46].

A typical multigrid method traverses a hierarchy of spaces (grids) of different dimensions (grid sizes) and solves the corresponding linear system at different resolutions.

*Received by the editors July 1, 2020; accepted for publication (in revised form) June 4, 2021; published electronically September 7, 2021.
https://doi.org/10.1137/20M1349618
Funding: The work of the first and second authors was partially supported by the National Science Foundation (NSF) under grant DMS-1812503 and CCF-1934553. The work of the third author was partially supported by NSF grants DMS-1720114 and DMS-1819157.
†Department of Mathematics, Tufts University, Medford, MA 02155 USA (Xiaozhe.Hu@tufts.edu, kaiyiwu0124@gmail.com).
‡Department of Mathematics, The Pennsylvania State University, Universtiy Park, PA 16802 USA (ludmil@psu.edu).

As is well known, an efficient AMG method should damp the algebraic high-frequency error using relaxations/smoothers (e.g., common iterative solvers like Jacobi and Gauss–Seidel methods) and eliminate the algebraically smooth (low-frequency) error via a coarse space (grid) correction [17]. The latter requires the "smooth" error obtained on the fine grids to be accurately approximated on the coarse spaces. Many different coarsening strategies have been developed based on good estimations of the error, for example, the classical AMG [9, 14], the smoothed aggregation AMG [13, 15, 16], the bootstrap AMG [7, 8], and the unsmoothed aggregation AMG [10, 11, 33, 40, 45, 49, 57]. Thus, an efficient, reliable, and computable estimate of the error a posteriori, during AMG iterations, is needed for developing robust AMG methods.

Generally, the a posteriori estimators provide a computable estimation for the true error locally. Our approach borrows several ideas from the finite-element literature (equilibrated error estimators [1, 2, 31, 58] and functional a posteriori error estimators [24, 48, 52, 56]). In [64], the authors derived a posteriori error estimator for solving graph Laplacian linear systems for the first time based on the functional a posteriori error estimation framework. Such a technique was used to predict the error of approximation from coarse grids for the multilevel unsmoothed aggregation AMG, and the estimator is computed by solving a perturbed global optimization problem (discussed in more details in section 2). Such an approach provides an accurate error estimator. However, it could be computationally expensive, which affects the effectiveness of the resulting adaptive AMG method. In this work, we propose a novel a posteriori error estimator and an efficient algorithm to reduce the computational cost, which could be used further to construct the efficient multilevel hierarchy for adaptive AMG. Roughly speaking, this is achieved by taking advantage of the Helmholtz decomposition [34] on the graph computationally, which splits the error into a divergence-free component and a curl-free component. The rationale of the proposed algorithm for computing the approximation to the true error has two main steps.

1. Solve a linear system on a spanning tree (defined in section 2) of the graph to get the curl-free component, or equivalently, the gradient component of the error.
2. Approximately solve a minimization problem in the cycle space to obtain the divergence-free component of the Helmholtz decomposition of the error.

The first step can be done in linear time with Gaussian elimination using lexicographical ordering as shown in [53, 59]. Solving exactly the minimization in the cycle space of the graph is computationally expensive as it is equivalent to solving a constrained minimization problem, which is as difficult as the original linear system (often even more difficult). Our algorithm solves the minimization approximately by applying several steps of a relaxation scheme, the one-level Schwarz method [19, 55]. This crucial improvement reduces the computational cost and gives an accurate a posteriori estimate of the true error in nearly optimal time for sparse graphs, which is further verified by our numerical experiments. Clearly, such an error estimator can be incorporated to construct multilevel hierarchies since it provides accurate estimate of the true error, which is important for constructing coarse levels adaptively. The corresponding adaptive coarsening scheme will preserve the smooth error accurately on the coarse levels and ensure the robustness of the resulting adaptive AMG method.

The rest of this paper is organized as follows. In section 2 we review backgrounds on graphs and graph Laplacians, along with some previous results in [64]. The main algorithm to compute a posteriori error estimates is stated in section 3. We present

and analyze some numerical experiments in section 4. Finally, in section 5 we summarize the main contribution and list some future work.

2. Preliminaries. In this section, we define the necessary notations and recall some fundamental results for the computation of an a posteriori error estimator for solving graph Laplacians as presented in [64].

2.1. Graphs and graph Laplacians. Consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \omega)$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the vertex set, $\mathcal{E} = \{\{i, j\},\ i, j \in \mathcal{V}, i > j\}$ is the edge set, and $\omega = \{\omega_e\}_{e\in\mathcal{E}}$ is the set of edge weights. Here, the weights are assumed to be positive, i.e., $\omega_e > 0$, and we take all the edge weights to be 1 for unweighted graphs. We only consider undirected graphs here, and that is why we have fixed $i > j$ for every $e = \{i, j\} \in \mathcal{E}$. Thus $\{1, 2\}$ cannot be an edge in our graph, while $\{2, 1\}$ could be in $\mathcal{E}$.

Denote $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. Let $\mathscr{V} = \mathbb{R}^n$ and $\mathscr{W} = \mathbb{R}^m$ be the vertex space and edge space, respectively. The inner product on vertex space and edge space are defined as

$$(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{v}^\intercal \boldsymbol{u} \quad \forall\ \boldsymbol{u}, \boldsymbol{v} \in \mathscr{V},$$
$$(\boldsymbol{\tau}, \boldsymbol{\phi}) = \boldsymbol{\phi}^\intercal \boldsymbol{\tau} \quad \forall\ \boldsymbol{\tau}, \boldsymbol{\phi} \in \mathscr{W}.$$

The weighted graph Laplacian matrix $L \in \mathbb{R}^{n\times n}$ can be defined via the bilinear form

$$(L\boldsymbol{u}, \boldsymbol{v}) := \boldsymbol{v}^\intercal L \boldsymbol{u} = \sum_{e=\{i,j\}\in\mathcal{E}} \omega_e (\boldsymbol{u}_i - \boldsymbol{u}_j)(\boldsymbol{v}_i - \boldsymbol{v}_j), \quad \forall\ \boldsymbol{u}, \boldsymbol{v} \in \mathscr{V}.$$

Associated with the graph is the discrete gradient operator (or edge-node incidence matrix) $G \in \mathbb{R}^{m\times n} : \mathscr{V} \to \mathscr{W}$ and the edge weight matrix $D \in \mathbb{R}^{m\times m} : \mathscr{W} \to \mathscr{W}$. They are defined as the following: for each edge $e = \{i, j\} \in \mathcal{E}$,

$$(G\boldsymbol{v})_e = \boldsymbol{v}_i - \boldsymbol{v}_j \quad \forall\ \boldsymbol{v} \in \mathscr{V},$$
$$(D\boldsymbol{\tau})_e = w_e \boldsymbol{\tau}_e \quad \forall\ \boldsymbol{\tau} \in \mathscr{W}. \tag{2.1}$$

The adjoint of G, denoted by $G^\intercal : \mathscr{W} \to \mathscr{V}$, is the divergence operator (represented by the node-edge incidence matrix) on the graph

$$(G\boldsymbol{u}, \boldsymbol{\tau}) = (\boldsymbol{u}, G^\intercal \boldsymbol{\tau}) \quad \forall\ \boldsymbol{u} \in \mathscr{V},\ \forall\ \boldsymbol{\tau} \in \mathscr{W}. \tag{2.2}$$

By direct computation, the following identity holds true:

$$(L\boldsymbol{u}, \boldsymbol{v}) = (DG\boldsymbol{u}, G\boldsymbol{v}).$$

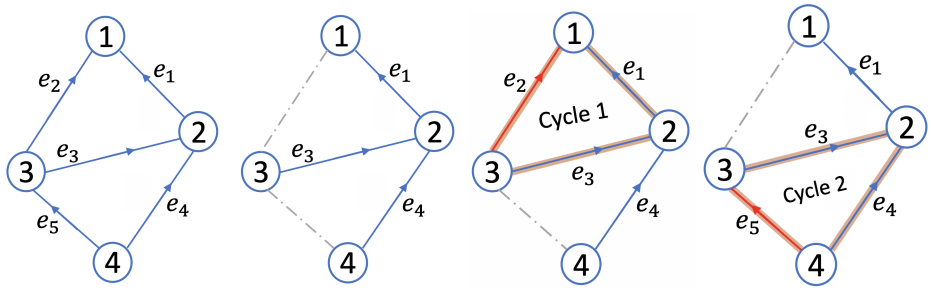
Thus, we can write $L := G^\intercal D G$. Based on this definition of the graph Laplacian L, it is straightforward to verify that,

$$\|DG\boldsymbol{u} - DG\boldsymbol{v}\|^2_{D^{-1}} = \|\boldsymbol{u} - \boldsymbol{v}\|^2_L \quad \forall\ \boldsymbol{u}, \boldsymbol{v} \in \mathscr{V},$$

where $\|\boldsymbol{\tau}\|^2_{D^{-1}} = (\boldsymbol{\tau}, \boldsymbol{\tau})_{D^{-1}} := (D^{-1}\boldsymbol{\tau}, \boldsymbol{\tau})\ \forall\ \boldsymbol{\tau} \in \mathscr{W}$, and $\|\boldsymbol{v}\|^2_L = (\boldsymbol{v}, \boldsymbol{v})_L := (L\boldsymbol{v}, \boldsymbol{v})$ $\forall\ \boldsymbol{v} \in \mathscr{V}$.

In addition to the vertex space $\mathscr{V}$ and edge space $\mathscr{W}$, another important space of a graph $\mathcal{G}$ is the so-called *cycle space* (see [4] for more details), denoted by $\mathscr{C}$, which is defined as

$$\mathscr{C} := \{\boldsymbol{c} \in \mathscr{W} \mid G^\intercal \boldsymbol{c} = \boldsymbol{0}\}. \tag{2.3}$$



(a) original graph (b) spanning tree (c) adding edge e_2 to get cycle 1. (d) add edge e_5 to get cycle 2.

FIG. 1. *Fundamental cycle basis.*

Each cycle on the graph $\mathcal{G}$ corresponds to an element $\boldsymbol{c}$ in the cycle space $\mathscr{C}$. To be more specific, if $\{i_1, i_2, \ldots, i_k, i_{k+1} = i_1\}$ is a cycle $\mathcal{G}$, that is,

$$i_j \in \mathcal{V}, \quad \text{and} \quad \{\max\{i_j, i_{j+1}\}, \min\{i_j, i_{j+1}\}\} \in \mathcal{E}, \quad j = 1, \ldots, k,$$

we define the components of $\boldsymbol{c} \in \mathscr{C}$ as

$$(\boldsymbol{c})_e = \operatorname{sign}(i_j - i_{j+1}), \quad \text{if} \quad e = \{\max\{i_j, i_{j+1}\}, \min\{i_j, i_{j+1}\}\},$$

with $(\boldsymbol{c})_e$ extended as zero for all edges that are not on the cycle. Besides its definition (2.3), we can also characterize the cycle space $\mathscr{C}$ by its basis. As discussed in [30], the cycle space of a connected simple graph has dimension $m - n + 1$ (by simple graphs we mean the graphs that contains neither self-loops nor duplicate edges). Note there is more than one way to find the basis of the cycle space (see the survey paper [30]). For general graphs, a commonly used set of basis for the cycle space is the *basis of fundamental cycles*. Such basis is not unique, but each such basis is induced by a spanning tree. In a connected graph $\mathcal{G}$ with n vertices, a spanning tree is any subgraph of $\mathcal{G}$ that connects all the vertices and has no cycles or, equivalently, has exactly $(n-1)$ edges. To construct the basis of fundamental cycles corresponding to a spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \omega_\mathcal{T})$ of a graph $\mathcal{G}$, we proceed as follows. For each edge that does not belong to the tree, i.e., $e = \{i, j\} \in \mathcal{E} \setminus \mathcal{E}_\mathcal{T}$, we can find a cycle $\{i, j\} \cup p(i, j)$ where $p(i, j)$ is the path from vertex i to vertex j on the tree $\mathcal{T}$. Since the spanning tree $\mathcal{T}$ has exactly $(n-1)$ edges, there are $(m - n + 1)$ such cycles. It can be shown that they are linearly independent [30] and, therefore, form a basis for the cycle space $\mathscr{C}$.

In Figure 1, we give a simple example of the fundamental cycle basis. The tree in Figure 1(b) is a spanning tree of the graph in Figure 1(a). First, the edge e_2 is added back (see Figure 1(c)) which results in the first cycle $\{2, 1, 3, 2\}$ consisting of edges e_1, e_2, and e_3. The vector representation of the cycle induced by adding back edge e_2 is given by $\boldsymbol{c}^{e_2} = [1, -1, 1, 0, 0]^\intercal$. Similarly, by adding edge e_5 back, we have the second cycle $\{2, 3, 4, 2\}$ formed by edges e_3, e_4, and e_5, which is represented by $\boldsymbol{c}^{e_5} = [0, 0, -1, -1, 1]^\intercal$. $\boldsymbol{c}^{e_2}$ and $\boldsymbol{c}^{e_5}$ form a cycle basis.

2.2. Previous results on a posteriori error estimators. We are interested in solving the following linear system of graph Laplacians

$$L\boldsymbol{u} = \boldsymbol{f}, \tag{2.4}$$

by some iterative methods. Equation (2.4) simulates a rich spectrum of weighted graph Laplacians problems [29, 38, 43, 54]. For example, in computational physics, the solution $\boldsymbol{u} \in \mathbb{R}^n$ models the potentials of an electrical flow where the resistance of each edge on the graph (electrical circuit) is the reciprocal of the edge weight, and $\boldsymbol{f} = [f_1, f_2, \ldots, f_n]^\intercal \in \mathbb{R}^n$ denotes the current supplied to each node i. After k iterations we get an approximated solution $\boldsymbol{u}^k$. If we can somehow construct the current error $\boldsymbol{e}^k = \boldsymbol{u} - \boldsymbol{u}^k$, then the true solution will be easily obtained by $\boldsymbol{u} = \boldsymbol{u}^k + \boldsymbol{e}^k$. In practice the true error $\boldsymbol{e}^k$ is not computable because $\boldsymbol{u}$ is unknown, so alternatively we seek to find $\widetilde{\boldsymbol{e}}^k$, an accurate estimation of $\boldsymbol{e}^k$, and use $\widetilde{\boldsymbol{e}}^k$ to improve the current approximation. Furthermore, an accurate estimation of the error gives an insight into the performance of the iterative methods. For example, in AMG methods, such an estimation approximates the so-called smooth error, which is responsible for the slow convergence of the AMG methods, and can be used to improve the AMG algorithm adaptively. This leads to the adaptive AMG methods [12, 14, 22, 42, 44] which has been actively researched over the past two decades.

Since our a posteriori estimator is motivated by the a posteriori error estimator developed in [64], we recall the main results and algorithms presented in [64] and start with the following fundamental lemma which relates the error and computed approximate solution. The proof of the lemma, as stated here, is found in [64].

LEMMA 2.1. *Let $\boldsymbol{u}$ be the solution to* (2.4). *Then for arbitrary $\boldsymbol{\tau} \in \mathscr{W}$, the following inequality holds for all $\boldsymbol{v} \in \mathscr{V}$:*

$$\|\boldsymbol{u} - \boldsymbol{v}\|_L \leq \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}} + C_p^{-1}\|G^\intercal\boldsymbol{\tau} - \boldsymbol{f}\|_{\mathscr{V}}, \tag{2.5}$$

where C_p is Poincaré's constant of the graph Laplacian L.

For a fixed $\boldsymbol{v}$, denote the right-hand side of (2.5) by

$$\eta(\boldsymbol{\tau}) = \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}} + C_p^{-1}\|G^\intercal\boldsymbol{\tau} - \boldsymbol{f}\|_{\mathscr{V}}.$$

This naturally provides the a posteriori error estimator for estimating the error $\boldsymbol{u} - \boldsymbol{v}$ if $\boldsymbol{v}$ is an approximate solution; i.e., $\boldsymbol{v} = \boldsymbol{u}^k$. Moreover, by minimizing the right-hand side of (2.5) with respect to $\boldsymbol{\tau}$, we can obtain an accurate estimator. To solve the minimization problem efficiently, in [64], an upper bound $E(\beta, \boldsymbol{\tau})$ of $\eta(\boldsymbol{\tau})$ was introduced as follows:

$$\eta^2(\boldsymbol{\tau}) \leq E(\beta, \boldsymbol{\tau}),$$

where

$$E(\beta, \boldsymbol{\tau}) := (1+\beta)\|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2 + \left(1 + \frac{1}{\beta}\right) C_p^{-1}\|G^\intercal\boldsymbol{\tau} - \boldsymbol{f}\|_{\mathscr{V}}^2.$$

An accurate estimator can be obtained by computing $\min_{\beta,\boldsymbol{\tau}} E(\beta, \boldsymbol{\tau})$. In [64], an alternating process is applied to minimize $E(\beta, \boldsymbol{\tau})$ with respect to β (with the techniques proposed in [36]) and $\boldsymbol{\tau}$ iteratively, as summarized in Algorithm 2.1 (see [64] for details).

Although the approach developed in [64] provides a reliable error estimator, the corresponding computational cost might be expensive due to the iterative minimization of $E(\beta, \boldsymbol{\tau})$ in step 3 and 4 in Algorithm 2.1. One iteration of step 4 can be as difficult as solving the original system, which makes this approach expensive computationally. In order to improve the accuracy of the a posteriori error estimator,

Algorithm 2.1 Alternating Process for Solving $\min_{\beta,\boldsymbol{\tau}} E(\beta,\boldsymbol{\tau})$.

```
procedure [β, τ] =MinimizeBound(β^0, τ^0)
    for k = 1, 2, ..., max_iter do
        compute τ^k = argmin_τ E(β^{k-1}, τ).
        compute β^k = argmin_β E(β, τ^k).
    end for
end procedure
```

and, more importantly, to improve the efficiency of computing it, we develop a novel technique for estimating the error based on (2.5), which we present next.

3. Efficient algorithm for computing a posteriori error estimator. As we have pointed out, the derivation of the a posteriori error estimator is based on approximating the Helmholtz decomposition of the error. This provides a tighter error bound than the one proposed in [64] and can be implemented efficiently.

3.1. Hypercircle identity and error estimation on graphs. Our design of an a posteriori error estimator is motivated by (2.5). For a given $\boldsymbol{f} \in \mathscr{V}$, we define the space $\mathscr{W}(\boldsymbol{f}) = \{\boldsymbol{\tau} \in \mathscr{W} \mid G^\intercal \boldsymbol{\tau} = \boldsymbol{f}\}$. If we choose $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$, then the second term on the right-hand side of (2.5) vanishes and we only have the first term left. If we minimize this term with respect to $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$, we can immediately get an accurate estimation. We summarize this in the following theorem.

THEOREM 3.1. *Let $\boldsymbol{u}$ be the solution to* (2.4). *Then for any $\boldsymbol{v} \in \mathscr{V}$, we have*

$$\|\boldsymbol{u} - \boldsymbol{v}\|_L = \min_{\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})} \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}. \tag{3.1}$$

To prove Theorem 3.1, we will make use of the next lemma, also known as *hypercircle identity* (see [51]).

LEMMA 3.2. *Let $\boldsymbol{u}$ be the solution to* (2.4). *Then for any $\boldsymbol{v} \in \mathscr{V}$ and any $\boldsymbol{\tau} \in \mathscr{W}(f)$ the following identity holds:*

$$\|\boldsymbol{u} - \boldsymbol{v}\|_L^2 + \|DG\boldsymbol{u} - \boldsymbol{\tau}\|_{D^{-1}}^2 = \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2.$$

Now we are ready to prove Theorem 3.1.

Proof. (Theorem 3.1) We first show $\|\boldsymbol{u} - \boldsymbol{v}\|_L \le \min_{\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})} \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}$. It follows from Lemma 3.2 that

$$\|\boldsymbol{u} - \boldsymbol{v}\|_L^2 = \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2 - \|DG\boldsymbol{u} - \boldsymbol{\tau}\|_{D^{-1}}^2 \le \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2.$$

Since the inequality holds for any $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$, we have

$$\|\boldsymbol{u} - \boldsymbol{v}\|_L^2 \le \min_{\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})} \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2.$$

To show the other direction, note that

$$\begin{aligned}
\|\boldsymbol{u} - \boldsymbol{v}\|_L^2 &= (L(\boldsymbol{u} - \boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v}) = (G^\intercal DG(\boldsymbol{u} - \boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v}) = (DG(\boldsymbol{u} - \boldsymbol{v}), G(\boldsymbol{u} - \boldsymbol{v})) \\
&= (DG(\boldsymbol{u} - \boldsymbol{v}), DG(\boldsymbol{u} - \boldsymbol{v}))_{D^{-1}} = \|DG(\boldsymbol{u} - \boldsymbol{v})\|_{D^{-1}}^2 = \|DG\boldsymbol{v} - DG\boldsymbol{u}\|_{D^{-1}}^2 \\
&\ge \min_{\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})} \|DG\boldsymbol{v} - \boldsymbol{\tau}\|_{D^{-1}}^2.
\end{aligned}$$

This completes the proof. ∎

From Theorem 3.1, we observe that

$$\|\boldsymbol{u}-\boldsymbol{v}\|_L \le \|DG\boldsymbol{v}-\boldsymbol{\tau}\|_{D^{-1}}$$

for any $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$. This motivates us to define the following computable quantity:

$$\psi(\boldsymbol{\tau}) := \|DG\boldsymbol{v}-\boldsymbol{\tau}\|_{D^{-1}}, \quad \forall\, \boldsymbol{\tau} \in \mathscr{W}(f). \tag{3.2}$$

If $\boldsymbol{v}$ is the approximate solution to (2.4), $\psi(\boldsymbol{\tau})$ gives an a posteriori estimator for the true error $\boldsymbol{u}-\boldsymbol{v}$ for any choice of $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$. If $\boldsymbol{\tau}^*$ is the minimizer of the right-hand side of (3.1), then $\psi(\boldsymbol{\tau^*}) = \|\boldsymbol{u}-\boldsymbol{v}\|_L$. Of course, computing the minimizer $\boldsymbol{\tau}^*$ exactly would be expensive. As an alternative we propose a Schwarz method for approximating $\boldsymbol{\tau}^*$. As our numerical tests show, we can obtain a reasonably good approximation $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$, $\boldsymbol{\tau} \approx \boldsymbol{\tau}^*$.

3.2. Efficient evaluation of the error estimator. Our approach is to solve the minimization problem

$$\min_{\boldsymbol{\tau}\in\mathscr{W}(\boldsymbol{f})} \|DG\boldsymbol{v}-\boldsymbol{\tau}\|_{D^{-1}} \tag{3.3}$$

based on the Helmholtz decomposition of $\boldsymbol{\tau}$ on the graph, i.e., $\boldsymbol{\tau} = \boldsymbol{\tau}_f + \boldsymbol{\tau}_0^*$, where $\boldsymbol{\tau}_f \in \mathscr{W}(\boldsymbol{f})$ and $\boldsymbol{\tau}^* \in \mathscr{C}$ such that $(\boldsymbol{\tau}_f, \boldsymbol{\tau}_0^*) = 0$. Here $\boldsymbol{\tau}_f$ is curl-free and $\boldsymbol{\tau}_0^*$ is divergence-free. In particular, we first find a $\boldsymbol{\tau}_f \in \mathscr{W}(\boldsymbol{f})$ by solving a graph Laplacian on a spanning tree of the graph. Then for a given $\boldsymbol{\tau}_f \in \mathscr{W}(\boldsymbol{f})$, the minimization problem (3.3) becomes

$$\min_{\boldsymbol{\tau}_0\in\mathscr{C}} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f-\boldsymbol{\tau}_0\|_{D^{-1}}.$$

Solving this constrained minimization problem exactly will give the exact minimizer $\boldsymbol{\tau}_0^*$ and thus theoretically give the true error, which is an overkill in terms of finding the a posteriori error estimator. In practice, we only need to solve it approximately since as long as $\boldsymbol{\tau} \in \mathscr{W}(\boldsymbol{f})$, $\psi(\boldsymbol{\tau})$ will provide an upper bound of the error, which can be used as an error estimator. Note that this approximation is (always) subject to a trade-off: the error estimator will approximate the true error very accurately if we solve the optimization problem at unacceptable computation cost, or we can approximate and obtain a not-so-tight error estimator at an optimal computational cost.

3.2.1. Computing the curl-free component of the error. In this subsection, we discuss how to compute $\boldsymbol{\tau}_f \in \mathscr{W}(\boldsymbol{f})$ with optimal computational complexity for a given graph. For any $\boldsymbol{\tau}_f \in \mathscr{W}(\boldsymbol{f})$, we have

$$G^\intercal \boldsymbol{\tau}_f = \boldsymbol{f}. \tag{3.4}$$

Since $G^\intercal$ is the discrete divergence operator on the graph, the solution to the above equation is not unique and difficult to compute in general. However, we just need to find one $\boldsymbol{\tau}_f$. Here, based on a spanning tree $\mathcal{T}$ of the graph $\mathcal{G}$, we present an approach with optimal complexity, i.e., $\mathcal{O}(n)$ computational cost.

For a given spanning tree $\mathcal{T} = (\mathcal{V}, \mathcal{E}_\mathcal{T}, \omega_\mathcal{T})$, we look for a $\boldsymbol{\tau}_f$ satisfying (3.4) but only has nonzero entries on the edges that belong to the spanning tree $\mathcal{T}$. In this case, we can rewrite (3.4) as:

$$\boldsymbol{f} = G^\intercal \boldsymbol{\tau}_f = \begin{pmatrix} G_\mathcal{T}^\intercal & G_{\mathcal{G}\setminus\mathcal{T}}^\intercal \end{pmatrix} \begin{pmatrix} \boldsymbol{\tau}_{f,\mathcal{T}} \\ \mathbf{0} \end{pmatrix}, \tag{3.5}$$

where $G_{\mathcal{T}}^{\intercal}$ is the discrete divergence operator that acts on edges in the tree $\mathcal{T}$, and $G_{\mathcal{G}\setminus\mathcal{T}}^{\intercal}$ is the discrete divergence operator on edges that are only in the graph $\mathcal{G}$ but not in the tree $\mathcal{T}$. From (3.5), we have

$$G_{\mathcal{T}}^{\intercal}\boldsymbol{\tau}_{f,\mathcal{T}} = \boldsymbol{f}. \tag{3.6}$$

Therefore, once we solve (3.6), we can assemble $\boldsymbol{\tau}_f$ by adding back the edges that are in the graph $\mathcal{G}$ but not in the tree $\mathcal{T}$. Note that (3.6) is defined only on the spanning tree $\mathcal{T}$. We can first solve

$$L_{\mathcal{T}}\boldsymbol{x} = \boldsymbol{f}, \tag{3.7}$$

$L_{\mathcal{T}}$ being the graph Laplacian of the tree $\mathcal{T}$. With the fact that $L_{\mathcal{T}} = G_{\mathcal{T}}^{\intercal}D_{\mathcal{T}}G_{\mathcal{T}}$, where $D_{\mathcal{T}}$ and $G_{\mathcal{T}}$ are the discrete diagonal edge weight matrix and discrete gradient operator on tree $\mathcal{T}$, respectively, (3.7) can be rewritten as follows:

$$G_{\mathcal{T}}^{\intercal}D_{\mathcal{T}}G_{\mathcal{T}}\boldsymbol{x} = \boldsymbol{f}. \tag{3.8}$$

Comparing (3.6) and (3.8), we naturally have

$$\boldsymbol{\tau}_{f,\mathcal{T}} := D_{\mathcal{T}}G_{\mathcal{T}}\boldsymbol{x}$$

and can thereafter assemble the full $\boldsymbol{\tau}_f = (\boldsymbol{\tau}_{f,\mathcal{T}}, \mathbf{0})^{\intercal}$. The procedure for computing $\boldsymbol{\tau}_f$ is summarized in Algorithm 3.1.

Algorithm 3.1 Computation of $\boldsymbol{\tau}_f$.

1: **procedure** $[\boldsymbol{\tau}_f, \mathcal{T}]$ =COMPUTE$\boldsymbol{\tau}_f(\mathcal{G}, \boldsymbol{f})$
2: Build the spanning tree $\mathcal{T}$ from $\mathcal{G}$.
3: Solve $L_{\mathcal{T}}\boldsymbol{x} = \boldsymbol{f}$, where $L_{\mathcal{T}} = G_{\mathcal{T}}^{\intercal}D_{\mathcal{T}}G_{\mathcal{T}}$.
4: Compute $\boldsymbol{\tau}_{f,\mathcal{T}} \leftarrow D_{\mathcal{T}}G_{\mathcal{T}}\boldsymbol{x}$.
5: Assemble $\boldsymbol{\tau}_f$ as $\boldsymbol{\tau}_f \leftarrow \begin{pmatrix} \boldsymbol{\tau}_{f,\mathcal{T}} \\ \mathbf{0} \end{pmatrix}$.
6: **end procedure**

Identifying a spanning tree via the classic breadth-first search algorithm (BFS tree) takes $\mathcal{O}(m+n)$ computational complexity for general graphs [21]. If we consider sparse graphs in which $m = \mathcal{O}(n)$ or $\mathcal{O}(n \log n)$, this step costs at most $\mathcal{O}(n \log n)$. The main computational cost of Algorithm 3.1 comes from step 3, i.e., solving the linear system (3.7). As discussed in [53, 59], it takes linear time to solve (3.7). Additionally, the matrix-vector multiplication in step 4 has $\mathcal{O}(n)$ complexity for sparse graphs. Therefore, the overall complexity of Algorithm 3.1 is at most $\mathcal{O}(n \log n)$.

3.2.2. Computing the the divergence-free component of the error. For a given $\boldsymbol{\tau}_f$, we need to solve the following constrained minimization problem:

$$\boldsymbol{\tau}_0^* = \operatorname*{argmin}_{\boldsymbol{\tau}_0 \in \mathscr{C}} \|DG\boldsymbol{v} - \boldsymbol{\tau}_f - \boldsymbol{\tau}_0\|_{D^{-1}}. \tag{3.9}$$

The difficulty here is that we need to satisfy the constraint exactly when we compute an approximate $\boldsymbol{\tau}_0^*$ to get the estimated value of the error. Our approach is to explicitly build the $(m-n+1)$ basis $\{\boldsymbol{c}^e\}$ of the cycle space $\mathscr{C}$ as discussed in section 2 and transform the constrained minimization problem (3.9) into a unconstrained minimization problem. Denote by Ω the index set of the cycle basis. Then for any $\boldsymbol{\tau}_0 \in \mathscr{C}$ we can write $\boldsymbol{\tau}_0$ as a linear combination of the cycle basis: $\boldsymbol{\tau}_0 = \sum_{e\in\Omega} \alpha_e \boldsymbol{c}^e$. Denote $\boldsymbol{\alpha} \in \mathbb{R}^{m-n+1}$, $(\boldsymbol{\alpha})_e = \alpha_e$. Thus, the minimization problem (3.9) becomes

$$\min_{\boldsymbol{\tau}_0\in\mathscr{C}} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f-\boldsymbol{\tau}_0\|_{D^{-1}} = \min_{\boldsymbol{\alpha}} \psi(\boldsymbol{\alpha}) = \min_{\alpha_e,\ e\in\Omega} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f-\sum_{e\in\Omega}\alpha_e\boldsymbol{c}^e\|_{D^{-1}}. \tag{3.10}$$

This is an unconstrained least-squares problem, and we can solve it with usual approaches. Moreover, the approximate solution is guaranteed to belong to the cycle space. Solving (3.10) exactly will eventually give us the exact error $\|\boldsymbol{u}-\boldsymbol{v}\|_L$. This step, however, has a computational complexity comparable to solving the original problem (2.4). Therefore, we solve it approximately via a few steps of the one-level overlapping Schwarz method (see [55, 19]). To describe such a method, we first decompose the cycle space $\mathscr{C}$ into the following subspaces:

$$\mathscr{C} = \mathscr{C}_1 + \mathscr{C}_2 + \cdots + \mathscr{C}_J. \tag{3.11}$$

Note that $\mathscr{C}_i \cap \mathscr{C}_j$ is not necessarily empty. Then, each step of the Schwarz method corresponds to a loop over all subspaces and solves the minimization problem (3.10) in each of the subspace $\mathscr{C}_i$. That is, for $i = 1, 2, \ldots, J$, compute

$$\min_{\Delta\boldsymbol{\tau}\in\mathscr{C}_{i+1}} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f-(\boldsymbol{\tau}_0^i+\Delta\boldsymbol{\tau})\|_{D^{-1}}, \tag{3.12}$$

where $\boldsymbol{\tau}_0^i$ is the approximation to $\boldsymbol{\tau}_0^*$ after solving (3.12) in the first i subspaces, and $\boldsymbol{\tau}_0^* = \operatorname{argmin}_{\boldsymbol{\tau}_0\in\mathscr{C}} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f-\boldsymbol{\tau}_0\|_{D^{-1}}$. The step of this relaxation method is completed after the approximation in $\mathscr{C}_J$ is computed. The solution obtained by iteratively solving (3.12) converges to the solution of the original minimization problem (3.9) since this can be interpreted as a subspace optimization method whose convergence property was discussed in [62].

To keep low computational cost in computing the error estimator, we only run $\mathcal{O}(1)$ iterations of the Schwarz method to approximately solve (3.12) for $\boldsymbol{\tau}_0^*$. Later in section 4 we show that the error estimator computed with such an approximation is indeed accurate enough to capture the true error. The steps to compute $\boldsymbol{\tau}_0^* \in \mathscr{C}$ approximately are summarized in Algorithm 3.2, and we denote the approximation of $\boldsymbol{\tau}_0^*$ by $\boldsymbol{\tau}_0$.

Algorithm 3.2 Computing an Approximation to $\boldsymbol{\tau}_0^*$.

procedure $\boldsymbol{\tau}_0$ =COMPUTE$\boldsymbol{\tau}_0(\mathcal{G}, \mathcal{T}, \boldsymbol{\tau}_f)$
  Build the cycle basis $\{\boldsymbol{c}^e\}$
  Given initial guess $\boldsymbol{\tau}_0^0 = \mathbf{0}$,
  for $i = 1, 2, \ldots,$ max_iter **do**
    for $k = 1, 2, \ldots, J$ **do** ▷ iterate over each subdomain
      $\Delta\boldsymbol{\tau}^* = \operatorname{argmin}_{\boldsymbol{\eta}\in\mathscr{C}_k} \|DG\boldsymbol{v}-\boldsymbol{\tau}_f+\boldsymbol{\tau}_0^{(i-1)J+k-1}+\boldsymbol{\eta}\|_{D^{-1}}$.
      $\boldsymbol{\tau}_0^{(i-1)J+k} = \boldsymbol{\tau}_0^{(i-1)J+k-1} + \Delta\boldsymbol{\tau}$.
    end for
  end for
  return $\boldsymbol{\tau}_0^{J\cdot\text{max\_iter}}$.
end procedure

In Algorithm 3.2, the cost of one step of the Schwarz method depends on the number of subspaces J and the cost of solving (3.12) in each subspace. Here, we choose the following overlapping subspace decomposition: the i-th subspace is the span of the basis for the cycles incident with the vertex i. Thus, we have $J = n$, and

$$\mathscr{C}_i = \operatorname{span}\{\boldsymbol{c}^j |\ \text{cycle } j \text{ incidents with vertex } i\}, \quad i = 1, \ldots, J. \tag{3.13}$$

Since there are n vertices we have $J = n$ subspaces. For sparse graphs using the special data structure proposed in [32], the solution of the minimization problem (3.12) on each subspace $\mathscr{C}_i$ will be obtained with computational complexity at most $\mathcal{O}(\log n)$. This holds even in the rare cases when the dimension of $\mathscr{C}_i$ is large (e.g. $\sim n$). As a consequence the overall computational cost of each iteration of the Schwarz method is $\mathcal{O}(n \log n)$ for this choice of subspace decomposition (3.13), and this assures a low computational cost in computing the proposed estimator.

3.2.3. An algorithm for a posteriori error estimation with Helmholtz decomposition. Now we are ready to present the overall Algorithm 3.3 to (approximately) solve the minimization problem (3.10) and compute a posteriori error estimation for solving the graph Laplacian (2.4).

Algorithm 3.3 Computation of the Error Estimator $\min_{\boldsymbol{\tau}_0 \in \mathscr{C}} \|DG\boldsymbol{v} - \boldsymbol{\tau}_f - \boldsymbol{\tau}_0\|_{D^{-1}}$.

1: **procedure** ψ =ErrorEstimates($\mathcal{G}, \boldsymbol{v}, \boldsymbol{f}$)
2: $[\boldsymbol{\tau}_f, \mathcal{T}]$ = Compute$\boldsymbol{\tau}_f(\mathcal{G}, \boldsymbol{f})$.
3: $\boldsymbol{\tau}_0$ = Compute$\boldsymbol{\tau}_0(\mathcal{G}, \mathcal{T}, \boldsymbol{\tau}_f)$.
4: $\psi \leftarrow \|DG\boldsymbol{v} - \boldsymbol{\tau}_f - \boldsymbol{\tau}_0\|_{D^{-1}}$. ▷ Compute the value of the estimator
5: **return** ψ.
6: **end procedure**

In Algorithm 3.3, step 2 to compute $\boldsymbol{\tau}_f$ takes $\mathcal{O}(n \log n)$ for any sparse graphs. Step 3 to compute $\boldsymbol{\tau}_0$ has complexity $\mathcal{O}(n \log n)$ for sparse graphs since the minimization problem (3.10) is solved approximately with $\mathcal{O}(1)$ iterations of the Schwarz method. As a result, the overall computational complexity of Algorithm 3.3 is $\mathcal{O}(n \log n)$ for sparse graph $\mathcal{G}$.

To make the a posteriori error estimator more useful, especially for developing the adaptive AMG methods for solving graph Laplacians [64, 37, 40], we need to localize the a posteriori error estimator. Since

$$\begin{aligned}\psi^2(\boldsymbol{\tau}) = \|DG\boldsymbol{v} - \boldsymbol{\tau}\|^2_{D^{-1}} &= (DG\boldsymbol{v} - \boldsymbol{\tau})^T D^{-1}(DG\boldsymbol{v} - \boldsymbol{\tau}) \\ &= \sum_{e \in \mathcal{E}} \frac{1}{\omega_e}\big((DG\boldsymbol{v} - \boldsymbol{\tau})e\big)^2,\end{aligned}$$

we can localize the error estimator on each edge e as follows:

$$\psi_e(\boldsymbol{\tau}) = \boldsymbol{\omega}_e^{-\frac{1}{2}} |(DG\boldsymbol{v} - \boldsymbol{\tau})_e|. \tag{3.14}$$

We comment that the above localized error estimator is obtained for free in practice, since we have $(DG\boldsymbol{v} - \boldsymbol{\tau})_e$ available from the computation of the global error estimator $\psi(\boldsymbol{\tau})$ (see step 4 in Algorithm 3.3).

This localized error estimator (3.14) then can be used to design adaptive AMG methods. For example, it can be utilized in generating coarser aggregations that approximate the fine aggregates (vertices) accurately [64] or generate approximations to the level sets of the error for the path cover adaptive AMG method [28].

4. Numerical results. In this section, we present results of some numerical experiments demonstrating the efficiency of the a posteriori error estimator.

4.1. Tests on two-dimensional uniform grids. We first test the performance of the algorithm on the unweighted graph Laplacian L of two-dimensional (2D) uniform triangle grids, which corresponds to solving a Poisson equation on a 2D square

TABLE 1

Efficiency of the error estimator on graph Laplacian systems on uniform triangle grids of different sizes. The value of the estimator $\psi(\boldsymbol{\tau})$ is computed by solving (3.10) *approximately with* 1*,* 3*, and* 5 *iterations of the overlapping Schwarz method. The CPU time (in seconds) is also shown in the table.*

		1 iter			3 iters			5 iters		
$\lvert\mathcal{V}\rvert$	$\lVert\boldsymbol{u}-\boldsymbol{v}\rVert_L$	$\psi(\boldsymbol{\tau})$	e_{ff}	time	$\psi(\boldsymbol{\tau})$	e_{ff}	time	$\psi(\boldsymbol{\tau})$	e_{ff}	time
1089	1.73	2.25	1.30	0.03	1.99	1.15	0.04	1.91	1.10	0.06
4225	1.73	2.67	1.55	0.05	2.28	1.32	0.11	2.16	1.25	0.16
16641	1.73	3.36	1.95	0.14	2.76	1.60	0.37	2.56	1.48	0.62
66049	1.72	4.43	2.57	0.53	3.51	2.03	1.40	3.20	1.86	2.31
263169	1.72	6.01	3.49	1.92	4.66	2.71	5.64	4.19	2.43	9.53

domain with the Neumann boundary condition. The uniform triangle grid with grid size $h = 2^{-l}$, $l = 5, 6, 7, 8, 9$ is used, and we take $\boldsymbol{u} = \sin(\frac{\pi}{2}\boldsymbol{x})\sin(\frac{\pi}{2}\boldsymbol{y})$. We set the approximate solution $\boldsymbol{v} = \mathbf{0}$ and obtain the a posteriori error estimator $\psi(\boldsymbol{\tau})$ with Algorithm 3.3, in which the minimization problem (3.10) is solved approximately with several iterations of the overlapping Schwarz method. We use the face cycle bases that correspond to the small triangles in the grid (cycle length is 3). With this choice of cycle basis, each of the decomposed subspaces in (3.13) have dimension $\mathcal{O}(1)$ since there are at most six cycles incident with a given vertex i. The low dimension of the subspaces assures that solving (3.12) costs no more than $\mathcal{O}(1)$ computation, and thus the computation cost of one iteration of the Schwarz method remains $\mathcal{O}(n)$.

In Table 1, we report the true error and the a posteriori error estimator $\psi(\boldsymbol{\tau})$ on graph Laplacian systems of different scales. $e_{\mathit{ff}} := \frac{\psi(\boldsymbol{\tau})}{\lVert\boldsymbol{u}-\boldsymbol{v}\rVert_L}$ is also reported to show the efficiency of the error estimator. From Table 1, we observe that the CPU time for one iteration of the Schwarz method grows linearly as the size of the graph Laplacian systems increases. The error estimator $\psi(\boldsymbol{\tau})$ gradually approaches the true error $\lVert\boldsymbol{u}-\boldsymbol{v}\rVert_L$ when we increase the steps of Schwarz iteration.

More importantly, we would like to know whether the localized error estimator (3.14) approximates the true error on each edge accurately, since the localized estimation is the key to an effective coarsening scheme in adaptive AMG. Take L as the weighted graph Laplacian of the uniform grid with grid size $h = 2^{-5}$, $\boldsymbol{u} = \sin(\frac{\pi}{2}\boldsymbol{x})\sin(\frac{\pi}{2}\boldsymbol{y})$, and $\boldsymbol{v}$ obtained by three iterations of Gauss-Seidel method with random initial guess. We compute the error estimator using three iterations of the Schwarz method to solve the minimization problem in Algorithm 3.3.

In Figure 2, we plot the difference between the true error and the error estimator on each edge. On most of the edges the error estimator captures the true error well since the difference $\psi(\boldsymbol{\tau}) - \lVert\boldsymbol{u}-\boldsymbol{v}\rVert_L$ is no larger than 0.02.

4.2. Tests on "real-world" graphs. In this section we test the proposed error estimator on some real-world graphs from the SuiteSparse Matrix Collection [23]. We preprocess the undirected graphs by extracting the largest connected component of each graph and deleting self-loops. For each of these graphs, if the original edge weight is negative, we take its absolute value.

In Table 2, we summarize the basic information of the graphs and the performance of the error estimator. In our setting $\boldsymbol{u}$ is the exact solution for a problem with an arbitrarily chosen right-hand side $\boldsymbol{f}$. The approximate solution $\boldsymbol{v}$ is obtained as a result of three iterations of the Gauss–Seidel method with this right-hand side. To compute the error estimator, we first use the BFS algorithm to find a spanning tree and then construct the spanning-tree-induced fundamental cycle basis. Then

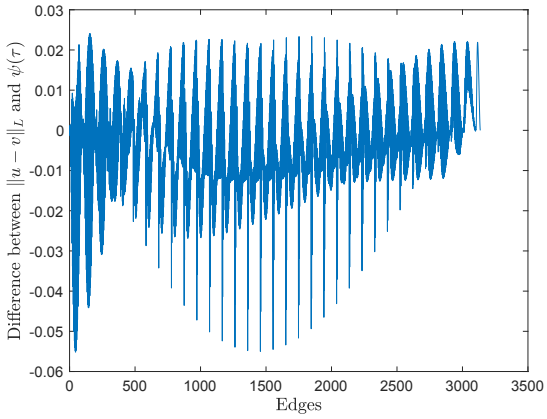


FIG. 2. *Difference between the true error* $\|\boldsymbol{u}-\boldsymbol{v}\|_L$ *and error estimator* $\|DG\boldsymbol{v}-\boldsymbol{\tau}\|_{D^{-1}}$ *on each edge* e.

TABLE 2

Efficiency of error estimator on graph Laplacian systems arising from real-world applications. The value of the estimator $\psi(\boldsymbol{\tau})$ *is computed by solving* (3.10) *approximately with* 3 *iterations of the Schwarz method. The graph types tested are unweighted (u) and weighted (w).*

ID	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Problem type	Type	$\|\boldsymbol{u}-\boldsymbol{v}\|_L$	$\psi(\boldsymbol{\tau})$	e_{ff}
8	292	958	Least-squares problem	u	1.74	1.75	1.00
1196	1879	5525	Circuit simulation	w	2.71	2.71	1.00
22	5300	8271	Power network	u	5.82	5.82	1.00
1614	2048	4034	Electromagnetic problem	w	0.47	0.50	1.07
33	1423	16342	Structural problem	w	14.5	19.7	1.36
791	8205	58681	Acoustic problem	w	23.8	37.7	1.58
2777	1857	13762	Social network	u	52.9	76.3	1.44
1533	2361	13828	Protein network	u	4.61	4.70	1.01

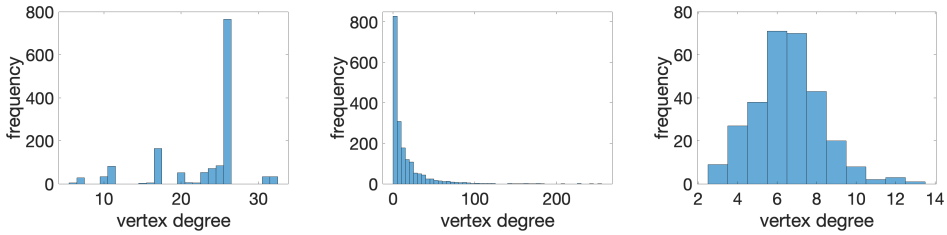
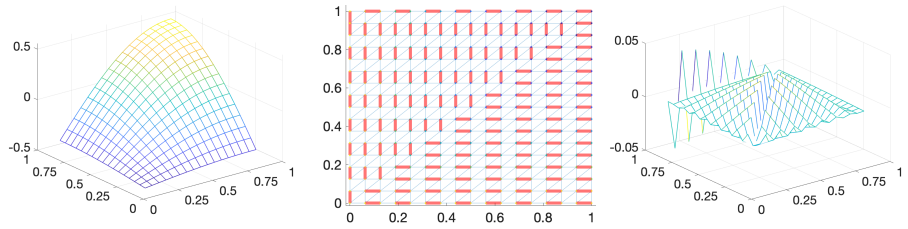


FIG. 3. *Representative degree distribution of networks studied in Table* 2. Left*: Network ID* 33. Middle*: Network ID:* 2777 *(power law distribution).* Right: *Network ID* 8 *(normal distribution).*

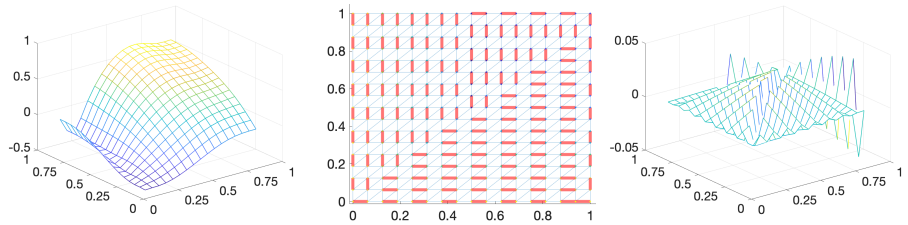
we apply three steps of Schwarz iterations to solve the minimization problem in Algorithm 3.3 to compute the overall error estimator. In Figure 3 we plot the degree distribution of selected networks. The differences in these degree distributions suggest that these networks have distinctive structural and dynamical properties. As we can see from the results, for real-world graphs with different sizes, structures, and densities, the error estimators approximate the true errors well in all cases, which demonstrates the effectiveness of our proposed algorithm for computing the a posteriori error estimator.



(a) True error of the current solution.

(b) Path cover aggregation based on the known exact error.

(c) Difference between the smooth error and the error after restriction and prolongation with aggregation based on true error.



A posteriori error estimates of the current solution.

(e) Path cover aggregation based on a posteriori error estimator.

(f) Difference between the smooth error and the error after restriction and prolongation with aggregation based on error estimates.

FIG. 4. *Path-cover aggregation generated with the exact error and the a posteriori error estimates and the difference between the smooth error and the error after restriction and prolongation. The aggregation based on the error estimator assembles the one based on the exact error, but the formal one is computable at a modest cost.*

4.3. Tests on building aggregations for AMG. Designing an effective coarsening scheme that projects the smooth error onto coarse levels accurately is the key to the AMG methods. One example is the path-cover adaptive AMG (PC-αAMG) proposed in [28] for solving weighted graph Laplacian linear systems. Provided an accurate estimation of the current error, PC-αAMG first forms A vertex-disjoint path cover where paths approximate the level sets of the smooth error, then aggregates vertices to form aggregations.

While an accurate error estimate makes PC-αAMG highly efficient, the step to compute estimation of the true error in the original PC-αAMG appears to be expensive in [28]. Our a posteriori error estimator offers an ideal replacement which gives an accurate approximation in an efficient manner.

Here, we present the resulting path-cover aggregation obtained by using the proposed a posteriori error estimator for solving the unweighted graph Laplacian on the 2D uniform grid. We solve the corresponding linear system approximately with several iterations of the relaxation scheme and curated the exact true error of the current solution (plotted in Figure 4(a)). Note that in practice this exact error is not directly accessible, and we approximate it with the proposed a posteriori error estimator plotted in Figure 4(d). In Figure 4(b) and 4(e) we compare the path-cover aggregation

generated with the true error and the a posteriori error estimator (computed approximately with 3 iterations of the Schwarz method). The aggregation patterns are very similar.

To check whether the smooth error is transferred and represented accurately on the coarse level, we restrict the smooth error to the coarse level and then prolongate it back. We plot the differences between the smooth error and the error after restriction and prolongation in Figure 4(c) for the case where the coarse level is constructed based on the exact smooth error and in Figure 4(f) for the case where the coarse level is constructed based on the a posteriori error estimator, respectively. As we can see, although the shapes of differences in the two cases are different, the magnitudes of both cases are 0.045, which indicates that the aggregation built with the error estimator is effective in capturing the true smooth error.

5. Conclusions. In this paper we proposed an a posteriori error estimator for solving linear systems of graph Laplacians. A novel approach is devised to reduce the computation cost of computing such an estimator in comparison to existing approaches. For sparse graphs this novel estimator can be calculated in nearly linear time. Our approach is based on the Helmholtz decomposition on the graphs. It includes solving a linear system on a spanning tree and solving (approximately) a minimization problem in the cycle space of the graph.

In the future, we plan to incorporate this error estimator in the adaptive AMG coarsening schemes. For example, as briefly discussed in subsection 4.3, the proposed estimator can be used in the path-cover adaptive AMG proposed in [28].

REFERENCES

[1] M. Ainsworth and J. Oden, *A posteriori error estimation in finite element analysis*, Comput. Methods Appl. Mech. Engrg., 142 (1997), pp. 1–88.

[2] I. Babuška and W. C. Rheinboldt, *A-posteriori error estimates for the finite element method*, Internat. J. Numer. Methods Engrg., 12 (1978), pp. 1597–1615.

[3] M. Belkin and P. Niyogi, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, in Advances in Neural Information Processing Systems, 14(NIPS2001)/T.G. Dietterich, S. Becker, and Z. Ghahramani, eds., MIT Press, Cambridge, 2001, pp. 585–591.

[4] B. Bollobás, *Modern Graph Theory*, vol. 184, Springer Science & Business Media, New York, 2013.

[5] M. Bolten, S. Friedhoff, A. Frommer, M. Heming, and K. Kahl, *Algebraic multigrid methods for Laplacians of graphs*, Linear Algebra Appl., 434 (2011), pp. 2225–2243.

[6] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, *Network analysis in the social sciences*, Science, 323 (2009), pp. 892–895.

[7] A. Brandt, J. Brannick, K. Kahl, and I. Livshits, *Bootstrap algebraic multigrid: Status report, open problems, and outlook*, Numer. Math. Theory, Methods Appl., 8 (2015), pp. 112–135.

[8] A. Brandt, J. J. Brannick, K. Kahl, and I. Livshits, *Bootstrap AMG*, SIAM J. Sci. Comput., 33 (2011), pp. 612–632.

[9] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, report, for Computational Studies, Fort Collins, CO, 1982.

[10] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov, *An algebraic multigrid method based on matching in graphs*, in Domain Decomposition Methods in Science and Engineering XX, Springer, Berlin, Heidelberg, 2013, pp. 143–150.

[11] J. Brannick, Y. Chen, J. Kraus, and L. Zikatanov, *Algebraic multilevel preconditioners for the graph Laplacian based on matching in graphs*, SIAM J. Numer. Anal., 51 (2013), pp. 1805–1827.

[12] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation* (αSA), SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[13] M. Brezina, R. Falgout, S. Maclachlan, T. Manteuffel, S. Mccormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.
[14] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[15] M. Brezina, R. Falgout, S. MacLachlan, T. A. Manteuffel, S. F. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.
[16] M. Brezina, T. Manteuffel, S. MCormick, J. Ruge, and G. Sanders, *Towards adaptive smoothed aggregation (αSA) for nonsymmetric problems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.
[17] W. Briggs, V. Henson, and S. McCormick, *A Multigrid Tutorial: Second Edition*, SIAM, Philadelphia, 2000.
[18] E. Bullmore and O. Sporns, *Complex brain networks: Graph theoretical analysis of structural and functional systems*, Nat. Rev. Neurosci., 10 (2009), pp. 186–198.
[19] L. Chen, X. Hu, and S. Wise, *Convergence analysis of the fast subspace descent method for convex optimization problems*, Math. Comp., 89 (2020), pp. 2249–2282.
[20] C. Colley, J. Lin, X. Hu, and S. Aeron, *Algebraic multigrid for least squares problems on graphs with applications to Hodgerank*, in Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Lake Bvena Vista, FL, May 2017, pp. 627–636.
[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge 2009.
[22] P. D'Ambra and P. S. Vassilevski, *Adaptive AMG with coarsening based on compatible weighted matching*, Comput. Vis. Sci., 16 (2013), pp. 59–76.
[23] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011).
[24] P. Destuynder and B. Métivet, *Explicit error bounds in a conforming finite element method*, Math. Comput., 68 (1999), pp. 1379–1396.
[25] F. Fouss, A. Pirotte, J. M. Renders, and M. Saerens, *Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation*, IEEE Trans. Knowl. Data Eng., 19 (2007), pp. 355–369.
[26] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, *Semisupervised image classification with laplacian support vector machines*, IEEE Geosci. Remote Sens. Lett., 5 (2008), pp. 336–340.
[27] W. L. Hamilton, R. Ying, and J. Leskovec, *Representation learning on graphs: methods and applications*, 2017, preprint, arXiv:1709.05584, http://arixiv.org/abs/1709.05584.
[28] X. Hu, J. Lin, and L. T. Zikatanov, *An adaptive multigrid method based on path cover*, SIAM J. Sci. Comput., 41 (2018), pp. S220–S241.
[29] N. T. I. Gutman, *Graph theory and molecular orbitals. Total ϕ-electron energy of alternant hydrocarbons*, Chem. Phys. Lett., 17 (1972), pp. 535–538.
[30] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. A. Zweig, *Cycle bases in graphs characterization, algorithms, complexity, and applications*, Comput. Sci. Rev., 3 (2009), pp. 199–243.
[31] D. W. Kelly, J. P. De S. R. Gago, O. C. Zienkiewicz, and I. Babuska, *A posteriori error analysis and adaptive processes in the finite element method: Part* I*-error analysis*, Internat. J. Numer. Methods Engrg., 19 (1983), pp. 1593–1619.
[32] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, *A simple, combinatorial algorithm for solving sdd systems in nearly-linear time*, in Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, New York, 2013, pp. 911–920.
[33] H. Kim, J. Xu, and L. Zikatanov, *A multigrid method based on graph matching for convection-diffusion equations*, Numer. Linear Algebra Appl., 10 (2003), pp. 181–195.
[34] L. Koenigsberger, *Hermann von Helmholtz*, F. Vieweg, Braunshweig, 1902 p. 357.
[35] I. Koutis, G. L. Miller, and D. Tolliver, *Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing*, Comput. Vis. Image Underst., 115 (2011), pp. 1638–1646.
[36] J. K. Kraus and S. K. Tomar, *Algebraic multilevel iteration method for lowest order Raviart-Thomas space and applications*, Internat. J. Numer. Methods Engrg., 86 (2011), pp. 1175–1196.
[37] D. Krishnan, R. Fattal, and R. Szeliski, *Efficient preconditioning of Laplacian matrices for computer graphics*, ACM Trans. Graph., 32 (2013).
[38] J. Lin, L. J. Cowen, B. Hescott, and X. Hu, *Computing the diffusion state distance on graphs via algebraic multigrid and random projections*, Numer. Linear Algebra Appl., 25 (2018), p. e2156.

[39] W. Liu, Y. Jiang, J. Luo, and S. Chang, *Noise resistant graph ranking for improved web image search*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 849–856.
[40] O. Livne and A. Brandt, *Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver*, SIAM J. Sci. Comput., 34 (2012), pp. 499–523.
[41] J. Ma, W. Huang, S. Segarra, and A. Ribeiro, *Diffusion filtering of graph signals and its use in recommendation systems*, in Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2016, pp. 4563–4567.
[42] S. P. MacLachlan, J. D. Moulton, and T. P. Chartier, *Robust and adaptive multigrid methods: Comparing structured and algebraic approaches*, Numer. Linear Algebra Appl., 19 (2012), pp. 389–413.
[43] R. Merris, *Laplacian matrices of graphs: A survey*, Linear Algebra Appl., 197-198 (1994), pp. 143–176.
[44] A. Nägel, R. D. Falgout, and G. Wittum, *Filtering algebraic multigrid and adaptive strategies*, Comput. Vis. Sci., 11 (2008), pp. 159–167.
[45] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 34 (2012), pp. A1079–A1109.
[46] A. Napov and Y. Notay, *An efficient multigrid method for graph laplacian systems*, Electron. Trans. Numer. Anal., 45 (2016), pp. 201–218.
[47] F. Nie, X. Wang, M. I. Jordan, and H. Huang, *The Constrained Laplacian Rank Algorithm for Graph-based Clustering*, in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI Press, 2016, pp. 1969–1976.
[48] R. H. Nochetto and A. Veeser, *Primer of Adaptive Finite Element Methods*, Springer, Berlin, Heidelberg, 2012, pp. 125–225.
[49] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.
[50] L. Page, S. Brin, R. Motwani, and T. Winograd, *The Pagerank Citation Ranking: Bringing Order to the Web*, Technical report, Stanford InfoLab, 1999.
[51] W. Prager and J. L. Synge, *Approximations in elasticity based on the concept of function space*, Q. Appl. Math., 5 (1947), pp. 241–269.
[52] S. I. Repin and S. K. Tomar, *Guaranteed and robust error bounds for nonconforming approximations of elliptic problems*, IMA J. Numer. Anal., 31 (2010), pp. 597–615.
[53] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.
[54] G. Rucker, *Network meta-analysis, electrical networks and graph theory*, Res. Syn. Methods, 3 (2012), pp. 312–324.
[55] X. Tai and J. Xu, *Global and uniform convergence of subspace correction methods for some convex optimization problems*, Math. Comput., 71 (2002), p. 105–124.
[56] S. Tomar and S. Repin, *Efficient computable error bounds for discontinuous Galerkin approximations of elliptic problems*, J. Comput. Appl. Math., 226 (2009), pp. 358–369.
[57] J. Urschel, J. Xu, X. Hu, and L. Zikatanov, *A cascadic multigrid algorithm for computing the Fiedler vector of graph laplacians*, J. Comput. Math., 33 (2015), pp. 209–226.
[58] R. V. urth, *A posteriori error estimates for nonlinear problems. Finite element discretizations of elliptic equations*, Math. Comput., 62 (1994), pp. 445–475.
[59] P. S. Vassilevski and L. T. Zikatanov, *Commuting projections on graphs*, Numer. Linear Algebra Appl., 21 (2014), pp. 297–315.
[60] M. Wang, H. Li, D. Tao, K. Lu, and X. Wu, *Multimodal graph-based reranking for web image search*, IEEE Trans. Image Process., 21 (2012), pp. 4649–4661.
[61] K. Q. Weinberger, F. Sha, Q. Zhu, and L. K. Saul, *Graph Laplacian regularization for large-scale semidefinite programming*, in Proceedings of the 19th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, 2006, pp. 1489–1496.
[62] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.
[63] J. Xu and L. Zikatanov, *Algebraic multigrid methods*, Acta Numer., 26 (2017), pp. 591–721.
[64] W. Xu and L. T. Zikatanov, *Adaptive aggregation on graphs*, J. Comput. Appl. Math., 340 (2018), pp. 718–730.
[65] J. Yu, D. Tao, and M. Wang, *Adaptive hypergraph learning and its application in image classification*, IEEE Trans. Image Process., 21 (2012), pp. 3262–3272.