Sparsified Block Elimination for Directed Laplacians[*]

Richard Peng
University of Waterloo
Waterloo, Canada
y5peng@uwaterloo.ca

Zhuoqing Song
Fudan University
Shanghai, China
zqsong19@fudan.edu.cn

ABSTRACT

We show that the sparsified block elimination algorithm for solving undirected Laplacian linear systems from [Kyng-Lee-Peng-Sachdeva-Spielman STOC'16] directly works for directed Laplacians. Given access to a sparsification algorithm that, on graphs with n vertices and m edges, takes time $\mathcal{T}_{\mathrm{S}}(m)$ to output a sparsifier with $\mathcal{N}_{\mathrm{S}}(n)$ edges, our algorithm solves a directed Eulerian system on n vertices and m edges to ϵ relative accuracy in time

$$O(\mathcal{T}_{\mathrm{S}}(m) + \mathcal{N}_{\mathrm{S}}(n) \log n \log(n/\epsilon)) + \tilde{O}(\mathcal{T}_{\mathrm{S}}(\mathcal{N}_{\mathrm{S}}(n)) \log n),$$

where the $\tilde{O}(\cdot)$ notation hides $\log\log(n)$ factors. By previous results, this implies improved runtimes for linear systems in strongly connected directed graphs, PageRank matrices, and asymmetric M-matrices. When combined with slower constructions of smaller Eulerian sparsifiers based on short cycle decompositions, it also gives a solver algorithm that, after pre-processing the matrix in $O(n^2 \log^{O(1)} n)$ time, takes $O(n \log^5 n \log(n/\epsilon))$ time per solve. At the core of our analyses are constructions of augmented matrices whose Schur complements encode error matrices.

CCS CONCEPTS

• **Theory of computation** → **Data structures design and analysis**; **Graph algorithms analysis**; *Preconditioning*; *Random walks and Markov chains.*

KEYWORDS

linear systems, matrices, Markov chains

ACM Reference Format:
Richard Peng and Zhuoqing Song. 2022. Sparsified Block Elimination for Directed Laplacians. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3519935.3520053

1 INTRODUCTION

The design of efficient solvers for systems of linear equations in graph Laplacian matrices and their extensions has been a highly fruitful topic in algorithms. Laplacian matrices directly correspond to undirected graphs: off-diagonal entries are negations of edge weights, while the diagonal entries contain weighted degrees. Solvers for Laplacian matrices led to breakthroughs in fundamental problems in combinatorial optimization. Tools developed during such studies have in turn influenced data structures, randomized numerical linear algebra, scientific computing, and network science [17, 18].

An important direction in this Laplacian paradigm of designing graph algorithms is extending tools developed for undirected Laplacian matrices to directed graphs. Here a perspective from random walks and Markov chains leads to directed Laplacian matrices [8]. Such matrices have directed edge weights in off-diagonal entries, and weighted out-degrees on diagonals. In contrast to solving linear systems in undirected Laplacians, solving linear systems in directed Laplacians is significantly less well-understood. Almost-linear time [7] and nearly-linear time solvers [6] were developed very recently, and involve many more moving pieces.

In particular, the nearly-linear time algorithm from [6] combined block Gaussian elimination with single variables/vertex elimination, analyzed using matrix Martingales. In contrast, for undirected Laplacians, both block elimination [10] or matrix Martingales [11] can give different nearly-linear time solver algorithms, and there also exists more combinatorial approaches [9]. In this paper, we simplify this picture for directed Laplacian solvers by providing an analog of the sparsified Cholesky/multi-grid solver from [10]. This algorithm's running time is close to the limit of sparsification based algorithms: the running time of invoking a sparsification routine on its own output. Formally, we show:

Theorem 1.1. *Given a strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and an error parameter $\epsilon \in (0,1)$, we can process it in time $O\left(\mathcal{T}_{\mathrm{S}}(m,n,1)\right) + \tilde{O}\left(\mathcal{T}_{\mathrm{S}}\left(\mathcal{N}_{\mathrm{S}}(n,1), n, 1\right)\log n\right)$ so that, with high probability, given any vector $\boldsymbol{b} \in \mathbb{R}^n$ with $\boldsymbol{b} \perp \boldsymbol{1}$, we can compute a vector $\boldsymbol{x} \in \mathbb{R}^n$ in time $O\left(\mathcal{N}_{\mathrm{S}}(n,1)\log n \log(n/\epsilon)\right)$ such that*

$$\left\|\boldsymbol{x} - \boldsymbol{L}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}[\boldsymbol{L}]} \le \epsilon \left\|\boldsymbol{L}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}[\boldsymbol{L}]},$$

where $\boldsymbol{U}[\boldsymbol{L}] = \left(\boldsymbol{L} + \boldsymbol{L}^{\top}\right)/2$.

This result improves the at least $\Omega(\log^5 n)$ factor overhead upon sparsification of the previous nearly-linear time directed Laplacian solver [6], and is analogous to the current best overheads for sparsification based solvers for undirected Laplacians [10]. From the existence of sparsifiers of size $O(n \log^4 n\epsilon^{-2})$ [5], we also obtain the existence of $O(n \log^5 n \log(n/\epsilon))$ time solver routines that require quadratic time preprocessing to compute. As with other improved solvers for directed Laplacians our improvements directly applies to applications of such solvers, including random walk related quantities [8], as well as PageRank / Perron-Frobenius vectors [1].

[*]A full version of this paper is available on arXiv [15]. Richard Peng is supported by National Science Foundation (NSF) under Grant No. 1846218. Part of this work was done at Georgia Tech. Zhuoqing Song is supported by Shanghai Science and Technology Program [Project No. 21JC1400600 and No. 20JC1412700].

Our result complements recent developments of better sparsifiers of Eulerian Laplacians [5, 12, 13]. By analyzing a pseudocode that's entirely analogous to the undirected block-elimination algorithm from [10], we narrow the gap between Laplacian solvers for directed and undirected graphs. Our result also emphasizes the need for better directed sparsification routines. While there is a rich literature on undirected sparsification [3], the current best directed sparsification algorithms rely on expander decompositions, so have rather large logarithmic factor overheads. We discuss such bounds in detail in the full version.

Finally, our analysis of this more direct algorithm require better understanding the accumulation errors in Eulerian Laplacians and their partially eliminated states, known as Schur Complements. It was observed in [6] that these objects are significantly less robust than their undirected analogs. Our analysis of these objects rely on augmentations of matrices: constructing larger matrices whose Schur complements correspond to the final objects we wish to approximate, and bounding errors on these larger matrices instead. This approach has roots in symbolic computation, and can be viewed as a generalization of low-rank perturbation formulas such as Sherman-Morrison-Woodbury [19]. We believe both this algebraic technique, and the additional robustness properties of directed Schur Complements we show, may be of independent interest.

1.1 Related Works

Directed Laplacian matrices arise in problems related to directed random walks / non-reversible Markov chains, such as computations of stationary distributions, hitting times and escape probabilities. A formal treatment of applying an Eulerian solver to these problems can be found in [8] and [1]. Adaptations of Eulerian Laplacian solvers have also led to improved bounded-space algorithms for estimating random walk probabilities [2].

Our algorithm is most closely related to the previous nearly-linear time directed Laplacian solver [6]. That algorithm is motivated by single variable elimination and a matrix Martingale based analysis. However, it invokes both components of block elimination algorithms: finding strongly diagonally dominant subsets, and invoking sparsification as black-boxes. The runtime overhead of this routine over sparsification is at least $\log^5 n$: in [6], Lemma 5.1 gives that each phase (for a constant factor reduction) invokes sparsification $O(\log^2 n)$ times, and each call is ran with error at most $\frac{1}{O(\log^3 n)}$ (divided by $\log^2 n$ in Line 2 of Algorithm 2, and also by $\log n$ in Line 5 of Algorithm 3).

While our algorithms are directed analogs of the undirected block elimination routines from in [10], our analyses rely on many structures developed in [6]. Specifically, our cumulative error during elimination steps is bounded via the matrix that's the sum of undiretifications of the intermediate directed matrices. On the other hand, we believe our algorithm is more natural: our sampling no longer needs to be locally unbiased, the per-step errors do not need to be decreased by polylog factors, and the algorithm is no longer divided into inner/outer phases. This more streamlined algorithm leads to our runtime improvements.

Our Schur Complement sparisifcation algorithm is based on the partial block elimination routine from [10], which is in turn based on a two-term decomposition formula for (pseudo-)inverses from [16]. We remark that there is a good sparsification routine in the low space setting [2]. There is a subsequent algorithm that replaces this decomposition with directly powering via random walks [4] that's also applicable for sparsifying undirected Schur Complements. However, that algorithm relies on sparsifying 3-step random walk polynomials, which to our knowledge, is a subroutine that has not been studied in directed settings. As a result, we are unable to utilize this later development directly.

The existence of $O(n \log^4 n)$ sized sparsifiers in [5] relies on decomposing unit weighted graphs into short cycles and $O(n)$ extra edges. While this decomposition has a simple $O(m^2)$ time algorithm (peel off all vertices with degree < 3, then return the lowest cross-edge in the BFS tree), the current fastest construction of it takes $m^{1+o(1)}$ time [5, 12, 13]. As a result, we need to instead invoke the more expensive, graph decomposition based, algorithms from [7] for sparsification. Also, we can only use the naive $O(m^2)$ construction of $O(\log n)$-lengthed cycle decompositions (after an initial sparsification call to make $m = O(n \log^{O(1)} n)$) because the almost-linear time algorithm in [12] produces $O(\log^2 n)$-lengthed cycles.

2 PRELIMINARY

2.1 Notations

The notation $\tilde{O}(\cdot)$ suppresses the *polyloglog*(n) factors in this paper. We let $[n] = \{1, 2, \cdots, n\}$.

Matrix: For matrix $\boldsymbol{A}$, $\mathbf{nnz}(\boldsymbol{A})$ denotes its number of nonzero entries. For subsets $T_1, T_2 \subseteq [n]$, $\boldsymbol{A}_{T_1 T_2} \in \mathbb{R}^{|T_1| \times |T_2|}$ is the submatrix containing the entries with row indexes and column indexes in (T_1, T_2); and $\boldsymbol{A}_{-T_1,-T_2}$ is the submatrix of $\boldsymbol{A}$ by removing the rows indexed by T_1 and columns indexed by T_2. For vector $\boldsymbol{v} \in \mathbb{R}^n$ and subset $C \subseteq [n]$, $\boldsymbol{v}_C$ is the subvector of $\boldsymbol{v}$ containing the entries indexed by C.

We use $\boldsymbol{I}_a$, $\mathbf{0}_{b \times c}$ to denote the identity matrix of size a and the b-by-c zero matrix, and we sometimes omit the subscripts when their sizes can be determined from the context. For any matrix $\boldsymbol{X} \in \mathbb{R}^{a \times b}$ and set $T_1, T_2 \subseteq [n]$ with $|T_1| = a$, $|T_2| = b$, $\mathcal{P}(\boldsymbol{X}, T_1, T_2, n)$ denotes an n-by-n matrix whose submatrix indexed by (T_1, T_2) equals $\boldsymbol{X}$ and all the other entries equal 0. In other words, $\mathcal{P}(\boldsymbol{X}, T_1, T_2, n)$ can be regarded as replacing the submatrix indexed by (T_1, T_2) with $\boldsymbol{X}$ in the zero matrix $\mathbf{0}_{n \times n}$.

For symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, we use $\lambda_i(\boldsymbol{A})$ to denotes its i-th smallest eigenvalue. For symmetric matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$, we use $\boldsymbol{A} \succeq \boldsymbol{B}$ ($\boldsymbol{A} \succ \boldsymbol{B}$) to indicate that for any $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} \geq \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}$ ($\boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} > \boldsymbol{x}^\top \boldsymbol{B} \boldsymbol{x}$). We define $\preceq$, $\prec$ analogously. A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semidefinite (PSD) iff $\boldsymbol{A}$ is symmetric and $\boldsymbol{A} \succeq \mathbf{0}$; $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive definite (PD) iff $\boldsymbol{A}$ is symmetric and $\boldsymbol{A} \succ \mathbf{0}$.

For PSD matrix $\boldsymbol{A}$, $\boldsymbol{A}^{1/2}$ is its square root; $\boldsymbol{A}^\dagger$ denotes its Moore-Penrose pseudoinverse; $\boldsymbol{A}^{\dagger/2}$ is the square root of its Moore-Penrose pseudoinverse.

Vector: $\mathbf{1}_a$, $\mathbf{0}_b$ denote the a-dimensional all-ones vector and b-dimensional all-zeros vector; when their sizes can be determined from the context, we sometimes omit the subscripts.

For matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, $\mathbf{Diag}(\boldsymbol{A})$ is an n-by-n diagonal matrix with the same diagonal entries as $\boldsymbol{A}$. For vector $\boldsymbol{x} \in \mathbb{R}^n$, $\mathbf{Diag}(\boldsymbol{x})$

denotes an n-by-n diagonal matrix with its i-th diagonal entry equalling $\boldsymbol{x}_i$.

For any positive semidefinite matrix $\boldsymbol{A}$, we define the vector norm $\|\boldsymbol{x}\|_{\boldsymbol{A}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}$. And $\|\cdot\|_p$ denotes the ℓ_p norm.

Matrix norm: $\|\cdot\|_p$ denotes the ℓ_p norm. For instance, for $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\|\boldsymbol{A}\|_2 = \sqrt{\lambda_n\left(\boldsymbol{A}^\top \boldsymbol{A}\right)}$; $\|\boldsymbol{A}\|_\infty = \max_{i\in[n]} \sum_{j=1}^n \left|\boldsymbol{A}_{ij}\right|$. For matrix $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ and PSD matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we denote $\|\boldsymbol{B}\|_{\boldsymbol{A}\to\boldsymbol{A}} = \sup_{\|\boldsymbol{x}\|_{\boldsymbol{A}}\neq 0} \frac{\|\boldsymbol{B}\boldsymbol{x}\|_{\boldsymbol{A}}}{\|\boldsymbol{x}\|_{\boldsymbol{A}}}$.

Schur complement: For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and F, C a partition of $[n]$ such that $\boldsymbol{A}_{FF}$ is nonsingular, the Schur complement of F in $\boldsymbol{A}$ is defined as $\mathrm{Sc}(\boldsymbol{A}, F) = \boldsymbol{A}_{CC} - \boldsymbol{A}_{CF}\boldsymbol{A}_{FF}^{-1}\boldsymbol{A}_{FC}$.

When we need to emphasize the support set of the entries that remain, we also denote $\mathrm{Sc}(\boldsymbol{A}, -C) = \mathrm{Sc}(\boldsymbol{A}, F)$.

2.2 (Directed) Laplacians, Symmetrizations

A matrix $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is called a directed Laplacian iff $\boldsymbol{1}^\top \boldsymbol{L} = \boldsymbol{0}$ and all off-diagonal entries of $\boldsymbol{L}$ are non-positive, i.e., $\boldsymbol{L}_{ii} = -\sum_{j:j\neq i} \boldsymbol{L}_{ji}$ for all $i \in [n]$ and $\boldsymbol{L}_{ij} \leq 0$ for all $i \neq j$. A (directed) Laplacian $\boldsymbol{L}$ can be associated with a (directed) graph $\mathcal{G}[\boldsymbol{L}]$ whose adjacency matrix is $\hat{\boldsymbol{A}} = \mathbf{Diag}(\boldsymbol{L}) - \boldsymbol{L}^\top$. The in-degrees/out-degrees of $\boldsymbol{L}$ are defined as the in-degrees/out-degrees of $\mathcal{G}[\boldsymbol{L}]$. For directed Laplacians, its out-degrees equal its diagonal entries. If $\mathcal{G}[\boldsymbol{L}]$ is strongly connected, we say the (directed) Laplacian $\boldsymbol{L}$ is strongly connected.

In addition, if $\boldsymbol{L}\boldsymbol{1} = \boldsymbol{0}^\top$, we call $\boldsymbol{L}$ an Eulerian Laplacian. These Laplacians have the property that in-degrees of vertices equal to out-degrees. The undirected Laplacian is a special case where $\boldsymbol{L} = \boldsymbol{L}^\top$. We often refer to these as symmetric Laplacians, or just Laplacians.

Symmetrization: For square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we define its matrix symmetrization as $\boldsymbol{U}[\boldsymbol{A}] = \frac{\boldsymbol{A}+\boldsymbol{A}^\top}{2}$. For a directed Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, we define its undirectification as

$$\mathcal{U}^{\mathrm{G}}[\boldsymbol{L}] = \frac{1}{2}\left(\boldsymbol{L} + \boldsymbol{L}^\top - \mathbf{Diag}\left(\left(\boldsymbol{L} + \boldsymbol{L}^\top\right)\boldsymbol{1}\right)\right).$$

$\mathcal{U}^{\mathrm{G}}[\boldsymbol{L}]$ is called the undirectification because it is a symmetric Laplacian whose adjacency matrix is $\boldsymbol{U}\left[\hat{\boldsymbol{A}}\right]$, where $\hat{\boldsymbol{A}} = \mathbf{Diag}(\boldsymbol{L}) - \boldsymbol{L}^\top$ is the adjacency matrix of $\mathcal{G}[\boldsymbol{L}]$. For an Eulerian Laplacian $\boldsymbol{L}$, its matrix symmetrization coincides with its undirectification, i.e., $\boldsymbol{U}[\boldsymbol{L}] = \mathcal{U}^{\mathrm{G}}[\boldsymbol{L}]$. Eulerian Laplacians are critically important in solvers for directed Laplacians because they are the only setting in which the undirectification is positive semidefinite.

Row Column Diagonal Dominant (RCDD): A square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is α-RCDD iff $\sum_{j\in[n]\setminus\{i\}} \left|\boldsymbol{A}_{ij}\right| \leq \frac{1}{1+\alpha}\boldsymbol{A}_{ii}$ and $\sum_{j\in[n]\setminus\{i\}} \left|\boldsymbol{A}_{ji}\right| \leq \frac{1}{1+\alpha}\boldsymbol{A}_{ii}$ for any $i \in [n]$. We also say $\boldsymbol{A}$ is RCDD if $\boldsymbol{A}$ is 0-RCDD.

2.3 Sparsification

All almost-linear time or faster solvers for directed Laplacians to date are built around sparsification: the approximation of graphs by ones with fewer edges. As it's difficult to even approximate reachability of directed graphs, [7] introduced the key idea of measuring approximations w.r.t. a symmetric PSD matrix. Such approximations are at the core of all subsequent algorithms, including ours.

Definition 2.1. (Asymmetrically bounded) Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and a PSD matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A}$ is *asymmetrically bounded* by $\boldsymbol{U}$ iff $\ker(\boldsymbol{U}) \subseteq \ker\left(\boldsymbol{A}^\top\right) \cap \ker(\boldsymbol{A})$ and $\left\|\boldsymbol{U}^{\dagger/2}\boldsymbol{A}\boldsymbol{U}^{\dagger/2}\right\|_2 \leq 1$. We denote it by $\boldsymbol{A} \overset{\mathrm{asym}}{\leq} \boldsymbol{U}$.

By our definition, $\boldsymbol{A} \overset{\mathrm{asym}}{\leq} \boldsymbol{U}$ is equivalent to $-\boldsymbol{A} \overset{\mathrm{asym}}{\leq} \boldsymbol{U}$. The following lemma is changed slightly from Lemma B.2 of [7].

Fact 2.2. For any matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and PSD matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$, the following statements are equivalent:

- $\boldsymbol{A} \overset{\mathrm{asym}}{\leq} \boldsymbol{U}$.
- $2\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{y} \leq \boldsymbol{x}^\top \boldsymbol{U}\boldsymbol{x} + \boldsymbol{y}^\top \boldsymbol{U}\boldsymbol{y}$, $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

Definition 2.3. (Approximation of directed Laplacians via undirectification) Given matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and directed Laplacian $\boldsymbol{B} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A}$ is an *ϵ-asymmetric approximation* of $\boldsymbol{B}$ iff $\boldsymbol{A} - \boldsymbol{B}$ is *asymmetrically bounded* by $\epsilon \cdot \mathcal{U}^{\mathrm{G}}[\boldsymbol{B}]$.

In particular, for strongly connected Eulerian Laplacians $\boldsymbol{A}$ and $\boldsymbol{B}$, $\boldsymbol{A}$ is an *ϵ-asymmetric approximation* of $\boldsymbol{B}$ iff

$$\left\|\boldsymbol{U}[\boldsymbol{B}]^{\dagger/2}(\boldsymbol{A} - \boldsymbol{B})\boldsymbol{U}[\boldsymbol{B}]^{\dagger/2}\right\|_2 \leq \epsilon.$$

We will utilize sparsifiers for Eulerian Laplacians [5, 7], as well as implicit sparsifiers for products of directed adjacency matrices as black boxes throughout our presentations. The formal statements of these black boxes are below.

Theorem 2.4. *(Directed Laplacian sparsification oracle) Given a directed Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{nnz}(\boldsymbol{L}) = m$ and error parameter $\delta \in (0, 1)$, there is an oracle* OraSparseLaplacian *which runs in at most $\mathcal{T}_{\mathrm{S}}(m, n, \delta)$ time, where $\mathcal{T}_{\mathrm{S}}(m, n, \delta) = O((m\log^{O(1)} n + n\log^{O(1)} n)\delta^{-O(1)})$, to return with high probability a directed Laplacian $\widetilde{\boldsymbol{L}}$ satisfying:*

(1) $\boldsymbol{nnz}\left(\widetilde{\boldsymbol{L}}\right) \leq \mathcal{N}_{\mathrm{S}}(n, \delta)$ *where* $\mathcal{N}_{\mathrm{S}}(n, \delta) = O(n\log^{O(1)} n\delta^{-O(1)})$;
(2) $\boldsymbol{Diag}\left(\widetilde{\boldsymbol{L}}\right) = \boldsymbol{Diag}(\boldsymbol{L})$;
(3) $\widetilde{\boldsymbol{L}} - \boldsymbol{L} \overset{\mathrm{asym}}{\leq} \delta \cdot \mathcal{U}^{\mathrm{G}}[\boldsymbol{L}]$.

Remark 2.5. Conditions (2), (3) in Theorem 2.4 above are equivalent to $\widetilde{\boldsymbol{L}}$ and $\boldsymbol{L}$ having the same in- and out-degrees, and

$$\left\|\mathcal{U}^{\mathrm{G}}[\boldsymbol{L}]^{\dagger/2}\left(\widetilde{\boldsymbol{L}} - \boldsymbol{L}\right)\mathcal{U}^{\mathrm{G}}[\boldsymbol{L}]^{\dagger/2}\right\|_2 \leq \delta$$

respectively. By having the same in-degrees and out-degrees, we mean $\mathbf{Diag}\left(\widetilde{\boldsymbol{L}}\right) = \mathbf{Diag}(\boldsymbol{L})$ and $\widetilde{\boldsymbol{L}}\boldsymbol{1} = \boldsymbol{L}\boldsymbol{1}$.

Lemma 2.6. *(Lemma 3.18 of [7]) Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ be nonnegative vectors with $\boldsymbol{nnz}(\boldsymbol{x}) + \boldsymbol{nnz}(\boldsymbol{y}) = m$ and let $\epsilon, p \in (0, 1)$. And we denote $\boldsymbol{G} = \left(\boldsymbol{1}^\top \boldsymbol{x}\right)\boldsymbol{Diag}(\boldsymbol{y}) - \boldsymbol{x}\boldsymbol{y}^\top$. Then, there is a routine* SparseProduct *which computes with probability at least $1 - p$ a nonnegative matrix $\boldsymbol{A}$ in $O\left(m\epsilon^{-2}\log\frac{m}{p}\right)$ time such that $\boldsymbol{nnz}(\boldsymbol{A}) = O\left(m\epsilon^{-2}\log\frac{m}{p}\right)$, $\boldsymbol{A} - \boldsymbol{x}\boldsymbol{y}^\top \overset{\mathrm{asym}}{\leq} \epsilon \cdot \mathcal{U}^{\mathrm{G}}[\boldsymbol{G}]$.*

Given an Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and a partition F, C of $[n]$, by invoking OraSparseLaplacian on subgraphs with edges inside (F, F), (F, C), (C, F), (C, C) respectively, we can get a Laplacian sparsification procedure SparseEulerianFC so that the sparsified Eulerian Laplacians returned by SparseEulerianFC not only satisfy all the properties mentioned in Theorem 2.4, but also keep the in-degrees and out-degrees of the subgraph supported by (F, F).

Analogously, a routine SparseProductFC can be constructed by applying SparseProduct four time.

Explicit definitions of SparseEulerianFC and SparseProductFC can be found in the full version.

2.4 Sufficiency of Solving Eulerian Systems to Constant Error

Previous works on solvers for directed Laplacians and their generalizations (to RCDD and M-matrices) established that it's sufficient to solve Eulerian systems to constant relative accuracy in their undirectification.

- The iterative refinement procedure shown in [7] shows that a constant accuracy solver can be amplified to one with ϵ relative accuracy in $O(\log(1/\epsilon))$ iterations.
- The stationary computation procedure in [8] showed that arbitrary strongly connected Laplacians with mixing time T_{mix} can be solved to 2-norm error ϵ by solving $O(\log(T_{mix}/\epsilon))$ systems in Eulerian Laplacians. This was subsequently simplified and extended to M-matrices and row-column-diagonally-dominant matrices in [1] (with an extra $\log n$ factor in running time). A purely random walk (instead of matrix perturbation) based outer loop is also given in the thesis of Peebles [14].

3 OVERVIEW

Our algorithm is based on sparse Gaussian elimination. Before we discuss the block version, it is useful to first describe how the single variable version works on Eulerian Laplacians.

Recall that Eulerian Laplacians store the (weighted) degrees on the diagonal, and the negation of the out edge weights from j in column j.

Suppose we eliminate vertex j. Then we need to add a rescaled version of row j to each row i where $\boldsymbol{L}_{j,:}$ is non-zero. Accounting for $\boldsymbol{L}_{jj} = \boldsymbol{d}_j$, this weight for row i is given by $\frac{\boldsymbol{w}_{j\to i}}{\boldsymbol{d}_j}$, and the corresponding decrease in entry j,k is then

$$\frac{\boldsymbol{w}_{j\to i}\boldsymbol{w}_{k\to j}}{\boldsymbol{d}_j}.$$

In other words, when eliminating vertex j, we add an edge $k \to i$ for each triple of vertices $k \to j \to i$, with weight given by above.

The effect of this elimination on the vector b can also be described through this 'distribution' of row j onto its out-neighbors. However, to start with, it's useful to focus on what happens to the matrix. The key observation for elimination based Laplacian solvers is that this new matrix remains a graph. In fact, it can be checked that this process exactly preserves the in and out degrees of all neighbors of i, so the graph also remains Eulerian.

However, without additional assumptions on the non-zero structures such as separators, directly performing the above process takes $O(n^3)$ time: the newly added entries quickly increases the density of the matrix until each row has $\Theta(n)$ entries. So the starting point of elimination based Laplacian solvers is to address the following two problems:

(1) Keeping the results of elimination sparse.
(2) Find vertices that are easy to eliminate.

3.1 Block Cholesky

One possible solution to the issue above is to directly sample the edges formed after eliminating each vertex. It leads to sparsified/incomplete Cholesky factorization based algorithms [6, 11], including the first nearly-linear time solver for directed Laplacians.

Our algorithm is based on eliminating blocks of vertices, and is closest to the algorithm from [10]. It aims to eliminate a block of $\Omega(n)$ vertices simultaneously. This subset, which we denote using F, is chosen to be almost independent. That is, F is picked so that each vertex in F has at least a constant portion of its out-degree going to $V \setminus F$, which we denote as C.

This property means that any random walk on F exits it with constant probability. This intuition, when viewed from iterative methods perspective, implies that power method allows rapid simulation of elimination onto $C = V \setminus F$. From a matrix perspective, it means these matrices are well-approximated by their diagonal. So subproblem $\boldsymbol{L}_{FF}^{-1}\boldsymbol{b}$ can be solved to high accuracy in $O(\log n)$ iterations via power method. We formalize the guarantees of such procedures, PR $(\cdot)$ in Lemma 6.5 in Section 6.2.

```
Algorithm 1: PRECONDITION $\left(\left\{\widetilde{S}^{(i)}, F_i\right\}_{i=1}^{d}, \boldsymbol{x}, N\right)$
Input: $\left(\alpha, \beta, \{\delta_i\}_{i=1}^{d}\right)$-Schur complement chain
       $\left\{\left\{\widetilde{S}^{(i)}\right\}_{i=1}^{d}, \{F_i\}_{i=1}^{d}\right\}$; vector $\boldsymbol{x} \in \mathbb{R}^n$; error parameter
       $\epsilon \in (0,1)$
Output: vector $\boldsymbol{x} \in \mathbb{R}^n$
1  for $i = 1, \cdots, d-1$ do
2      $\boldsymbol{x}_{F_i} \leftarrow \mathrm{PR}\left(\widetilde{S}^{(i)}_{F_iF_i}, \boldsymbol{x}_{F_i}, \mathbf{Diag}\left(\widetilde{S}^{(i)}\right)^{-1}_{F_iF_i}, \frac{1}{2}, N\right)$;
3      $\boldsymbol{x}_{C_i} \leftarrow \boldsymbol{x}_{C_i} - \widetilde{S}^{(i)}_{C_iF_i}\boldsymbol{x}_{F_i}$
4  end
5  $\boldsymbol{x}_{F_d} \leftarrow \left(\widetilde{S}^{(d)}\right)^{\dagger}\boldsymbol{x}_{F_d}$;
6  for $i = d-1, \cdots, 1$ do
7      $\boldsymbol{x}_{F_i} \leftarrow \boldsymbol{x}_{F_i} - \mathrm{PR}\left(\widetilde{S}^{(i)}_{F_iF_i}, \widetilde{S}^{(i)}_{F_iC_i}\boldsymbol{x}_{C_i}, \mathbf{Diag}\left(\widetilde{S}^{(i)}\right)^{-1}_{F_iF_i}, \frac{1}{2}, N\right)$
8  end
9  Let $\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{\mathbf{1}^\top\boldsymbol{x}}{n}\cdot\mathbf{1}$;
10 Return $\boldsymbol{x}$
```

Compared to single-vertex elimination schemes, block elimination has the advantage of having less error accumulation. Single elimination can be viewed as eliminating 1 vertex per step, while we will show that the block method eliminates $\Omega(n)$ vertices in $O(\log\log n)$ steps. This smaller number of dependencies in turn provides us the ability to bound errors more directly.

Formally, given the partition $F, C \subseteq [n]$ with the permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{P}\boldsymbol{L}\boldsymbol{P}^\top = \begin{bmatrix}\boldsymbol{L}_{FF} & \boldsymbol{L}_{FC}\\ \boldsymbol{L}_{CF} & \boldsymbol{L}_{CC}\end{bmatrix}$, the block Cholesky factorization of $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is given as

$$\boldsymbol{L} = \boldsymbol{P}^\top\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{L}_{CF}\boldsymbol{L}_{FF}^{-1} & \boldsymbol{I}\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{L}_{FF} & \boldsymbol{0}\\ \boldsymbol{0} & \mathrm{Sc}(\boldsymbol{L}, F)\end{bmatrix}\cdot\begin{bmatrix}\boldsymbol{I} & \boldsymbol{L}_{FF}^{-1}\boldsymbol{L}_{FC}\\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{P}.$$

Using the above factorization iteratively (with sparsification) generates a Schur complement chain $\left\{\widetilde{S}^{(i)}, F_i\right\}_{i=1}^{d}$ (Definition 6.1), the solver algorithm loops through these and solves the subproblems via the projection / prologation maps which are defined via the random walk on F_i respectively. Its pseudocode is given in Algorithm 1 for completeness.

We remark that the iteration numbers of the preconditioned Richardson iterations in Algorithm 1 can differ with each other. Here, we set a uniform N merely for simplicity. If we have unlimited (or quadratic) precomputation power, the method described above suffices to give us a fast solver. However, due to the exact Schur Complements being dense, the major remaining difficulty is to efficiently compute an approximate Schur complement.

3.2 Schur Complement Sparsification via Partial Block Elimination

The main bottleneck toward an efficient algorithm is the fast construction of approximate Schur complements. We will give such an algorithm whose running time is close to those of sparsification primitives via a partial block elimination process.

In simple terms, a step of this process squares the (F, F) block. Repeating this gives quadratic convergence. With α about $O(1)$, $O(\log\log n)$ iterations suffice, so the resulting error is easier to control than martingales.

Lemma 3.1 ([16]). *For any diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{D} - \boldsymbol{A}$ nonsingular, we have*

$$\begin{aligned}&(\boldsymbol{D}-\boldsymbol{A})^{-1}\\ =&\frac{1}{2}\boldsymbol{D}^{-1}+\frac{1}{2}\left(\boldsymbol{I}+\boldsymbol{D}^{-1}\boldsymbol{A}\right)\left(\boldsymbol{D}-\boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{-1}\left(\boldsymbol{I}+\boldsymbol{A}\boldsymbol{D}^{-1}\right).\end{aligned} \tag{1}$$

The main identity (1) gives rise to our definition for partial-block-eliminated Laplacian of $\boldsymbol{L}$. Let $\boldsymbol{D} = \mathbf{Diag}(\boldsymbol{L})$ and $\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L}$, let $\boldsymbol{P}$ be the permutation matrix such that $\boldsymbol{PLP}^\top = \begin{bmatrix}\boldsymbol{L}_{FF} & \boldsymbol{L}_{FC}\\ \boldsymbol{L}_{CF} & \boldsymbol{L}_{CC}\end{bmatrix}$. The partial-block-eliminated operator $\boldsymbol{\Phi}(\boldsymbol{L}|\boldsymbol{D}_{FF}, F)$ is defined as

$$\boldsymbol{\Phi}(\boldsymbol{L}|\boldsymbol{D}_{FF}, F) = \boldsymbol{P}^\top\begin{bmatrix}\boldsymbol{L}_{FF} & -\boldsymbol{A}_{FC}\\ -\boldsymbol{A}_{CF} & 2\boldsymbol{L}_{CC}\end{bmatrix}\boldsymbol{P} - \boldsymbol{A}_{:,F}\boldsymbol{D}_{FF}^{-1}\boldsymbol{A}_{F,:}.$$

We define the first exact partial-block-elimination of $\boldsymbol{L}$ by $\boldsymbol{L}^{(1)} = \boldsymbol{\Phi}(\boldsymbol{L}|\boldsymbol{D}_{FF}, F)$. Then, $\frac{1}{2}\boldsymbol{L}^{(1)}$ is an Eulerian Laplacian which has the same Schur complement of F in $\boldsymbol{L}$, i.e.,

$$\mathrm{Sc}(\boldsymbol{L}, F) = \frac{1}{2}\mathrm{Sc}\left(\boldsymbol{L}^{(1)}, F\right).$$

The 2-nd to the K-th partially-block-eliminated Laplacians are defining iteratively by $\boldsymbol{L}^{(k+1)} = \boldsymbol{\Phi}\left(\boldsymbol{L}^{(k)}|\boldsymbol{D}_{FF}, F\right)$. $\boldsymbol{L}^{(k)}$ can also be regarded as a partially powered matrix of $\boldsymbol{L}$, which uses the powering to obtain better spectral properties. Specifically, when we focus on the (F, F) block of $\boldsymbol{L}^{(k)}$, it is easy to see that $\left\|\boldsymbol{D}_{FF}^{-1}\boldsymbol{A}_{FF}^{(k)}\right\|_\infty$ converges at a quadratic rate, where $\boldsymbol{A}_{FF}^{(k)} = \boldsymbol{D}_{FF} - \boldsymbol{L}_{FF}^{(k)}$. Formal construction of the k-th partially-block-eliminated Laplacians and their properties can be found in the appendix of the full version.

To encounter the increasing density of $\boldsymbol{L}^{(k)}$, sparsification black-boxes in Section 2.3 are naturally accompanied with the partial block elimination to yield a Schur complement sparsification method (Algorithm 2). Our algorithm is essentially a directed variant of the one from [10]. A slight difference is that in the last step, we need to fix the degree discrepancies caused by approximating the strongly RCDD matrix by its diagonal.

The running time of Algorithm 2 is shown in Theorem 3.3. The k-th iterand $\widetilde{\boldsymbol{L}}^{(k)}$ of Algorithm 2 is termed the approximate k-th partially-block-eliminated Laplacian, while its exact version is just the (exact) k-th partially-block-eliminated Laplacian $\boldsymbol{L}^{(k)}$ defined above. To show the running time of Algorithm 2, the most important thing is to provide relatively tight bounds for the difference $\widetilde{\boldsymbol{L}}^{(k)} - \boldsymbol{L}^{(k)}$.

Algorithm 2: SparseSchur $(\boldsymbol{L}, F, \delta)$

```
Input: strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$;
       partition $F, C$ of $[n]$; error parameter $\delta \in (0,1)$
Output: Sparse approximate Schur complement $\boldsymbol{S}$
1  If $\mathbf{nnz}(\boldsymbol{L}) \geq O(\mathcal{N}_S(n,\delta))$, call OraSparseLaplacian to
     sparsify $\boldsymbol{L}$ with error parameter $O(\delta)$ ;
2  Find a permutation matrix $\boldsymbol{P}$ such that
     $\boldsymbol{PLP}^\top = \begin{bmatrix}\boldsymbol{L}_{FF} & \boldsymbol{L}_{FC}\\ \boldsymbol{L}_{CF} & \boldsymbol{L}_{CC}\end{bmatrix}$ ;
3  Set $K \leftarrow O\left(\log\log\frac{n}{\delta}\right)$, $\epsilon \leftarrow O\left(\frac{\delta}{K}\right)$, $\widetilde{\boldsymbol{L}}^{(0)} \leftarrow \boldsymbol{L}$,
     $\boldsymbol{D} \leftarrow \mathbf{Diag}\left(\widetilde{\boldsymbol{L}}^{(0)}\right)$, $\widetilde{\boldsymbol{A}}^{(0)} \leftarrow \boldsymbol{D} - \widetilde{\boldsymbol{L}}^{(0)}$ ;
4  for $k = 1, \cdots, K$ do
5      for $i \in F$ do
6          $\widetilde{\boldsymbol{Y}}^{(k,i)} \leftarrow$
             SparseProductFC$\left(\widetilde{\boldsymbol{A}}^{(k-1)}_{:,i}, \left(\widetilde{\boldsymbol{A}}^{(k-1)}_{i,:}\right)^\top, \epsilon, F\right)$
7      end
8      Let $\widetilde{\boldsymbol{Y}}^{(k)} \leftarrow \sum_{i\in F}\frac{1}{\boldsymbol{D}_{ii}}\widetilde{\boldsymbol{Y}}^{(k,i)}$,
         $\widetilde{\boldsymbol{L}}^{(k,0)} \leftarrow \boldsymbol{P}^\top\begin{bmatrix}\boldsymbol{D}_{FF} & -\widetilde{\boldsymbol{A}}^{(k-1)}_{FC}\\ -\widetilde{\boldsymbol{A}}^{(k-1)}_{CF} & 2\widetilde{\boldsymbol{L}}^{(k-1)}_{CC}\end{bmatrix}\boldsymbol{P} - \widetilde{\boldsymbol{Y}}^{(k)}$ ;
9      $\widetilde{\boldsymbol{L}}^{(k)} \leftarrow$ SparseEulerianFC$\left(\widetilde{\boldsymbol{L}}^{(k,0)}, \epsilon, F\right)$ and
         $\widetilde{\boldsymbol{A}}^{(k)} \leftarrow \boldsymbol{P}^\top\begin{bmatrix}\boldsymbol{D}_{FF} & \\ & \mathbf{Diag}\left(\widetilde{\boldsymbol{L}}^{(k)}_{CC}\right)\end{bmatrix}\boldsymbol{P} - \widetilde{\boldsymbol{L}}^{(k)}$
10 end
11 for $i \in F$ do
12     $\widetilde{\boldsymbol{X}}^{(i)} \leftarrow$ SparseProduct$\left(\widetilde{\boldsymbol{A}}^{(K)}_{C,i}, \left(\widetilde{\boldsymbol{A}}^{(K)}_{i,C}\right)^\top, \epsilon\right)$
13 end
14 Let $\widetilde{\boldsymbol{X}} \leftarrow \sum_{i\in F}\frac{1}{\boldsymbol{D}_{ii}}\widetilde{\boldsymbol{X}}^{(i)}$ and $\widetilde{\boldsymbol{S}}^{(0)} \leftarrow \frac{1}{2^K}\left(\widetilde{\boldsymbol{L}}^{(K)}_{CC} - \widetilde{\boldsymbol{X}}\right)$ ;
15 Compute a patching matrix $\boldsymbol{R} \in \mathbb{R}^{|C|\times|C|}$ with
     $\boldsymbol{R}_{2:|C|,1} = -\widetilde{\boldsymbol{S}}^{(0)}_{2:|C|,:}\mathbf{1}$, $\boldsymbol{R}_{1,2:|C|} = -\mathbf{1}^\top\widetilde{\boldsymbol{S}}^{(0)}_{:,2:|C|}$,
     $\boldsymbol{R}_{1,1} = -\boldsymbol{R}_{1,2:|C|}\mathbf{1} - \mathbf{1}^\top\boldsymbol{R}_{2:|C|,1} - \mathbf{1}^\top\widetilde{\boldsymbol{S}}^{(0)}\mathbf{1}$, and $\boldsymbol{R}_{ij} = 0$ for
     $i \neq 1$ and $j \neq 1$ ;
16 Set $\widehat{\boldsymbol{S}} = \widetilde{\boldsymbol{S}}^{(0)} + \boldsymbol{R}$ ;
17 Return $\boldsymbol{S} =$ OraSparseLaplacian$\left(\widehat{\boldsymbol{S}}, \delta/8\right)$
```

Remark 3.2. This permutation matrix $\boldsymbol{P}$ defined on line 2 of Algorithm 2 is only used to simplify the pseudocodes. We use the same $\boldsymbol{D}_{FF}$ in each iteration to simplify our analysis. It is possible to replace $\boldsymbol{D}_{FF}$ by $\mathbf{Diag}\left(\widetilde{\boldsymbol{L}}^{(k-1)}\right)_{FF}$ in iterations k and achieve similar running time.

Theorem 3.3. *(Schur complement sparsification) For a strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, let F, C be a partition of $[n]$ such that $\boldsymbol{L}_{FF}$ is α-RCDD ($\alpha = O(1)$) and let $\delta \in (0,1)$ be an error parameter, the subroutine SparseSchur (Algorithm 2) runs in time*

$$O\left(\mathcal{T}_{\mathrm{S}}(m,n,\delta)\right) + \tilde{O}\left(\mathcal{T}_{\mathrm{S}}\left(\mathcal{N}_{\mathrm{S}}(n,\delta)\,\delta^{-2}, n, \delta\right)\log n\right)$$

to return with high probability a strongly connected Eulerian Laplacian $\boldsymbol{S}$ satisfying $\boldsymbol{nnz}(\boldsymbol{S}) = O\left(\mathcal{N}_{\mathrm{S}}(|C|,\delta)\right)$ and

$$\boldsymbol{S} - \mathit{Sc}(\boldsymbol{L}, F) \overset{\mathrm{asym}}{\leq} \delta \cdot \boldsymbol{U}\left[\mathit{Sc}(\boldsymbol{L},F)\right].$$

Compared to the undirected analog from [10], powered directed matrices exhibit significantly more complicated spectral structures. To analyze them, we develop new interpretations of directed Schur complements based on matrix extensions.

3.3 Bounding Error Accumulations in Partially-Eliminated Laplacians by Augmented Matrices

When considering the approximate partial block elimination, in one update step, not only new sparsification errors are added into $\widetilde{\boldsymbol{L}}^{(k)}$, the errors accumulated from previous steps will multiply with each other and get possibly amplified. In addition, error accumulations in Schur complements of directed Laplacians are not as straightforward as their undirected counterparts. It's not the case that for two directed Eulerian Laplacians with the same undirectifiation, the undirectification of their Schur complements are the same. For instance, consider the undirected vs. directed cycle,eliminated till only two originally diametrically opposite vertices remain. The former has a Schur complement that has weight $2/n$, while the latter has a Schur complement that has weight 1.

By the definition of *ϵ-asymmetric approximation*, we need to essentially show the following inequality in order to obtain the approximations needed for a nearly-linear time algorithm:

$$\begin{aligned}\frac{1}{2^k}\boldsymbol{U}\left[\boldsymbol{L}^{(k)}\right] &= \frac{1}{2^k}\boldsymbol{U}\left[\Phi\left(\boldsymbol{L}^{(k-1)}|\boldsymbol{D}_{FF}, F\right)\right]\\ &\leq O(1)\cdot \boldsymbol{U}\left[\boldsymbol{L}\right],\ \forall 1\leq k\leq K.\end{aligned}$$

Here significant difficulties arise due to the already complicated formula of $\boldsymbol{L}^{(k)}$. So we instead express the exact and approximate partial block elimination as Schur complements of large augmented matrices introduced below.

In the remainder of Section 3 and the entire Section 4 and Section 5, unless otherwise specified, we assume $C = \{1, 2, \cdots, |C|\}$ for simplicity. We define

$$F_a = \left\{b \in \mathbb{Z}:\ |C| + (a-1)|F| + 1 \leq b \leq |C| + a|F|\right\}.$$

Note that in our notation, $F = F_1$.

3.3.1 *A Reformulation for Partial Block Elimination .*

For the exact and approximate k-th partially-block-eliminated matrices $\boldsymbol{L}^{(k)}$, $\widetilde{\boldsymbol{L}}^{(k)}$, we define augmented matrices $\boldsymbol{\mathcal{M}}^{(0,k)}$, $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ of size $2^k|F| + |C|$. We start with the construction of a desirable $\boldsymbol{\mathcal{M}}^{(0,k)}$. To this end, we define a sequence of augmented matrices $\left\{\boldsymbol{\mathcal{M}}^{(i,k)}\right\}_{i=0}^{k}$, where $\boldsymbol{\mathcal{M}}^{(k,k)} = \boldsymbol{L}^{(k)}$ and each $\boldsymbol{\mathcal{M}}^{(i,k)}$ is a Schur complement of $\boldsymbol{\mathcal{M}}^{(i-1,k)}$. Here we only give an informal explanation of how we construct $\boldsymbol{\mathcal{M}}^{(i,k)}$. The formal definitions of these augmented matrices are given in Section 4. To begin with, for some fixed $k \in [K]$, we define $\boldsymbol{\mathcal{M}}^{(k,k)} \overset{\mathrm{def}}{=} \boldsymbol{L}^{(k)}$.

Next, we take $\boldsymbol{\mathcal{M}}^{(k-1,k)}$ and $\boldsymbol{\mathcal{M}}^{(k-2,k)}$ as examples to show how we define such a sequence of matrices $\boldsymbol{\mathcal{M}}^{(k-1,k)}, \cdots, \boldsymbol{\mathcal{M}}^{(0,k)}$.

Define $\boldsymbol{\mathcal{M}}^{(k-1,k)}$ as follows

$$\boldsymbol{\mathcal{M}}^{(k-1,k)} \overset{\mathrm{def}}{=} \begin{bmatrix} 2\boldsymbol{L}_{CC}^{(k-1)} & -\boldsymbol{A}_{CF}^{(k-1)} & -\boldsymbol{A}_{CF}^{(k-1)} \\ -\boldsymbol{A}_{FC}^{(k-1)} & \boldsymbol{D}_{FF} & -\boldsymbol{A}_{FF}^{(k-1)} \\ -\boldsymbol{A}_{FC}^{(k-1)} & -\boldsymbol{A}_{FF}^{(k-1)} & \boldsymbol{D}_{FF} \end{bmatrix}$$

Then, it follows by direct calculations that

$$\mathrm{Sc}\left(\boldsymbol{\mathcal{M}}^{(k-1,k)}, F_2\right) = \boldsymbol{L}^{(k)}.$$

Next, we define $\boldsymbol{\mathcal{M}}^{(k-2,k)}$ as follows

$$\begin{aligned}&\boldsymbol{\mathcal{M}}^{(k-2,k)}\\ &\overset{\mathrm{def}}{=} \begin{bmatrix} 4\boldsymbol{L}_{CC}^{(k-2)} & -\boldsymbol{A}_{CF}^{(k-2)} & -\boldsymbol{A}_{CF}^{(k-2)} & -\boldsymbol{A}_{CF}^{(k-2)} & -\boldsymbol{A}_{CF}^{(k-2)} \\ -\boldsymbol{A}_{FC}^{(k-2)} & \boldsymbol{D}_{FF} & & -\boldsymbol{A}_{FF}^{(k-2)} & \\ -\boldsymbol{A}_{FC}^{(k-2)} & & \boldsymbol{D}_{FF} & & -\boldsymbol{A}_{FF}^{(k-2)} \\ -\boldsymbol{A}_{FC}^{(k-2)} & & -\boldsymbol{A}_{FF}^{(k-2)} & \boldsymbol{D}_{FF} & \\ -\boldsymbol{A}_{FC}^{(k-2)} & -\boldsymbol{A}_{FF}^{(k-2)} & & & \boldsymbol{D}_{FF} \end{bmatrix}\end{aligned}$$

It follows by direct calculations that

$$\mathrm{Sc}\left(\boldsymbol{\mathcal{M}}^{(k-2,k)}, F_3 \cup F_4\right) = \boldsymbol{\mathcal{M}}^{(k-1,k)}.$$

We will show $\frac{1}{2^k}\boldsymbol{U}\left[\boldsymbol{L}^{(k)}\right] \leq O(1)\cdot \boldsymbol{U}\left[\boldsymbol{L}\right]$ later by analyzing the properties of $\boldsymbol{\mathcal{M}}^{(0,k)}$.

We believe this representation may be of independent interest. We also remark that these augmented matrices only arise during analysis, and are not used in the algorithms.

3.3.2 *Bounding Error Accumulation in Algorithm 2 .*

Next, we will mainly use Lemma 5.2 to bound the errors after taking Schur complements. However, in our analysis, iteratively applying Lemma 5.2 to bound $\frac{1}{2^k}\left(\widetilde{\boldsymbol{L}}^{(k)} - \boldsymbol{L}^{(k)}\right)$ will lead to more $\log n$ factors in the running time. To derive a tighter bound, we introduce another group of augmented matrices $\left\{\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}\right\}$ which are defined by attaching sparsification errors to $\boldsymbol{\mathcal{M}}^{(0,k)}$. $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ can help us disentangle the sparsification errors generated from different iterations and see how these errors accumulate as we do partial block eliminations more clearly. We use another group of augmented matrices $\left\{\boldsymbol{Q}^{(k)}\right\}$ to bound the difference between $\boldsymbol{\mathcal{M}}^{(0,k)}$ and $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$. The augmented matrix $\boldsymbol{Q}^{(k)}$ is defined as the sum of a group of "reptition matrix" (Section 4.2). $\boldsymbol{Q}^{(k)}$ adopts many properties similar to $\boldsymbol{\mathcal{M}}^{(0,k)}$, so it is easy to analyze. Then, we can give tighter bound for $\widetilde{\boldsymbol{L}}^{(K)} - \boldsymbol{L}^{(K)}$

using the robustness of Schur complements in this case with the properties of $\boldsymbol{Q}^{(k)}$ (Section 5).

4 PARTIAL BLOCK ELIMINATION VIA AUGMENTED MATRICES

In this section, we introduce our augmented matrices based view of partial block elimination. As we will show later, after $O(\log\log n)$ steps of partial elimination, the (F,F) block of the approximate partially-block-eliminated Laplacian $\widetilde{\boldsymbol{L}}^{(k)}$ can be approximated by its diagonal "safely". So, what remains is to bound the error accumulations in the difference $\frac{1}{2^k}\left(\mathrm{Sc}\left(\widetilde{\boldsymbol{L}}^{(k)},F\right)-\mathrm{Sc}\left(\boldsymbol{L}^{(k)},F\right)\right)$, which we do by bounding differences in $\frac{1}{2^k}\left(\widetilde{\boldsymbol{L}}^{(k)}-\boldsymbol{L}^{(k)}\right)$.

4.1 A Reformulation of the Exact Partial Block Elimination

In this section, we provide a reformulation of the exact version of partial block elimination which is more friendly to error analysis. To be specific, our strategy is to construct a large matrix $\boldsymbol{\mathcal{M}}^{(0,k)}\in\mathbb{R}^{(2^k|F|+|C|)\times(2^k|F|+|C|)}$ such that $\boldsymbol{L}^{(k)}$ is a Schur complement of the large matrix $\boldsymbol{\mathcal{M}}^{(0,k)}$. And there is a partition of $\boldsymbol{\mathcal{M}}^{(0,k)}$ such that each block is a zero matrix or equals some submatrix of $\boldsymbol{L}$. To construct $\boldsymbol{\mathcal{M}}^{(0,k)}$, we will construct a sequence of augmented matrices $\left\{\boldsymbol{\mathcal{M}}^{(i,k)}\right\}_{i=0}^{k}$ satisfying Lemma 4.1. Later, by analyzing the large matrix $\boldsymbol{\mathcal{M}}^{(0,k)}$, we can derive tighter bounds for quantities related to $\boldsymbol{L}^{(k)}$.

Now, we give a rigorous way to construct such a sequence of matrices $\left\{\boldsymbol{\mathcal{M}}^{(i,k)}\right\}_{i=0}^{k}$ $(0\le k\le K)$.

We construct a sequence of bijections $\left\{\psi^{(i)}(\cdot)\right\}_{i=0}^{K}$ which indicate the "positions" of the blocks equalling $-\boldsymbol{A}_{FF}^{(i)}$ in the large augmented matrix $\boldsymbol{\mathcal{M}}^{(k-i,k)}$.

We start with $\psi^{(0)}(\cdot)$ and will define these $\psi^{(i)}$ iteratively. The mapping $\psi^{(0)}(\cdot)$ is defined as a trivial mapping from $\{1\}$ to $\{1\}$ with

$$\psi^{(0)}(1)=1.$$

Then, assume we have defined $\psi^{(i-1)}(\cdot)$. Now, we define $\psi^{(i)}$ as follows:

$$\psi^{(i)}(a)=\begin{cases}a+2^{i-1},\ a\in[2^{i-1}]\\ \psi^{(i-1)}\left(a-2^{i-1}\right),\ 2^{i-1}+1\le a\le 2^i\end{cases}$$

If $\psi^{(i-1)}(\cdot)$ is a bijection from $[2^{i-1}]$ to $[2^{i-1}]$, then $\psi^{(i)}(\cdot)\Big|_{\{2^{i-1}+1,\cdots,2^i\}}$ is a bijection from $\left\{2^{i-1}+1,\cdots,2^i\right\}$ to $[2^{i-1}]$. And by the definition, $\psi^{(i)}(\cdot)\Big|_{[2^{i-1}]}$ is a bijection from $[2^{i-1}]$ to $\left\{2^{i-1}+1,\cdots,2^i\right\}$. Then, $\psi^{(i)}$ is a bijection from $[2^i]$ to $[2^i]$.

It follows by induction that for any $k\in[K]$, $\psi^{(k)}(\cdot)$ is a bijection from $[2^k]$ to $[2^k]$. And by the fact that $2^{i-1}+1\le\psi^{(i)}(a)\le 2^i$ for $a\in[2^{i-1}]$ and $\psi^{(i)}(a)\in[2^{i-1}]$ for $2^{i-1}+1\le a\le 2^i$, we have the following relation

$$\psi^{(j)}(a)\neq a,\ \forall 1\le j\le K,\ a\in[2^j].$$

With the notations defined above, we define the matrix $\boldsymbol{\mathcal{M}}^{(i,k)}$ as

$$\begin{aligned}\boldsymbol{\mathcal{M}}^{(i,k)}&=2^{k-i}\mathcal{P}\left(\boldsymbol{L}_{CC}^{(i)},C,C,2^{k-i}|F|+|C|\right)\\&+\sum_{a=1}^{2^{k-i}}\Big(\mathcal{P}\left(\boldsymbol{D}_{FF},F_a,F_a,2^{k-i}|F|+|C|\right)\\&+\mathcal{P}\left(-\boldsymbol{A}_{FF}^{(i)},F_a,F_{\psi^{(k-i)}(a)},2^{k-i}|F|+|C|\right)\\&+\mathcal{P}\left(-\boldsymbol{A}_{FC}^{(i)},F_a,C,2^{k-i}|F|+|C|\right)\\&+\mathcal{P}\left(-\boldsymbol{A}_{CF}^{(i)},C,F_a,2^{k-i}|F|+|C|\right)\Big).\end{aligned}$$

where the notation $\mathcal{P}(\boldsymbol{X},A,B,n)$ has been defined in Section 2, which means putting matrix $\boldsymbol{X}$ in the submatrix indexed by (A,B) in a zero matrix $\boldsymbol{0}_{n\times n}$; $\boldsymbol{L}^{(k)}$ is the exact k-th partially-block-eliminated Laplacian and formal definitions of $\boldsymbol{L}^{(k)}$, $\boldsymbol{A}^{(k)}$ are in the appendix of the full version.

We have the following properties of $\left\{\boldsymbol{\mathcal{M}}^{(i,k)}\right\}$.

Lemma 4.1. *For any $0\le k\le K$, $0\le i\le k$, $\boldsymbol{\mathcal{M}}^{(i,k)}$ is an Eulerian Laplacian; $\boldsymbol{\mathcal{M}}^{(i,k)}_{-[n],-[n]}$, $\boldsymbol{\mathcal{M}}^{(i,k)}_{-C,-C}$ are α-RCDD; the Schur complement satisfies*

$$Sc\left(\boldsymbol{\mathcal{M}}^{(i,k)},\cup_{a=2^{k-i-1}+1}^{2^{k-i}}F_a\right)=\boldsymbol{\mathcal{M}}^{(i+1,k)}.$$

Further,

$$Sc\left(\boldsymbol{\mathcal{M}}^{(i,k)},-[n]\right)=\boldsymbol{L}^{(k)}.$$

In addition, for any $\boldsymbol{x}\in\mathbb{R}^n$, let

$$\widehat{\boldsymbol{x}}=(\boldsymbol{x}_C^\top\quad\underbrace{\boldsymbol{x}_F^\top\ \cdots\ \boldsymbol{x}_F^\top}_{2^k\text{ repetitions of }\boldsymbol{x}_F^\top}\quad)^\top,$$

then,

$$\widehat{\boldsymbol{x}}^\top\boldsymbol{\mathcal{M}}^{(0,k)}\widehat{\boldsymbol{x}}=2^k\boldsymbol{x}^\top\boldsymbol{L}\boldsymbol{x}.$$

The following lemma answers a question in Section 3.3. That is, $\frac{1}{2^k}\boldsymbol{U}\left[\boldsymbol{L}^{(k)}\right]\le O(1)\,\boldsymbol{U}[\boldsymbol{L}]$.

Lemma 4.2. *For any $0\le k\le K$,*

$$\frac{1}{2^k}\boldsymbol{U}\left[\boldsymbol{L}^{(k)}\right]\le\left(3+\frac{2}{\alpha}\right)\boldsymbol{U}[\boldsymbol{L}].$$

4.2 Bounding Error Accumulation Using Repetition Matrices

In this section, we define $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ as an inexact version of $\boldsymbol{\mathcal{M}}^{(0,k)}$ where sparsification errors accumulate. Then, we introduce a special type of augmented matrices, which we term *reptition matrices*, and bound the difference $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}-\boldsymbol{\mathcal{M}}^{(0,k)}$ in norms based on the matrices $\left\{\boldsymbol{Q}^{(k)}\right\}$ which are defined as linear combinations of some *reptition matrices* of $\left\{\boldsymbol{\mathcal{M}}^{(0,k)}\right\}$.

Before we define $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$, we introduce our notations for the errors induced by sparsification:

$$\boldsymbol{E}^{(k)} = \widetilde{\boldsymbol{L}}^{(k)} - \Phi\left(\widetilde{\boldsymbol{L}}^{(k-1)} | \boldsymbol{D}_{FF}, F\right),$$

$$\boldsymbol{E}_{\boldsymbol{X}} = \widetilde{\boldsymbol{A}}_{CF}^{(K)} \boldsymbol{D}_{FF}^{-1} \widetilde{\boldsymbol{A}}_{FC}^{(K)} - \widetilde{\boldsymbol{X}}.$$

We also denote in the rest of this paper,

$$\widehat{\boldsymbol{R}} = \boldsymbol{R} + \frac{1}{2^K} \widetilde{\boldsymbol{A}}_{CF}^{(K)} \left(\boldsymbol{D}_{FF} - \widetilde{\boldsymbol{A}}_{FF}^{(K)}\right)^{-1} - \frac{1}{2^K} \widetilde{\boldsymbol{A}}_{CF}^{(K)} \boldsymbol{D}_{FF}^{-1} \widetilde{\boldsymbol{A}}_{FC}^{(K)}.$$

Some elementary facts of the results of Algorithm 2 is given by the following lemma.

Lemma 4.3. *With high probability, the following statements hold:*

(1) $\left\{\widetilde{\boldsymbol{L}}^{(k)}\right\}_{k=0}^{K}$ *are Eulerian Laplacians;*

(2) $\left\{\widetilde{\boldsymbol{A}}^{(k)}\right\}_{k=0}^{K}$ *are nonnegative matrices satisfying*

$$\left\|\boldsymbol{D}_{FF}^{-1} \widetilde{\boldsymbol{A}}_{FF}^{(k)}\right\|_{\infty} \leq \left(\frac{1}{1+\alpha}\right)^{2^k}, \ \forall 0 \leq k \leq K;$$

(3) $\boldsymbol{S}, \widehat{\boldsymbol{S}}$ *are Eulerian Laplacians;*

(4) *The matrix* $\widehat{\boldsymbol{R}}$ *satisfies* $\widehat{\boldsymbol{R}}\boldsymbol{1} = \widehat{\boldsymbol{R}}^{\top}\boldsymbol{1} = \boldsymbol{0}$ *and*

$$\left\|\widehat{\boldsymbol{R}}\right\|_2 \leq \frac{n^2 \|\boldsymbol{D}_{FF}\|_2}{2^{K-1}\alpha} \left(\frac{1}{1+\alpha}\right)^{2^K}.$$

Then, we can provide bounds for the one-step errors in the next lemma.

Lemma 4.4. *The error matrices satisfies*

$$\boldsymbol{E}^{(k)} \overset{\text{asym}}{\preceq} \epsilon_0 \boldsymbol{U}\left[\widetilde{\boldsymbol{L}}^{(k-1)}\right], \tag{2}$$

$$\boldsymbol{E}_{\boldsymbol{X}} \overset{\text{asym}}{\preceq} \epsilon \boldsymbol{U}\left[Sc\left(\widetilde{\boldsymbol{L}}^{(K)}, F\right)\right], \tag{3}$$

where $\epsilon_0 = 2\left(3 + \frac{2}{\alpha}\right)\left(2\epsilon + \epsilon^2\right)$.

In the remainder of this paper, we write $\epsilon_0 = 2\left(3 + \frac{2}{\alpha}\right)\left(2\epsilon + \epsilon^2\right)$.

Recall that we define an augmented matrix $\boldsymbol{\mathcal{M}}^{(0,k)}$ such that $\boldsymbol{L}^{(k)}$ is its Schur complement in Section 4.1. Now, we define $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ which is an inexact version of $\boldsymbol{\mathcal{M}}^{(0,k)}$ to analyze the properties of $\widetilde{\boldsymbol{L}}^{(k)}$. We first define

$$\begin{aligned}\mathcal{R}\left(k, a, \boldsymbol{E}^{(i)}\right) = & \mathcal{P}\left(\boldsymbol{E}_{FF}^{(i)}, F_a, F_{\psi^{(k-i)}(a)}, 2^{k-i}|F| + |C|\right) \\ & + \mathcal{P}\left(\boldsymbol{E}_{FC}^{(i)}, F_a, C, 2^{k-i}|F| + |C|\right) \\ & + \mathcal{P}\left(\boldsymbol{E}_{CF}^{(i)}, C, F_{\psi^{(k-i)}(a)}, 2^{k-i}|F| + |C|\right) \\ & + \mathcal{P}\left(\boldsymbol{E}_{CC}^{(i)}, C, C, 2^{k-i}|F| + |C|\right).\end{aligned}$$

Then, we define the error matrices

$$\boldsymbol{\mathcal{E}}^{(i,k)} = \sum_{a=1}^{2^{k-i}} \mathcal{R}\left(k, a, \boldsymbol{E}^{(i)}\right)$$

and

$$\boldsymbol{\mathcal{E}}^{(1:k,k)} = \sum_{i=1}^{k} \boldsymbol{\mathcal{E}}^{(i,k)}.$$

The matrix $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ is defined as follows

$$\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)} = \boldsymbol{\mathcal{M}}^{(0,k)} + \boldsymbol{\mathcal{E}}^{(1:k,k)}.$$

Lemma 4.5. *The Schur complement of* $[2^k|F| + |C|] \backslash [n]$ *in* $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ *satisfies:*

$$Sc\left(\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}, -[n]\right) = \widetilde{\boldsymbol{L}}^{(k)}.$$

To help bound $\boldsymbol{\mathcal{E}}^{(1:k,k)}$, we define some special kinds of matrices termed *"repetition matrices"*. We will construct the matrices $\left\{\boldsymbol{Q}^{(k)}\right\}$ as linear combinations of *"repetition matrices"*.

Definition 4.6. ("Repetition matrices") We will use the following 3 kinds of *"repetition matrices"*: consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ and subset $C \subseteq [m]$ and $E = [m] \backslash C$,

(1) the *k-"repetition matrix"* of $\boldsymbol{A}$ is defined as follows:

$$\operatorname{Rep}(k, C, \boldsymbol{A}) = \begin{bmatrix} k\boldsymbol{A}_{CC} & \boldsymbol{A}_{CE} & \boldsymbol{A}_{CE} & \cdots & \boldsymbol{A}_{CE} \\ \boldsymbol{A}_{EC} & \boldsymbol{A}_{EE} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}_{EC} & \boldsymbol{0} & \boldsymbol{A}_{EE} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{A}_{EC} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{A}_{EE} \end{bmatrix},$$

where the repetition numbers of the blocks $\boldsymbol{A}_{CE}, \boldsymbol{A}_{EC}, \boldsymbol{A}_{EE}$ are k;

(2) $\operatorname{Rep}^{+0}(k, C, \boldsymbol{A}, N)$ is a larger matrix by appending all-zeros rows and columns to $\operatorname{Rep}(k, C, \boldsymbol{A})$:

$$\operatorname{Rep}^{+0}(k, C, \boldsymbol{A}, N) = \begin{bmatrix} \operatorname{Rep}(k, C, \boldsymbol{A}) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{N \times N},$$

where $N \geq k|E| + |C|$ is used to indicate the size of the matrix $\operatorname{Rep}^{+0}(k, C, \boldsymbol{A}, N)$;

(3) if F, F_+ is a partition of E, $\operatorname{Rep}(k, C, F, \boldsymbol{A})$ is defined as a permutation of the *k-"repetition matrix"* of $\boldsymbol{A}$, which has the following form:

$$\operatorname{Rep}(k, C, F, \boldsymbol{A}) = \begin{bmatrix} k\boldsymbol{A}_{CC} & \boldsymbol{A}_{CF} & \cdots & \boldsymbol{A}_{CF} & \boldsymbol{A}_{CF^+} & \cdots & \boldsymbol{A}_{CF^+} \\ \boldsymbol{A}_{FC} & \boldsymbol{A}_{FF} & & & & & \\ \vdots & & \ddots & & & & \\ \boldsymbol{A}_{FC} & & & \boldsymbol{A}_{FF} & & & \\ \boldsymbol{A}_{F^+C} & & & & \boldsymbol{A}_{F^+F^+} & & \\ \vdots & & & & & \ddots & \\ \boldsymbol{A}_{F^+C} & & & & & & \boldsymbol{A}_{F^+F^+} \end{bmatrix}.$$

Now, we define the matrices $\left\{\boldsymbol{Q}^{(k)}\right\}_{0 \leq k \leq K}$ which are used to bound $\boldsymbol{\mathcal{M}}^{(0,k)} - \widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ and then $\boldsymbol{L}^{(k)} - \widetilde{\boldsymbol{L}}^{(k)}$. We define the matrices $\left\{\boldsymbol{Q}^{(k)}\right\}_{0 \leq k \leq K}$ iteratively together with the error quantities $\{\gamma_k\}_{0 \leq k \leq K}$.

We start from $\boldsymbol{Q}^{(0)} = \boldsymbol{U}[\boldsymbol{L}]$ and $\gamma_0 = 0$. If we have defined $\left\{\boldsymbol{Q}^{(i)}\right\}_{0 \le i < k}$, $\{\gamma_i\}_{0 \le i < k}$, then $\boldsymbol{Q}^{(k)} \in \mathbb{R}^{(2^k|F|+|C|)\times(2^k|F|+|C|)}$ and $\gamma_k \in \mathbb{R}_+$ are defined as follows:

$$\begin{aligned}\boldsymbol{Q}^{(k)} &\overset{\text{def}}{=} \frac{k}{4k + \frac{2k}{\alpha} + \sum_{i=0}^{k-1}\gamma_i}\boldsymbol{U}\left[\boldsymbol{\mathcal{M}}^{(0,k)}\right] \\ &+ \frac{1}{4k + \frac{2k}{\alpha} + \sum_{i=0}^{k-1}\gamma_i}\sum_{i=0}^{k-1}\gamma_i \mathrm{Rep}\left(2^{k-i}, C, F, \boldsymbol{Q}^{(i)}\right) \\ &+ \frac{3 + \frac{2}{\alpha}}{4k + \frac{2k}{\alpha} + \sum_{i=0}^{k-1}\gamma_i}\sum_{i=0}^{k-1}\mathrm{Rep}\left(2^{k-i}, C, F, \boldsymbol{U}\left[\boldsymbol{\mathcal{M}}^{(0,i)}\right]\right),\end{aligned}$$

$$\gamma_k \overset{\text{def}}{=} \left\|\mathrm{Sc}\left(\boldsymbol{Q}^{(k)}, -[n]\right)^{\dagger/2}\left(\widetilde{\boldsymbol{L}}^{(k)} - \boldsymbol{L}^{(k)}\right)\mathrm{Sc}\left(\boldsymbol{Q}^{(k)}, -[n]\right)^{\dagger/2}\right\|_2.$$

The following lemma shows some elementary properties of $\boldsymbol{Q}^{(k)}$.

Fact 4.7. $\boldsymbol{Q}^{(k)}$ is a Laplacian satisfying:

(1) $\boldsymbol{U}\left[\boldsymbol{\mathcal{M}}^{(0,k)}\right] \preceq \left(4 + \frac{2}{\alpha} + \frac{\sum_{i=0}^{k-1}\gamma_i}{k}\right)\boldsymbol{Q}^{(k)}$;

(2) $\mathbf{Diag}\left(\boldsymbol{Q}^{(k)}\right) = \mathbf{Diag}\left(\boldsymbol{\mathcal{M}}^{(0,k)}\right)$;

(3) $\boldsymbol{Q}^{(k)}_{-C,-C}$, $\boldsymbol{Q}^{(k)}_{-[n],-[n]}$ are α-RCDD;

(4) $\left\|\left(\boldsymbol{Q}^{(k)}\right)^{1/2}\mathbf{Diag}\left(\boldsymbol{\mathcal{M}}^{(0,k)}\right)^{-1/2}\right\|_2^2 \le 2$;

Lemma 4.8. $\mathit{Sc}\left(\boldsymbol{Q}^{(k)}, -C\right) \preceq 2^k\boldsymbol{U}\left[\mathit{Sc}\left(\boldsymbol{L}, F\right)\right]$.

The following lemma shows that the sparsification errors attached to $\widetilde{\boldsymbol{\mathcal{M}}}^{(0,k)}$ can be bounded by $\boldsymbol{Q}^{(k)}$.

Lemma 4.9.

$$\boldsymbol{\mathcal{E}}^{(1:k,k)} \overset{\text{asym}}{\preceq} \epsilon_0\left(4k + \frac{2k}{\alpha} + \sum_{i=0}^{k-1}\gamma_i\right)\boldsymbol{Q}^{(k)}.$$

5 ROBUSTNESS OF SCHUR COMPLEMENTS AND FULL ERROR ANALYSIS

In this section, we show additional robustness properties of Schur complements suitable for analyzing errors on the augmented matrices. Specifically, we establish conditions on $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{U}$ where $\boldsymbol{A} - \boldsymbol{B} \overset{\text{asym}}{\preceq} \epsilon \cdot \boldsymbol{U}$, as well as the set to be eliminated, F, so that $\mathrm{Sc}(\boldsymbol{A}, F) - \mathrm{Sc}(\boldsymbol{B}, F) \overset{\text{asym}}{\preceq} \delta \cdot \mathrm{Sc}(\boldsymbol{U}, F)$.

Using these properties, we bound the norms of errors in Schur complements of the $\boldsymbol{Q}^{(k)}$ and γ_k. Such bounds allow us to complete the proof of Theorem 3.3.

The next lemma is used to prove Lemma 5.2 below.

Lemma 5.1. *Suppose that $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is an Eulerian Laplacian, $\boldsymbol{D} = \boldsymbol{Diag}(\boldsymbol{L})$, $\boldsymbol{W}$ is PSD, $\left\|\boldsymbol{W}^{1/2}\boldsymbol{D}^{-1/2}\right\|_2 \le a$, and the matrix $\boldsymbol{E} \in \mathbb{R}^{n\times n}$ satisfies $\boldsymbol{E} \overset{\text{asym}}{\preceq} b\boldsymbol{W}$ with $a^2 b < 2$. Then the matrix $\boldsymbol{M} = \boldsymbol{L} + \boldsymbol{E}$ satisfies:*

$$\boldsymbol{M}\boldsymbol{D}^{-1}\boldsymbol{M}^\top \preceq \frac{1}{2 - a^2 b}\left(\left(4 + 2a^2 b\right)\boldsymbol{U}[\boldsymbol{L}] + 2b\boldsymbol{W}\right).$$

The following lemma shows the robustness of the Schur complements. It's used in the proof of Lemma 5.3 to bound γ_k.

Lemma 5.2. *Let $\boldsymbol{N} \in \mathbb{R}^{n\times n}$ be an Eulerian Laplacian, let $\boldsymbol{M}$ be an n-by-n matrix, let $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ be PSD and F, C a partition of $[n]$. Suppose that $\boldsymbol{U}_{FF}$ is nonsingular, $\boldsymbol{U}\boldsymbol{1} = \boldsymbol{0}$, $\boldsymbol{N}_{FF}$ is ρ-RCDD ($\rho > 0$), $\boldsymbol{U}[\boldsymbol{N}]_{FF} \succeq \frac{1}{\mu}\boldsymbol{U}_{FF}$, $\boldsymbol{U}[\boldsymbol{N}] \preceq \beta\boldsymbol{U}$, $\left\|\boldsymbol{U}^{1/2}\boldsymbol{Diag}(\boldsymbol{N})^{-1/2}\right\|_2 \le a$, and the matrix $\boldsymbol{E} = \boldsymbol{M} - \boldsymbol{N}$ satisfies $\boldsymbol{E} \overset{\text{asym}}{\preceq} b \cdot \boldsymbol{U}$ with $b < \min\left\{\frac{2}{a^2}, \frac{1}{\mu}\right\}$.*

Then, $\boldsymbol{M}_{FF}$, $\boldsymbol{N}_{FF}$ are nonsingular and

$$\begin{aligned}&\mathit{Sc}(\boldsymbol{M}, F) - \mathit{Sc}(\boldsymbol{N}, F) \\ &\overset{\text{asym}}{\preceq} b\left(1 + \frac{1}{\rho}\right)\frac{\mu\left(\beta\left(4 + 2a^2 b\right) + 2b\right)}{(1 - \mu b)^2\left(2 - a^2 b\right)} \cdot \mathit{Sc}(\boldsymbol{U}, F).\end{aligned}$$

We can now obtain a relatively tight bound for $\mathrm{Sc}\left(\widetilde{\boldsymbol{L}}^{(K)}, F\right) - \mathrm{Sc}\left(\boldsymbol{L}^{(K)}, F\right)$ by bounding γ_k iteratively.

Lemma 5.3. *For any $\delta_0 \in (0, 1)$, with a small $\epsilon = O\left(\frac{\delta_0}{K}\right)$ in Algorithm 2, the exact and approximate K-th partially-block-eliminated Laplacians $\boldsymbol{L}^{(K)}, \widetilde{\boldsymbol{L}}^{(K)}$ satisfies*

$$\begin{aligned}&\frac{1}{2^K}\left(\mathit{Sc}\left(\widetilde{\boldsymbol{L}}^{(K)}, F\right) - \mathit{Sc}\left(\boldsymbol{L}^{(K)}, F\right)\right) \\ &\overset{\text{asym}}{\preceq} O(\delta_0) \cdot \boldsymbol{U}\left[\mathit{Sc}(\boldsymbol{L}, F)\right].\end{aligned} \tag{4}$$

Remark 5.4. Since $\widetilde{\boldsymbol{S}}^{(0)} = \frac{1}{2^K}\left(\widetilde{\boldsymbol{L}}^{(K)}_{CC} - \widetilde{\boldsymbol{X}}\right)$ (in Algorithm 2), the $\frac{1}{2^K}$ factor on the LHS of (4) doesn't matter. The parameter choice $\epsilon = O\left(\frac{\delta_0}{K}\right)$ implies that when running Algorithm 2, the error accumulates linearly in k rather than exponentially.

6 A NEARLY-LINEAR TIME SOLVER

In this section, we complete the Sparsified Schur Complement based algorithm by invoking the nearly-linear time Schur complement sparsification procedure derived above in Sections 4 and 5. We first call this Schur complement sparsification procedure repeatedly to construct a sparse Schur complement chain in Section 6.1. Then, in Section 6.2, we show that this Schur complement chain gives a preconditoner Precondition for the initial Eulerian Laplacian matrix. The full high accuracy solver then follows from invoking this preconditioner inside Richardson iteration.

6.1 Schur Complement Chains

We first define Schur complement chains over directed graphs, which is a variant of the Schur complement chain for undirected graphs in [10].

Definition 6.1. (Schur complement chain) Given a strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, an $\left(\alpha, \beta, \{\delta_i\}_{i=1}^d\right)$-Schur complement chain of $\boldsymbol{L}$ is a sequence of strongly connected Eulerian Laplacians and subsets $\left\{\left\{\widetilde{\boldsymbol{S}}^{(i)}\right\}_{i=1}^d, \{F_i\}_{i=1}^d\right\}$ satisfying

(1) $\{F_i\}_{i=1}^d$ is a partition of $[n]$; each $\widetilde{\boldsymbol{S}}^{(i)}$ is supported on the submatrix indexed by (C_{i-1}, C_{i-1}), where $C_i \overset{\text{def}}{=} [n]\backslash\left(\cup_{j=1}^i F_j\right)$ $(i = 0, 1, \cdots, d-1)$; $|C_i| \le (1-\beta)^i n$; $|F_d| = |C_{d-1}| = O(1)$;

(2) For $1 \le i \le d-1$, $\widetilde{\boldsymbol{S}}^{(i)}_{F_iF_i}$ is α-RCDD;

Algorithm 3: Block Cholesky solver for directed Laplacians

```
Input: strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$;
       query vectors $\left\{\boldsymbol{b}^{(q)}\right\}_{q=1}^{Q} \subseteq \mathbb{R}^n$ with each $\boldsymbol{b}^{(q)} \perp \mathbf{1}$;
       error parameters $\{\epsilon_q\}_{q=1}^{Q} \subseteq (0,1)$
Output: solutions $\left\{\boldsymbol{x}^{(q)}\right\}_{q=1}^{Q} \subseteq \mathbb{R}^n$
1 Call SchurChain $(\boldsymbol{L}, 0.25, 0.1)$ to compute a
    $\left\{0.25, 0.05, \left\{\frac{0.1}{i^2}\right\}_{i=1}^{O(\log n)}\right\}$-Schur complement chain
    $\left\{\left\{\widetilde{\boldsymbol{S}}^{(i)}\right\}_{i=1}^{d}, \{F_i\}_{i=1}^{d}\right\}$ (Sections 4, 5, 6.1)
2 Generate the operator $\boldsymbol{Z}(\boldsymbol{x}) =$
    Precondition $\left(\left\{\left\{\widetilde{\boldsymbol{S}}^{(i)}\right\}_{i=1}^{d}, \{F_i\}_{i=1}^{d}\right\}, \boldsymbol{x}, O(\log n)\right)$
    (Section 6.2) ;
3 Using the preconditioned Richardson iteration with the
    preconditioner $\boldsymbol{Z}$ to solve the Laplacian systems: for each
    query vector $\boldsymbol{b}^{(q)}$, compute
    $\boldsymbol{x}^{(q)} \leftarrow \mathrm{PR}\left(\boldsymbol{L}, \boldsymbol{b}^{(q)}, \boldsymbol{Z}(\cdot), 1, O\left(\log\left(n/\epsilon_q\right)\right)\right)$
```

(3) $\widetilde{\boldsymbol{S}}^{(1)} - \boldsymbol{L} \overset{\text{asym}}{\preceq} \delta_1 \cdot \boldsymbol{U}[\boldsymbol{L}]$ and $\widetilde{\boldsymbol{S}}^{(i+1)} - \mathrm{Sc}\left(\widetilde{\boldsymbol{S}}^{(i)}, F_i\right) \overset{\text{asym}}{\preceq} \delta_{i+1} \cdot \boldsymbol{U}\left[\mathrm{Sc}\left(\widetilde{\boldsymbol{S}}^{(i)}, F\right)\right]$, $1 \le i \le d-1$;

(4) $\boldsymbol{U}\left[\widetilde{\boldsymbol{S}}^{(1)}\right] \succeq \boldsymbol{U}[\boldsymbol{L}]$ and $\boldsymbol{U}\left[\widetilde{\boldsymbol{S}}^{(i+1)}\right] \succeq \boldsymbol{U}\left[\mathrm{Sc}\left(\widetilde{\boldsymbol{S}}^{(i)}, F_i\right)\right]$, $1 \le i \le d-1$.

We also denote $F_0 = C_d = \emptyset$, $C_0 = [n]$ for notational simplicity.

Remark 6.2. Compared with the Schur complement chains for undirected graphs from [10], the only new condition is Condition (4). It guarantees the positive semi-definiteness of the symmetrization of the sparsified approximate Eulerian Laplacian $\widetilde{\boldsymbol{L}}$ and the error-bounding matrix $\widetilde{\boldsymbol{B}}$ defined in Section 6.2.

To construct a Schur complement chain, we first use the following lemma to find an α-RCDD subset F_1, and then apply the Schur complement sparsification method SparseSchur to compute $\widetilde{\boldsymbol{S}}^{(1)}$ which is an approximation for $\mathrm{Sc}(\boldsymbol{L}, F_1)$. Then, we repeat this process to get a desirable Schur complement chain.

Lemma 6.3. *(Theorem A.1 of [6]) Given an Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ with $\boldsymbol{nnz}(\boldsymbol{L}) = m$, the routine FindRCDDBlock outputs a subset $F \subseteq [n]$ such that $|F| \ge \frac{n}{16(1+\alpha)}$ and $\boldsymbol{L}_{FF}$ is α-RCDD in time $O\left(m \log \frac{1}{p}\right)$ with probability at least $1-p$.*

By Lemma 6.3, we can choose for instance $\alpha = 0.1$. So, we assume $\alpha = O(1)$, when analyzing the complexities below. Our method to construct a Schur complement chain is illustrated in Algorithm 4. It running time is shown in Theorem 6.4.

Theorem 6.4. *Given a strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ and parameters $\alpha = O(1)$, $\delta \in (0,1]$, the routine SchurChain*

Algorithm 4: SchurChain $(\boldsymbol{L}, \alpha, \delta)$

```
Input: strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$;
       parameters $\alpha > 0$, $\delta \in (0,1]$
Output: $\left(\alpha, \frac{1}{16(1+\alpha)}, \left\{\frac{\delta}{i^2}\right\}_{i=1}^{d}\right)$-Schur complement chain
        $\left\{\left\{\widetilde{\boldsymbol{S}}^{(i)}\right\}_{i=1}^{d}, \{F_i\}_{i=1}^{d}\right\}$
1  Set $\delta'_i = \frac{\delta}{3i^2}$ for $i \ge 1$. ;
2  Compute $\boldsymbol{S}^{(1)} \leftarrow$ OraSparseLaplacian $\left(\boldsymbol{L}, \delta'_1\right)$ ;
3  Let $\widetilde{\boldsymbol{S}}^{(1)} \leftarrow \boldsymbol{S}^{(1)} + \frac{\delta'_1}{1-\delta'_1}\boldsymbol{U}\left[\boldsymbol{S}^{(1)}\right]$. ;
4  Set $i \leftarrow 0$, $C_0 = [n]$ ;
5  while $|C_i| > 100$ do
6      $i \leftarrow i+1$ ;
7      $F_i \leftarrow$ FindRCDDBlock $\left(\widetilde{\boldsymbol{S}}^{(i)}, \alpha\right)$ ;
8      $C_i \leftarrow C_{i-1} \backslash F_i$ ;
9      $\boldsymbol{S}^{(i+1)} \leftarrow$ SparseSchur $\left(\widetilde{\boldsymbol{S}}^{(i)}, F_i, \delta_{i+1}\right)$ ;
10     $\widetilde{\boldsymbol{S}}^{(i+1)} \leftarrow \boldsymbol{S}^{(i)} + \frac{\delta'_{i+1}}{1-\delta'_{i+1}}\boldsymbol{U}\left[\boldsymbol{S}^{(i)}\right]$
11 end
12 Return $\left\{\left\{\widetilde{\boldsymbol{S}}^{(i)}\right\}_{i=1}^{d}, \{F_i\}_{i=1}^{d}\right\}$
```

runs in time

$$O\left(\mathcal{T}_{\mathrm{S}}(m,n,\delta)\right) + \tilde{O}\left(\mathcal{T}_{\mathrm{S}}\left(\mathcal{N}_{\mathrm{S}}(n,\delta)\,\delta^{-2}, n, \delta\right)\log n\right)$$

with high probability to return an $\left(\alpha, \frac{1}{16(1+\alpha)}, \left\{\frac{\delta}{i^2}\right\}_{i=1}^{d}\right)$-Schur complement chain, where $d = O(\log n)$. In addition,

$$\sum_{i=1}^{d} \boldsymbol{nnz}\left(\widetilde{\boldsymbol{S}}^{(i)}\right) = O\left(\mathcal{N}_{\mathrm{S}}(n,\delta)\right).$$

6.2 Construction of the Preconditioner and the Solver

After constructing a desirable Schur complement chain, we use the Schur complement chain to construct a preconditioner and solve $\boldsymbol{Lx} = \boldsymbol{b}$ via the preconditioned Richardson iteration.

Consider a linear system $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{b}$ is in the image space of $\boldsymbol{A}$. Given a preconditioner $\boldsymbol{Z}$, the classical preconditioned Richardson iteration updates as follows:

$$\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \eta Z\left(\boldsymbol{b} - \boldsymbol{Ax}^{(k)}\right).$$

We initialize $\boldsymbol{x}^{(0)} = \boldsymbol{0}$ for simplicity. This procedure is denoted by $\boldsymbol{x}^{(N)} = \mathrm{PR}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{Z}, \eta, N)$.

We will use the following fundamental lemma to guarantee the convergence rate of the preconditioned Richardson iteration in our methods.

Lemma 6.5. *(Lemma 4.2 of [7]) Let $\boldsymbol{A}, \boldsymbol{Z}, \boldsymbol{U} \in \mathbb{R}^{n\times n}$, where $\boldsymbol{U}$ is PSD and $\ker(\boldsymbol{U}) \subseteq \ker(\boldsymbol{Z}) = \ker\left(\boldsymbol{Z}^{\top}\right) = \ker(\boldsymbol{A}) = \ker\left(\boldsymbol{A}^{\top}\right)$. Let $\boldsymbol{b} \in \mathbb{R}^n$ be a vector inside the image space of $\boldsymbol{A}$. Denote the projection onto the image space of $\boldsymbol{A}$ by $\boldsymbol{P}_{\boldsymbol{A}}$. Denote $\boldsymbol{x}^{(N)} = PR(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{Z}, \eta, N)$.*

Then, $\boldsymbol{x}^{(N)}$ satisfies

$$\left\|\boldsymbol{x}^{(N)} - \boldsymbol{A}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}} \leq \|\boldsymbol{P}_{\boldsymbol{A}} - \eta \boldsymbol{Z}\boldsymbol{A}\|_{\boldsymbol{U}\to\boldsymbol{U}}^{N} \left\|\boldsymbol{A}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}}.$$

In addition, the preconditioned Richardson iteration is a linear operator with

$$\boldsymbol{x}^{(N)} = \eta \sum_{k=0}^{N-1} (\boldsymbol{P}_{\boldsymbol{A}} - \eta \boldsymbol{Z}\boldsymbol{A})^{k} \boldsymbol{Z}\boldsymbol{b}.$$

Our construction for the preconditioner is illustrated in Algorithm 1. We analyze Algorithm 1 by representing the routine PR $(\cdot)$ as a linear operator which is equivalent to multiplying vector $\boldsymbol{x}$ with the matrix $\boldsymbol{\Pi}\widehat{\boldsymbol{Z}}$, where the definition of $\widehat{\boldsymbol{Z}} \in \mathbb{R}^{n\times n}$ is in the full version, and $\boldsymbol{\Pi} = \boldsymbol{I} - \frac{\boldsymbol{1}\boldsymbol{1}^{\top}}{n}$ is the projection matrix onto the image space of $\boldsymbol{L}$. To analyze the quality of the predconditioner $\boldsymbol{\Pi}\widehat{\boldsymbol{Z}}$, we need to provide bounds for $\boldsymbol{\Pi} - \boldsymbol{\Pi}\widehat{\boldsymbol{Z}}\boldsymbol{L}$. Define $\widetilde{\boldsymbol{L}}$ as an approximation for $\boldsymbol{L}$ with the errors induced by the Schur complement sparsification procedure

$$\widetilde{\boldsymbol{L}} = \widetilde{\boldsymbol{S}}^{(1)} + \sum_{i=1}^{d-1} \mathcal{P}\left(\widetilde{\boldsymbol{S}}^{(i+1)} - \mathrm{Sc}\left(\widetilde{\boldsymbol{S}}^{(i)}, F_i\right), C_i, C_i, n\right).$$

Define an auxiliary matrix

$$\widetilde{\boldsymbol{B}} = \delta_1 \boldsymbol{U}\left[\widetilde{\boldsymbol{L}}\right] + \sum_{i=1}^{d-1} \delta_{i+1} \mathcal{P}\left(\boldsymbol{U}\left[\mathrm{Sc}\left(\widetilde{\boldsymbol{L}}, \cup_{j=1}^{i} F_j\right)\right], C_i, C_i, n\right).$$

We can prove that $O(1) \cdot \boldsymbol{U}[\boldsymbol{L}] \preceq \boldsymbol{U}\left[\widetilde{\boldsymbol{B}}\right] \preceq O(\mathrm{poly}(n)) \cdot \boldsymbol{U}[\boldsymbol{L}]$.

Lemma 6.6. *Given $\left\{\alpha, \beta, \{\delta_i\}_{i=1}^{d}\right\}$-Schur complement chain with $d = O(\log n)$ and $\sum_{i=1}^{d} \delta_i \leq \frac{1}{4}$, by setting $N = O(\log n)$ in Algorithm 1, we have $\left\|\boldsymbol{\Pi} - \boldsymbol{\Pi}\widehat{\boldsymbol{Z}}\boldsymbol{L}\right\|_{\widetilde{\boldsymbol{B}}\to\widetilde{\boldsymbol{B}}} \leq \frac{1}{2}$.*

Then, Theorem 1.1 follows by Lemma 6.4 and Lemma 6.6.

Using the smaller Eulerian Laplacian sparsifiers based on short cycle decompositions to sparsify the approximate Schur complements returned by Algorithm 2, we get the following solver which has quadratic processing time, but faster solve time.

Corollary 6.7. *Given a strongly connected Eulerian Laplacian $\boldsymbol{L} \in \mathbb{R}^{n\times n}$, we can process it time $O(n^2 \log^{O(1)} n)$. Then, for each query vector $\boldsymbol{b} \in \mathbb{R}^n$ with $\boldsymbol{b} \perp \boldsymbol{1}$, we can compute a vector $\boldsymbol{x} \in \mathbb{R}^n$ with $\left\|\boldsymbol{x} - \boldsymbol{L}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}[\boldsymbol{L}]} \leq \epsilon \left\|\boldsymbol{L}^{\dagger}\boldsymbol{b}\right\|_{\boldsymbol{U}[\boldsymbol{L}]}$ in time $O(n \log^5 n \log(n/\epsilon))$.*

Remark 6.8. Combining Theorem 1.1 or Corollary 6.7 with Appendix D of [7] yields full solvers for strongly connected directed Laplacians.

REFERENCES

[1] AmirMahdi Ahmadinejad, Arun Jambulapati, Amin Saberi, and Aaron Sidford. 2019. Perron-Frobenius Theory in Nearly Linear Time: Positive Eigenvectors, M-matrices, Graph Kernels, and Other Applications. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, San Diego, California, USA, 1387–1404. https://doi.org/10.1137/1.9781611975482.85

[2] AmirMahdi Ahmadinejad, Jonathan A. Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil P. Vadhan. 2020. High-precision Estimation of Random Walks in Small Space. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, November 16-19, 2020*. IEEE, Durham, NC, USA, 1295–1306. https://doi.org/10.1109/FOCS46700.2020.00123

[3] Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. 2013. Spectral sparsification of graphs: theory and algorithms. *CACM* 56, 8 (Aug. 2013), 87–94. https://doi.org/10.1145/2492007.2492029

[4] Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. 2015. Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, July 3-6, 2015 (JMLR Workshop and Conference Proceedings, Vol. 40)*, Peter Grünwald, Elad Hazan, and Satyen Kale (Eds.). JMLR.org, Paris, France, 364–390. http://proceedings.mlr.press/v40/Cheng15.html

[5] Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. 2018. Graph Sparsification, Spectral Sketches, and Faster Resistance Computation, via Short Cycle Decompositions. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, October 7-9, 2018*, Mikkel Thorup (Ed.). IEEE Computer Society, Paris, France, 361–372. https://doi.org/10.1109/FOCS.2018.00042

[6] Michael B Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B Rao, and Aaron Sidford. 2018. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, October 7-9, 2018*. IEEE Computer Society, Paris, France, 898–909. https://doi.org/10.1109/FOCS.2018.00089

[7] Michael B Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B Rao, Aaron Sidford, and Adrian Vladu. 2017. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, June 19-23, 2017*. ACM, Montreal, QC, Canada, 410–419. https://doi.org/10.1145/3055399.3055463

[8] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. 2016. Faster Algorithms for Computing the Stationary Distribution, Simulating Random Walks, and More. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016*, Irit Dinur (Ed.). IEEE Computer Society, Hyatt Regency, New Brunswick, New Jersey, USA, 583–592. https://doi.org/10.1109/FOCS.2016.69

[9] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. 2013. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Symposium on Theory of Computing Conference, STOC'13, June 1-4, 2013*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM, Palo Alto, CA, USA, 911–920. https://doi.org/10.1145/2488608.2488724

[10] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. 2016. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, June 18-21, 2016*. ACM, Cambridge, MA, USA, 842–850. https://doi.org/10.1145/2897518.2897640

[11] Rasmus Kyng and Sushant Sachdeva. 2016. Approximate Gaussian Elimination for Laplacians - Fast, Sparse, and Simple. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016*, Irit Dinur (Ed.). IEEE Computer Society, Hyatt Regency, New Brunswick, New Jersey, USA, 573–582. https://doi.org/10.1109/FOCS.2016.68

[12] Yang P. Liu, Sushant Sachdeva, and Zejun Yu. 2019. Short Cycles via Low-Diameter Decompositions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, January 6-9, 2019*, Timothy M. Chan (Ed.). SIAM, San Diego, California, USA, 2602–2615. https://doi.org/10.1137/1.9781611975482.161

[13] Merav Parter and Eylon Yogev. 2019. Optimal Short Cycle Decomposition in Almost Linear Time. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019 (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Patras, Greece, 89:1–89:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.89

[14] John Peebles. 2019. *Fast spectral primitives for directed graphs.* Ph. D. Dissertation. Massachusetts Institute of Technology. Available at: https://dspace.mit.edu/handle/1721.1/124075.

[15] Richard Peng and Zhuoqing Song. 2021. Sparsified Block Elimination for Directed Laplacians. *arXiv preprint arXiv:2111.10257* (2021).

[16] Richard Peng and Daniel A Spielman. 2014. An efficient parallel solver for SDD linear systems. In *Symposium on Theory of Computing, STOC 2014, May 31 - June 03, 2014*. ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/2591796.2591832

[17] Daniel A. Spielman. 2010. Algorithms, Graph Theory, and Linear Equations in Laplacian Matrices. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010)*. World Scientific, Hyderabad, India, 2698–2722. https://doi.org/10.1142/9789814324359_0164

[18] Shang-Hua Teng. 2010. The Laplacian Paradigm: Emerging Algorithms for Massive Graphs. In *Theory and Applications of Models of Computation, 7th Annual Conference, TAMC 2010, June 7-11, 2010. Proceedings (Lecture Notes in Computer Science, Vol. 6108)*, Jan Kratochvíl, Angsheng Li, Jirí Fiala, and Petr Kolman (Eds.). Springer, Prague, Czech Republic, 2–14. https://doi.org/10.1007/978-3-642-13562-0_2

[19] Jan van den Brand. 2021. Unifying Matrix Data Structures: Simplifying and Speeding up Iterative Algorithms. In *4th Symposium on Simplicity in Algorithms, SOSA 2021, January 11-12, 2021*, Hung Viet Le and Valerie King (Eds.). SIAM, Virtual Conference, 1–13. https://doi.org/10.1137/1.9781611976496.1