Solving Directed Laplacian Systems in Nearly-Linear Time through Sparse LU Factorizations

Michael B. Cohen*, Jonathan Kelner†, Rasmus Kyng‡, John Peebles§, Richard Peng¶, Anup B. Rao‖, and Aaron Sidford**
* *MIT, micohen@mit.edu*
† *MIT, kelner@mit.edu*
‡ *Harvard, rjkyng@gmail.com*
§ *MIT, jpeebles@mit.edu*
¶ *Georgia Tech, rpeng@cc.gatech.edu*
‖ *Adobe, raoanupb@gmail.com*
** *Stanford, sidford@stanford.edu*

***Abstract*—In this paper, we show how to solve directed Laplacian systems in nearly-linear time. Given a linear system in an $n \times n$ Eulerian directed Laplacian with m nonzero entries, we show how to compute an ϵ-approximate solution in time $O(m \log^{O(1)}(n) \log(1/\epsilon))$. Through reductions from [Cohen et al. FOCS'16] , this gives the first nearly-linear time algorithms for computing ϵ-approximate solutions to row or column diagonally dominant linear systems (including arbitrary directed Laplacians) and computing ϵ-approximations to various properties of random walks on directed graphs, including stationary distributions, personalized PageRank vectors, hitting times, and escape probabilities. These bounds improve upon the recent almost-linear algorithms of [Cohen et al. STOC'17], which gave an algorithm to solve Eulerian Laplacian systems in time $O((m + n2^{O(\sqrt{\log n \log \log n})}) \log^{O(1)}(n\epsilon^{-1}))$.**

To achieve our results, we provide a structural result that we believe is of independent interest. We show that Eulerian Laplacians (and therefore the Laplacians of all strongly connected directed graphs) have sparse approximate LU-factorizations. That is, for every such directed Laplacian there are lower upper triangular matrices each with at most $\tilde{O}(n)$ nonzero entries such that there product spectrally approximates the directed Laplacian in an appropriate norm. This claim can be viewed as an analog of recent work on sparse Cholesky factorizations of Laplacians of undirected graphs. We show how to construct such factorizations in nearly-linear time and prove that once constructed they yield nearly-linear time algorithms for solving directed Laplacian systems. [1]

***Keywords*-Laplacian, directed Laplacian, diagonal dominance, LU-factorization, stationary distribution, PageRank, random walk, linear system solving, asymmetric matrices**

I. Introduction

A matrix $\mathbf{M}$ is *(column) diagonally dominant* if $\mathbf{M}_{ii} \geq \sum_{j \neq i} |\mathbf{M}_{ji}|$ for all i. Such matrices, which notably include Laplacians of graphs, are ubiquitous in computer science, with applications spanning scientific computing, graph theory, machine learning, and the analysis of random processes, among others. For *symmetric* diagonally dominant matrices, which include the Laplacians of *undirected* graphs, Spielman and Teng gave an algorithm in 2004 [1] to solve the corresponding linear systems in nearly-linear time. Since then, the ability solve such linear systems has emerged as a powerful algorithmic primitive, serving as a crucial subroutine in the design of faster algorithms for a long list of problems (e.g., [2]–[14]).

Moreover, the pursuit of faster, simpler, and more parallelizable Laplacian system solvers has driven numerous algorithmic advances, comprising both improvements in the underlying linear algebraic machinery [15]–[23] and applications of their ideas and techniques to problems in other domains (e.g., [19], [24]–[27]).

However, while this approach has been incredibly successful for symmetric linear systems and undirected graph optimization problems, comparable results for their asymmetric or directed counterparts have proven quite elusive. In particular the techniques for solving Laplacian systems seemed to rely intrinsically on multiple properties of undirected graphs, and, until recently, the best algorithms in the directed case simply treated the Laplacians as unstructured matrices and applied general linear algebraic routines, leading to super-quadratic running times.

Two recent papers [28], [29] suggested that it may be possible to close this gap, potentially laying the foundation for a new class of nearly-linear time algorithms for directed graphs and asymmetric linear systems. The first paper [28] showed that linear systems involving several natural classes of asymmetric matrices, including Laplacians of directed graphs, general square column diagonally dominant matrices, and their transposes (called *row diagonally dominant* matrices), could be reduced with only polylogarithmic overhead to solving linear systems in the Laplacians of Eulerian graphs. They further showed how to use these solvers, with only polylogarithmic overhead, to compute a wide range of fundamental quantities associated with

[1]This paper is an extended abstract. A the full version of the paper will appear on arXiv which will have complete proofs and further exposition; we encourage the reader to consult this full version instead.

2575-8454/18/$31.00 ©2018 IEEE
DOI 10.1109/FOCS.2018.00089

random walks on directed graphs, including the stationary distribution, personalized PageRank vectors, hitting times, and escape probabilities. The paper combined these reductions with an algorithm to solve Eulerian Laplacian systems in time $\widetilde{O}(m^{3/4}n + mn^{2/3})$ to achieve faster (but still significantly super-linear) algorithms for all of these problems.[2] The second paper [29] provided an improved solver for Eulerian systems that runs in almost-linear time $\widetilde{O}(m + n2^{O(\sqrt{\log n \log\log n})})$, providing almost-linear time algorithms for the problems reduced to such a solver in [28].

In this paper, we close the algorithmic gap between the directed and undirected cases (up to polylogarithmic factors) by providing an algorithm to solve Eulerian Laplacian systems in time $\widetilde{O}(m)$. Combining this with the reductions from [28] yields the first nearly-linear-time algorithms for all of the problems listed above.

To achieve our results we provide a structural result that we believe to be of independent interest. We show that Eulerian Laplacians (and therefore the Laplacians of all strongly connected directed graphs) have sparse approximate LU-factorizations. More precisely, we show that for every such directed Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$, there is a lower triangular matrix $\mathfrak{L} \in \mathbb{R}^{n\times n}$ and an upper triangular matrix $\mathfrak{U} \in \mathbb{R}^{n\times n}$ such that both matrices have at most $\widetilde{O}(n)$ nonzero entries, and $\mathfrak{L}\mathfrak{U}$ spectrally approximates $\mathbf{L}$ in an appropriate norm. We show how to construct such factorizations in nearly-linear time and prove that they yield nearly-linear-time algorithms for solving directed Laplacian systems.

This claim is analogous to the result first obtained by Lee, Peng and Spielman in [30], which showed that undirected Laplacians have sparse Cholesky factorizations, and our algorithm builds on a combination of the techniques in [23], [30] as well as the sparsification machinery developed in [29]. Unfortunately, as we will discuss below (Subsection I-B), there were several aspects of the approach in [23] that relied strongly on properties of symmetric Laplacians that are not present in the asymmetric setting, and obtaining a similar result for directed Laplacians required the development of new algorithmic and analytical techniques.

A. Our Results

The main technical result of this paper is the following theorem stating we can compute sparse approximate LU-factorizations of an Eulerian Laplacian in nearly-linear time.

Theorem I.1 (Sparse LU). *Given Eulerian Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ with m nonzero entries, $\epsilon \in (0, 1/2)$, and $\delta < 1/n$, in $\widetilde{O}(m + n\epsilon^{-8}\log^{O(1)}(1/\delta))$ time, with probability at least $1 - O(\delta)$ the* EULERIANLU$(\mathbf{L}, \delta, \epsilon)$, *produces lower and upper triangular matrices $\mathfrak{L}, \mathfrak{U} \in \mathbb{R}^{n\times n}$ such that for some symmetric PSD $\mathbf{F} \approx_{poly(n)} (\mathbf{L} + \mathbf{L}^\intercal)/2$, $(\mathfrak{L}\mathfrak{U})^\top \mathbf{F}^\dagger (\mathfrak{L}\mathfrak{U}) \succeq 1/O(\log^2 n) \cdot \mathbf{F}$, $\|\mathbf{F}^{\dagger/2}(\mathbf{L} - \mathfrak{L}\mathfrak{U})\mathbf{F}^{\dagger/2}\|_2 \leq \epsilon$ and $\max\{\mathrm{nnz}(\mathfrak{L}), \mathrm{nnz}(\mathfrak{U})\} \leq n\log^{O(1)}(1/\delta)\epsilon^{-6}$.*

The conditions of this theorem can be easily shown to yield that the pseudoinverse of $\mathfrak{L}\mathfrak{U}$ is a good preconditioner for solving $\mathbf{L}\boldsymbol{x} = \boldsymbol{b}$ for Eulerian $\mathbf{L}$. Consequently, we show in Section II as a corollary of this main theorem we show we can solve Eulerian Laplacian systems in nearly-linear time.

Theorem I.2 (Nearly-Linear Time Solver for Eulerian Laplacians). *Given an Eulerian Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$ with $\boldsymbol{b} \perp \mathbf{1}$, and $\epsilon \in (0, 1/2)$, in $O(m\log^{O(1)} n \log(1/\epsilon))$ time we can w.h.p. compute an ϵ-approximate solution $\tilde{\boldsymbol{x}}$ to $\mathbf{L}\boldsymbol{x} = \boldsymbol{b}$ in the sense that $\left\|\tilde{\boldsymbol{x}} - \mathbf{L}^\dagger\boldsymbol{b}\right\|_{\mathbf{U_L}} \leq \epsilon\left\|\mathbf{L}^\dagger\boldsymbol{b}\right\|_{\mathbf{U_L}}$ where $\mathbf{U_L} = (\mathbf{L} + \mathbf{L}^\top)/2$.*[3]

For simplicity, we assume all real number computations are exact throughout this paper. However, we believe a crude numerical stability analysis for constant lengthed inputs in the fixed point precision model similar to the one sketched in Appendix A of [28] is possible. Combining this result with the reductions from general directed Laplacians to Eulerian Laplacians proved in [28], leads to nearly-linear running times for all of the following problems:

- solving row diagonally dominant (or column diagonally dominant) linear systems including arbitrary (non-Eulerian) directed Laplacian systems
- computing the stationary distribution of a Markov chain
- personalized PageRank
- obtaining polynomially good estimates of the mixing time of a Markov chain
- computing the hitting time from one vertex to another
- computing escape probabilities for any triple of vertices
- approximately computing all-pairs commute times[4]

B. Overview of Approach

Here we present the broad algorithmic and analytical approach we take to provably solving Eulerian Laplacian systems in nearly-linear time. Our algorithm is broadly inspired by recent nearly-linear time (undirected) Laplacian system solvers based on performing repeated vertex elimination and Schur complement sparsification to compute a sparse approximate Cholesky factorization [22], [23].

Initially, one might hope to obtain our results by simply adapting the algorithm in [23], which is arguably the simplest known for provably solving undirected Laplacian systems in nearly-linear time. However, there are multiple substantial barriers in adapting any Cholesky factorization based undirected Laplacian system solver and its analysis for solving arbitrary Eulerian Laplacians. We obtain our

[2] Following the notation and terminology of the previous papers, we use $\widetilde{O}$ notation to suppress terms that are polylogarithmic in n, the natural condition number of the problem κ, and the desired accuracy ϵ. We use the term "nearly linear" for runtimes of $\widetilde{O}(m) = O(m)\log^{O(1)}(n\kappa\epsilon)$, and "almost linear" for runtimes of $O(m(n\kappa\epsilon^{-1})^{o(1)})$.

[3] Note Eulerian Laplacians have the special property that $\mathbf{U_L}$ is PSD.

[4] If one wishes to compute commute times for a number of pairs greater than the number of edges in the graph, the runtime will be nearly-linear in the output size instead of the number of the number of edges.

results by providing new algorithmic and analytical tools to systematically overcome these issues.

In the remainder of this section we outline the approach of this paper and these new tools and techniques as motivated by the barriers they overcome. We believe these insights may be useful for the analysis of directed graphs and asymmetric matrices more broadly. Here we simply outline the high level-ideas that underly them and connect these tools to the later sections which present them rigorously. For the details of the mathematical notation we use see Subsection I-D.

Vertex Elimination and Approximate LU-Factorization (Section II): To solve a system of equations in an Eulerian Laplacian $\mathbf{L} \in \mathbb{R}^{V \times V}$ we leverage the well know fact that it suffices to compute a matrix $\mathbf{Z}$ that is a good *preconditioner* or *approximate pseudoinverse* (See Definition II.5). As was shown in [29] if $\mathbf{Z}$ can be applied in nearly-linear time and $\mathbf{ZL} \approx \mathbf{I}_{\mathrm{im}(\mathbf{Z})}$ in a suitable norm then this suffices to solve Laplacian systems in $\mathbf{L}$ in nearly-linear time through an iterative method known as *preconditioned Richardson iterations* (See Lemma II.6).

Whereas [29] constructed $\mathbf{Z}$ through repeated sparsification and squaring, here we take the approach of [22], [23] of vertex elimination and repeated sparsification of Schur complements. The crux of this approach is the following classic line of reasoning. For any square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where F, C partition $[n]$ and $\mathbf{A}_{FF}$ is invertible it holds that

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{FF} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where $\mathbf{S} = \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC}$. Applying this expansion repeatedly to the *Schur complement*, $\mathbf{S}$ provided that each $\mathbf{A}_{FF}^{-1}$ is easily applied to vectors yields an efficient linear system solver due to the fact that upper and lower block triangular matrices are trivially inverted, e.g.

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{CF}\mathbf{A}_{FF}^{-1} & \mathbf{I} \end{bmatrix} .$$

The recent work of [22], [23] move beyond the classic analysis by leveraging that when $\mathbf{A}$ is a undirected Laplacian then so too is its Schur complement. By cleverly choosing the F, i.e. the blocks of vertices to eliminate, sparsifying the Schur complements, and recursing, these papers compute efficient preconditioners. Careful analysis is needed by these works to adapt this argument to deal with the fact that Laplacians are non-invertible (though their kernel is well understood) and bound the error incurred by sparsification.

In the work of [23], which we build upon, each F is simply a single coordinate or vertex of the associated graph. In this case eliminating this vertex induces a Schur complement with the vertex removed and an appropriately weighted clique added to its neighbors. Repeated elimination and recursion then yields a LU factorization of the original matrix $\mathbf{A}$ and in the case where $\mathbf{A}$ is symmetric a Cholesky factorization, i.e. where the upper triangular and lower triangular matrices are transposes of each other. [23] showed that by picking the vertex to eliminate randomly and sparsifying the cliques directly yields a sparse Cholesky factorization of symmetric Laplacian and a nearly-linear time solver.

It is easy to see that for Eulerian Laplacians, it is also the case that its Schur complements are Eulerian Laplacians and that eliminating a single vertex simply corresponds to deleting that vertex and adding a weighted complete bipartite graph or *biclique* from the vertices that yield incoming edges to the eliminated vertex to the vertices that yield outgoing edges from the eliminated vertex. Consequently, [23] suggests a natural approach for solving directed Laplacian systems and producing sparse LU-factorizations of Eulerian Laplacians, simply repeatedly eliminate random vertices and sparsify the bicliques that eliminating these vertices induce.

Unfortunately, there are multiple issues to this approach. First, sparsifying directed Eulerian Laplacians is more delicate than sparsifying undirected Laplacians and even showing that sparsifiers exist was a major contribution of [29]. Second, analyzing the error induced by sparsification is much more difficult for directed Laplacians and reasoning about this error was the major contributor to the almost linear rather than nearly-linear running time of [29]. Third, the reasoning of [23] required rather tight bounds on the sparsity induced by sparsification and without this showing that their algorithm works seems impossible.

Nevertheless, we show how to overcome these issues by providing a biclique sparsification technique with favorable properties for our analysis, providing new results on the error induced by Schur complement sparsification as it relates to the quality of a preconditioner, and modifying the [23] algorithm to only eliminate carefully chosen sets of vertices and alternate with full sparsification of the resulting graph. We discuss each of these further below. For a more detailed description of our algorithm see Section II.

Unbiased Degree Preserving Vertex Elimination (Section III): The first immediate issue in leveraging the insights from [23] to develop a a single vertex elimination algorithm capable of producing sparse LU factorizations of an Eulerian Laplacians is to determine how to sparsify the bicliques that elimination creates. On the one hand, the analysis of [23] crucially leverages that the sparsification procedure is unbiased, i.e. it produces a Laplacian that is in expectation the sparsified graph, with low variance so that matrix martingale arguments can be used to bound the error induced by the entire procedure. On the other the known sparsification results for Eulerian graphs [29] require that the sparsified graph is Eulerian and typically work by exactly preserving the in-degrees and out-degrees of vertices.

Unfortunately, these two constraints on a sparsification procedure seem at odds with each other as independent sampling to create an unbiased estimator likely does not preserves degrees. Moroever, it is difficult to see how completely drop either constraint. However, a different algorithmic primitive suggests optimism. Consider the edges of the vertex we eliminate. If we treat these edges as flows coming into and out of the vertex, then running a standard flow path decomposition on the these flows will result in a collection of flow paths – and if we treat these flow paths as edges that bypass the middle vertex, we get a graph on the neighbors of the eliminated vertex. This graph will be sparse and have the same in- and out-degree for every neighbor as the biclique created by elimination. We show that it is possible to randomize this flow decomposition in such a way that the sparse graph becomes an unbiased estimator of the dense biclique.

Formally, in Section III we remedy this issue by providing an efficient biclique sparsification procedure that is unbiased, with low variance, that exactly preserves the in and out degrees of vertices. This procedure works by careful sampling edges from the biclique so that no vertex has an excess of in-degree or out-degree and then carefully sampling from the remaining graph so that the conditional expectation is preserved. Crucially this procedure is not independent sampling from the entire graph. Moreover, we show that this procedure has low variance as desired. We hope the ideas of Section III could be useful for sparsification more broadly.

Schur Complement Blow Up and Bounding Error Increase (Section IV): While our biclique sparsification procedure of Section III provides hope of developing a single vertex elimination algorithm, it leaves significant issues in bounding the error induced by sparsification.

First, a key fact used in the previous analysis [23] is that, when recursing, the Schur complement matrices that result are always spectrally dominated by the original matrix. This is used to show that the overall error of the algorithm is small as long as the error is small in every recursive phase. It is not even obvious how to define the right notion of "spectral domination," in our context since the typical notion only applies to positive semidefinite matrices. Moreover, once it is appropriately defined (using notions from [29]), this spectral domination property fails even for simple graphs like the directed cycle, which can increase by a factor that is linear in the number of vertices.

Nevertheless, in Section IV we show that it is possible to efficiently find a large fraction of the vertices such that eliminating them only causes a constant multiplicative growth in the relevant norms. We show that (highly) row-column diagonally dominant sets are well-behaved in this sense and thus we can simply perform random vertex elimination restricted to these large sets. We note that though an analogous notion for undirected graphs was used in [30], [31], it played a different role there. They show that Jacobi iterations for the Laplacian minor defined by a diagonally dominant set converges quickly and also, that a sparse approximation of Schur complement can be constructed efficiently. On the other hand, we use the row-column diagonal dominance to show that an appropriately defined spectral norm of the Schur complement does not blow up.

Unfortunately, if the multiplicative errors from eliminating the highly row-column diagonally dominant sets compounded the overall error would still be too large for our elimination procedure to yield a good preconditioner. This compounding of error was a critical issue which prevented [29] from achieving a nearly-linear running time.

To circumvent this we develop a deeper understanding of the norms used to prove convergence of iterative solvers for linear equations in asymmetric matrices. Until the works [28], [29], the theory of iterative solvers for linear equations in asymmetric matrices (particular in the case of directed Laplacians) was relatively undeveloped. A key difficulty in this area is that convergence of a preconditioned solver for linear equations in asymmetric matrices relies on the preconditioner being an extremely accurate (w.r.t certain norms) to obtain any convergence at all. In contrast, iterative solvers for symmetric matrices will always converge provided small enough step sizes are used. Results in [28], [29] showed that the undirected Laplacian obtained from an Eulerian Laplacian by removing the direction on every edge leads to a norm that is conducive to both analyzing sparsification and proving convergence of an iterative solver. We show that there is a surprising amount of flexibility in family of norms that can be used to prove convergence of iterative solvers.

Our difficulties stem from vertex elimination on asymmetric matrices resulting in new matrices with larger norms in ways that make it difficult to control error accumulation from sparsification. But, in a twist, we show that vertex elimination *also* helps construct a new family of norms that that can be used to prove convergence of iterative solvers. Based on our vertex elimination sequence, we can construct a new matrix that is both sufficient for proving convergence of an iterative solver and spectrally dominates the "large" matrices created by vertex elimination. This spectral domination allows us to convert our local control on errors created by eliminating batches of vertices into global control on error accumulation across the entire algorithm (Lemma II.3). We then prove that our iterative solver converges quickly in this new norm, before converting the final error guarantee back to the norms in Theorem I.2 used for application purposes.

We believe that these tools for analyzing the error induced by sparsifying Schur complements could be useful for reasoning about asymmetric matrices more broadly.

Sparsification, Algorithm Design, and Analysis (Section II and Section V): The above reasoning give our key building blocks to developing a nearly-linear time algorithm for solving Eulerian Laplacian systems. They suggest that picking a block of row column diagonally dominant vertices, eliminating them with our unbiased degree preserving vertex elimination procedure, and then analyzing this in a carefully chosen norm induced by the recursion could work. Unfortunately, there is still one more issue that the sparsity of the resulting Schur complements this procedure induces grows much faster then in [23]. However, to overcome this we simply leverage black box previous work from [29] that Eulerian Laplacians have sparsifiers which can be efficiently computed. Consequently, by occasionally sparsifying the resulting Schur complement we can fix this final issue.

Putting together these pieces yields our algorithm. Algorithm 1 gives a rough sketch of this algorithm which alternates between vertex elimination on carefully chosen subsets and sparsification.

Algorithm 1: Overall Sketch of the Algorithm

```
1 for $i \leftarrow 1$ to $O(\log n)$ do
2     Choose a highly row-column diagonally
        dominant set $F \subset V$ of size proportional to $|V|$.
      for $v \in F$ do
3         Vertex Elimination $v$ : Use
            SingleVertexElim to eliminate vertex $v$
            and to add a sparse approximation of its
            Schur complement
4         Sparsify Graph: If the number of edges in the
            resulting graph is above a threshold, then
            use SparsifyEulerian to sparsify it.
5     $V \leftarrow V \backslash F$
```

Though this algorithm only describes how to eliminate vertices, as in Gaussian elimination, this also gives an LU factor decomposition. For a more detailed description of the algorithm, see Algorithm 3 and routines it calls. Details of line 3 of Algorithm 1 can be found in Section III.

We provide the analysis of this procedure (assuming the pieces of the rest of the paper) in Section II and in Section V we provide the careful martingale analysis of elimination and sparsification within a single highly row column diagonally dominant block. Though it has several pieces, ultimately, we believe this framework for solving Eulerian solvers is simpler than that in [29] and we hope that these pieces may find further use and possibly lead to even simpler algorithms.

C. Paper Outline

The presentation of these results is split into several pieces. In Section II we give the main algorithm that computes an approximate LU-factorization of an Eulerian Laplacian and prove its correctness as well as Theorem I.1 and Theorem I.2, assuming the analysis of later parts of the paper. In Section III, we analyze our single vertex elimination or biclique sparsification procedure. In Section IV we analyze the error induced by sparsifying Schur complements of highly diagonally dominant sets. In Section V we perform the matrix martingale analysis of the error incurred by single vertex elimination and graph sparsification. See the full version for complete proofs.

D. Preliminaries

Given a square matrix $\mathbf{M}$, we denote its symmetric part by $\mathbf{U_M} \stackrel{\text{def}}{=} (1/2)(\mathbf{M} + \mathbf{M}^\top)$. When we use $\mathbf{U_M}$ typically $\mathbf{U_M} \succeq 0$ and $\ker(\mathbf{M}) = \ker(\mathbf{M}^\top)$.

For matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and subsets $F, C \subseteq [n]$ we let $\mathbf{A}_{FC} \in \mathbb{R}^{F \times C}$ denote the sub-matrix of $\mathbf{A}$ corresponding to the F and C entries. Furthermore, for $v \in \mathbb{R}^N$ and $F \subseteq [n]$ we let v_F denote the restriction of v to the coordinates of v. Consequently, if $F, C \subseteq [n]$ partition $[n]$ then we have that $[\mathbf{A}v]_F = \mathbf{A}_{FF}v_F + \mathbf{A}_{FC}v_C$. Furthermore, we let

$$\textsc{Sc}[\mathbf{A}]_F \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{0}_{FF} & \mathbf{0}_{FC} \\ \mathbf{0}_{CF} & \mathbf{A}_{CC} - \mathbf{A}_{CF}\mathbf{A}_{FF}^{-1}\mathbf{A}_{FC} \end{bmatrix}$$

where $\mathbf{0}_{FC} \in \mathbb{R}^{F \times C}$ is the all zero matrix.

Norms:: Given an $n \times n$ PSD matrix $\mathbf{H}$, we define a semi-norm on vector $\|\cdot\|_\mathbf{H}$ by $\|x\|_\mathbf{H} = \sqrt{x^\top \mathbf{H} x}$. For any norm $\|\cdot\|$ defined on vectors in $\mathbb{R}^n$ we define the *seminorm* it induces on $\mathbb{R}^{n \times n}$ by $\|\mathbf{A}\| = \max_{x \neq 0} \|\mathbf{A}x\|/\|x\|$ for all $\mathbf{A} \in \mathbb{R}^{n \times n}$. When we wish to make clear that we are considering such a ratio we use the $\rightarrow$ symbol; e.g.,

$$\|\mathbf{A}\|_{\mathbf{H}\rightarrow\mathbf{H}} = \max_{x \neq 0} \frac{\|\mathbf{A}x\|_\mathbf{H}}{\|x\|_\mathbf{H}} = \max_{x \neq 0} \frac{(\mathbf{A}x)^\top \mathbf{H}(\mathbf{A}x)}{x^\top \mathbf{H} x} ,$$

but we may also simply write $\|\mathbf{A}\|_\mathbf{H} \stackrel{\text{def}}{=} \|\mathbf{A}\|_{\mathbf{H}\rightarrow\mathbf{H}}$.

Pseudoinverse and Square Roots: For symmetric PSD matrix $\mathbf{A}$ we use $\mathbf{A}^\dagger$ to denote its Moore-Penrose pseudoinverse of $\mathbf{A}$, we use $\mathbf{A}^{1/2}$ to denote its square root, i.e. the unique PSD matrix such that $[\mathbf{A}^{1/2}]^2 = \mathbf{A}$, and we use $\mathbf{A}^{\dagger/2}$ to denote $[\mathbf{A}^\dagger]^{1/2} = [\mathbf{A}^{1/2}]^\dagger$.

Upper and Lower Triangular Matrices: We say a square matrix $\mathfrak{U}$ is upper triangular if it has non-zero entries $\mathfrak{U}(i,j) \neq 0$ only for $i \leq j$ (i.e. above the diagonal). Similarly, we say a square matrix $\mathfrak{L}$ is lower triangular if it has non-zero entries $\mathfrak{U}(i,j) \neq 0$ only for $i \geq j$ (i.e. below the diagonal). Often, we will work with matrices that are not upper or lower triangular, but which for we know a permutation matrix $\mathbf{P}$ s.t. $\mathbf{P}\mathfrak{U}\mathbf{P}^\top$ is upper (respectively lower) triangular. For computational purposes, this is essentially equivalent to having a upper or lower triangular matrix, and *we will refer to such matrices as upper (or lower) triangular*. The algorithms we develop for factorization will always compute the necessary permutation.

Definition I.3 (Asymmetric Matrix Approximation [29])**.** *Matrix $\mathbf{B}$ is said to be an ϵ-*approximation *of matrix $\mathbf{A}$ if and only if $\mathbf{U_A}$ is symmetric PSD with $\ker(\mathbf{U_A}) \subseteq \ker(\mathbf{A} - \mathbf{B}) \cap \ker((\mathbf{A} - \mathbf{B})^\top)$ and $\|\mathbf{U_A}^{\dagger/2}(\mathbf{A} - \mathbf{B})\mathbf{U_A}^{\dagger/2}\|_2 \leq \epsilon$.*

In this paper we only use Definition I.3 in the restricted setting where $\ker(\mathbf{A}) = \ker(\mathbf{A}^\top)$.

Definition I.4 (RCDD Subset)**.** *$F \subseteq V$ is an α-RCDD subset of Laplacian $\mathbf{L} \in \mathbb{R}^{V \times V}$ if $\sum_{j \in F, j \neq i} |\mathbf{L}_{ij}| \leq \frac{1}{1+\alpha}|\mathbf{L}_{ii}|$ and $\sum_{j \in F, j \neq i} |\mathbf{L}_{ji}| \leq \frac{1}{1+\alpha}|\mathbf{L}_{ii}|$.*

Directed Laplacian Notation:: Here we provide notation regarding directed Laplacians and review basic facts regarding these matrices that were proved in [28]. A matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ is called a *directed Laplacian* if its off diagonal entries are non-positive, i.e. $\mathbf{L}_{i,j} \leq 0$ for all $i \neq j$, and $\mathbf{1}^\top \mathbf{L} = \mathbf{0}$, i.e. $\mathbf{L}_{ii} = -\sum_{j \neq i} \mathbf{L}_{ji}$ for all i. To every directed Laplacian $\mathbf{L} \in \mathbb{R}^{n \times n}$ we associate a graph $G_{\mathbf{L}} = (V, E, w)$ with vertices $V = [n]$, and edges (i,j) of weight $w_{ij} = -\mathbf{L}_{ji}$, for all $i \neq j \in [n]$ with $\mathbf{L}_{ji} \neq 0$. Occasionally we write $\mathbf{L} = \mathbf{D} - \mathbf{A}^\top$ to denote that we decompose $\mathbf{L}$ into the diagonal matrix $\mathbf{D}$ (where $\mathbf{D}_{ii} = \mathbf{L}_{ii}$ is the out degree of vertex i in $G_{\mathbf{L}}$) and non-negative matrix $\mathbf{A}$ (which is weighted adjacency matrix of $G_{\mathbf{L}}$, with $\mathbf{A}_{ij} = w_{ij}$ if $(i,j) \in E$, and $\mathbf{A}_{ij} = 0$ otherwise). Note $\mathbf{D}_{ii} = (\mathbf{A}\mathbf{1})_i$.

Let $\boldsymbol{\chi}_v \in \mathbb{R}^n$ be the vector whose v'th coordinate is set to one and all others to be zero, and $\boldsymbol{b}_{(u,v)} = \boldsymbol{\chi}_v - \boldsymbol{\chi}_u$. A directed Laplacian can be written as $\mathbf{L} = \sum_{(v,u) \in E} w_{(v,u)} \boldsymbol{b}_{(u,v)} \boldsymbol{\chi}_v^\top$.

A matrix $\mathbf{L}$ is called an *Eulerian Laplacian* if it is a directed Laplacian with $\mathbf{L}\mathbf{1} = \mathbf{0}$. Note that $\mathbf{L}$ is an *Eulerian Laplacian* if and only if its associated graph is Eulerian.

II. Main Algorithm

In this section, we give an overview of the key components of our algorithm for LU factorization and show how to use an LU factorization it generates to solve Eulerian Laplacians. We also give proofs of our two main statements on Eulerian Laplacians (Theorem I.1 and Theorem I.2) assuming auxiliary statements proven in this section and later in the chapter.

Standard LU Factorizations: The core of our algorithm is a sparsified version of LU factorization. The starting point for standard LU factorization is the formula for eliminating a single variable. Letting $C = V \setminus \{v\}$, the formula for elminating a single variable in a matrix $\mathbf{L}$ is

$$\mathbf{L} = \begin{bmatrix} 1 & \mathbf{0} \\ \frac{1}{\mathbf{L}_{vv}^{1/2}} \mathbf{L}_{Cv} & \mathbf{I} \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\mathbf{L}_{vv}^{1/2}} \mathbf{L}_{vC} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where $\mathbf{S} = \mathbf{L}_{CC} - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Cv} \mathbf{L}_{vC}$. This decomposes the matrix into a lower triangular matrix, a block diagonal matrix, and an upper triangular matrix. Recursively applying the same elimination procedure to the matrix $\mathbf{L}_{CC} - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Cv} \mathbf{L}_{vC}$ on the block diagonal, we can then get a decompostion into a product of a sequence of lower triangular matrices times a sequence of upper triangular matrices. Since the product of lower triangular matrices is lower triangular, and similar the product of upper triangular matrices is upper triangular, we can then collect these into one lower triagular factor $\mathfrak{L}$ and an upper triangular factor $\mathfrak{U}$ and write $\mathbf{L} = \mathfrak{L}\mathfrak{U}$. However, one can check that in fact the factors $\mathfrak{L}$ and $\mathfrak{U}$ have a simple structure: If d_i the diagonal entry corresponding to the variable being eliminated in round i, and $\boldsymbol{c}_i$ is the corresponding column scaled by $d_i^{-1/2}$, and $\boldsymbol{r}_i$ the row scaled scaled by $d_i^{-1/2}$ (e.g. so that in the first round $d_1 = \mathbf{L}_{vv}$, $\boldsymbol{c}_1 = d_1^{-1/2} \mathbf{L}_{vV}$, and $\boldsymbol{r}_1 = d_1^{-1/2} \mathbf{L}_{Vv}$), then

$$\mathfrak{L} = \begin{bmatrix} \boldsymbol{c}_1 & 0 & 0 & \\ \vert & \boldsymbol{c}_2 & 0 & \cdots \\ \vert & \vert & \boldsymbol{c}_3 & \end{bmatrix} \text{ and } \mathfrak{U} = \begin{bmatrix} \boldsymbol{r}_1 & \text{———} & \\ 0 & \boldsymbol{r}_2 & \text{——} \\ 0 & 0 & \boldsymbol{r}_3 \\ & \vdots & \end{bmatrix}.$$

For us, it will be more convenient to consider all rows and columns to be of length n, and abusing notation by redefining $\boldsymbol{c}_2 \leftarrow \begin{pmatrix} 0 \\ \boldsymbol{c}_2 \end{pmatrix}$ and so on yielding $\mathbf{L} = \mathfrak{L}\mathfrak{U} = \sum_i \boldsymbol{c}_i \boldsymbol{r}_i$.

The matrix $\mathbf{L}_{CC} - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Cv} \mathbf{L}_{vC}$ is known as the Schur complement onto C. We will use a convenient notation

$$\textsc{Sc}[\mathbf{L}]_C = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{CC} - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Cv} \mathbf{L}_{vC} \end{bmatrix}.$$

With a little effort, one can also see that

$$\mathbf{L} = \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Vv} \mathbf{L}_{vV} + \textsc{Sc}[\mathbf{L}]_C = \boldsymbol{c}_1 \boldsymbol{r}_1 + \textsc{Sc}[\mathbf{L}]_C \quad (1)$$

LU Factorization of Eulerian Laplacians: Let us restrict our attention to LU factorization of a matrix $\mathbf{L}$ which is a strongly connected Eulerian Laplacian. In general, LU factorization may run into trouble if one tries to eliminate a variable with a zero diagonal entry. But, one can show that this will never happen if the input matrix is an Eulerian Laplacian – except for the final diagonal entry.

We introduce notation for the directed Laplacian corresponding to the star graph of edges coming into and going out of a vertex v of in the graph corresponding to $\mathbf{L}$, which we write as $\textsc{St}[\mathbf{L}]_v = \sum_{(v,u) \in E} w_{(v,u)} \boldsymbol{b}_{(u,v)} \boldsymbol{\chi}_v^T + \sum_{(u,v) \in E} w_{(u,v)} \boldsymbol{b}_{(v,u)} \boldsymbol{\chi}_u^T$. Rearranging Equation (1) yields

$$\begin{aligned} \textsc{Sc}[\mathbf{L}]_{V \setminus \{v\}} &= \mathbf{L} - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Vv} \mathbf{L}_{vV} \\ &= \mathbf{L} - \textsc{St}[\mathbf{L}]_v + \textsc{St}[\mathbf{L}]_v - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Vv} \mathbf{L}_{vV} \end{aligned}$$

Importantly, when $\mathbf{L}$ is a strongly connected Eulerian Laplacian, then so is $\textsc{Sc}[\mathbf{L}]_{V \setminus \{v\}}$. Eulerian Laplacians are closed under taking Schur complements, and strong connectivity is preserved. Furthermore, $\mathbf{L} - \textsc{St}[\mathbf{L}]_v$ is simply the directed Laplacian corresponding to the graph of $\mathbf{L}$ with v and the edges incident on it removed, and $\textsc{St}[\mathbf{L}]_v - \frac{1}{\mathbf{L}_{vv}} \mathbf{L}_{Vv} \mathbf{L}_{vV}$, is a directed Laplacian of a weighted biclique on the neighbors of v. In general, neither of these parts of the Schur complement are individually Eulerian, but together they are.

Our algorithm.: As stated in Theorem I.1, we assume that the input matrix, $\mathbf{L}$, is a strongly connected Eulerian Laplacian. Our algorithm is based on standard LU factorization as outlined above. Since each elimination creates a biclique on the neighbors of the vertex being eliminated, the graph corresponding to the remaining Schur complement matrix can quickly grow dense. To rememedy this, we develop a method for computing sparse approximations of the Schur complement, without ever having to write down the dense biclique. This throws up several difficulties: The sparse approximation must accumulate very little error over many iterations and it must preserve Eulerianness. Subsection I-B for an overview of the challenges and the techniques we use to resolve these difficulties.

The main algorithm for LU factorization, Algorithm 3, has $p_{\max}$ phases or iterations. The routine for a single phase, given in Algorithm 2, iteratively eliminates vertices belonging to a random set selected to be RCDD. We need to perform the eliminations in phases because eliminating from RCDD sets helps us constrain the growth in the norm of certain matrices and control overall error accumulation.

In every iteration within any phase of the algorithm, a vertex, say v, is eliminated. We can afford to store the row and column that this creates in our LU factorization, but we cannot afford to compute the dense biclique that is created in the graph corresponding to the Schur complement.

Since the biclique directed Laplacian $\textsc{St}[\mathbf{L}]_v - \frac{1}{\mathbf{L}_{vv}}\mathbf{L}_{Vv}\mathbf{L}_{vV}$ is dense in general, we use the routine $\textsc{SingleVertexElim}(\cdot)$, given in Section III, to sparsify it at every elimination, while ensuring we always output an Eulerian Laplacian as our overall sparse approximation of $\textsc{Sc}[\mathbf{L}]_{V\setminus\{v\}}$. To increase the accuracy of our approximation, we average the result of $\widetilde{O}(1)$ calls to $\textsc{SingleVertexElim}(\cdot)$ at every elimination step, and this results in a slow increase in the the total number of edges at every iteration. We will use $\textsc{SparsifyEulerian}(\cdot)$ from [29] to sparsify the current graph every once in a while to keep the total edge count low enough. This is a routine to sparsify any Eulerian graph, and can be found in [29]. Recalling the notion of asymmetric matrix approximation from Definition I.3, the guarantees stated in Theorem 3.16 of [29] can be stated as follows:

Theorem II.1 (Eulerian Spectral Sparsification - Theorem 3.16 of [29] Rephrased). *For Eulerian Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ and $\epsilon, \delta, \in (0,1)$ with probability at least $1-\delta$ the $\textsc{SparsifyEulerian}(\mathbf{L}, \delta, \epsilon)$ computes in $\widetilde{O}(\mathrm{nnz}(\mathbf{L}) + n\epsilon^{-2}\log(1/\delta))$ time an Eulerian Laplacian $\widetilde{\mathbf{L}} \in \mathbb{R}^{n\times n}$ that is an ϵ-asymmetric spectral approximation of $\mathbf{L}$, has at most $\widetilde{O}(n\epsilon^{-2}\log(1/\delta))$ non-zeros entries, and has the same weighted in and out degrees as $\mathbf{L}$.*

Let the vertices be labeled in the order in which we eliminate them so that in phase p of the algorithm, we eliminate vertices $(i_p+1)\ldots i_{p+1}$. The phases are numbered from 0 to $p_{\max}-1$, while vertices are numbered from 1 to n. Note, this implies that at the start of phase p, we have a graph on $n - i_p$ vertices, starting initially with $i_0 = 0$, before any eliminations have taken place. We then index the elimination steps using the superscript $^{(i)}$ to denote the state of the algorithm **just after** we make the ith elimination step. $\mathbf{S}^{(i)}$ denotes the remaining matrix defined on variables we have not yet eliminated before ith elimination step. It is a sparse approximation of the Schur complement of $\mathbf{L}$ onto the remaining variables.

We define $\mathbf{S}^{(i)}$ as an $n \times n$ matrix, but it is always non-zero only on entries that correspond to pairs of variables neither of which have been eliminated. $\mathbf{L}^{(i)}$ denotes the original matrix perturbed by the perturbations introduced by $\textsc{SingleVertexElim}(\cdot)$ and $\textsc{SparsifyEulerian}(\cdot)$. We can express it as: $\mathbf{L}^{(i+1)} \overset{\mathrm{def}}{=} \mathbf{L}^{(i)} + (\mathbf{S}^{(i+1)} - \textsc{Sc}\left[\mathbf{L}^{(i)}\right]_{\{i+1,\ldots n-1\}})$. The quantity $\mathbf{S}^{(i)}$ is formally defined in Algorithm 3, along with a set of indices i_j (sometimes denoted i_p) for $j = 0\ldots p_{\max}-1$, where $p_{\max}$ is the total number of phases. Recall the index i_j is used to denote the state after the i_jth elimination step, and the index of the initial state is $i_0 = 0$.

Using a matrix martingale concentration inequality, we prove the following statement about the distortion bounded in each phase. (See Section V.)

Theorem II.2. *Given an $n \times n$ Eulerian matrix $\mathbf{L}$ with m non-zeros, an 0.1-RCDD subset J, an error parameter $\epsilon \in (0, 1/2)$, and a probability bound $\delta < 1/n$, $\textsc{SinglePhase}(\mathbf{L}, \delta, \epsilon)$ (Algorithm 2) creates with probability $1 - O(\delta)$ matrices $\widetilde{\mathbf{S}}, \mathfrak{L}', \mathfrak{U}'$, where $\widetilde{\mathbf{S}}$ is an Eulerian Laplacian, and $\mathfrak{L}', \mathfrak{U}'$ are upper and lower triangular respectively. The algorithm also finds a subset $\widehat{J}$ such that $\widetilde{\mathbf{S}} = \textsc{Sc}\left[\widetilde{\mathbf{L}}\right]_{V\setminus\widehat{J}}$ for the matrix $\widetilde{\mathbf{L}} = \widetilde{\mathbf{S}} + \mathfrak{L}'\mathfrak{U}'$ such that $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}(\widetilde{\mathbf{L}} - \mathbf{L})\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 \leq \epsilon$ and $|\widehat{J}| \geq \frac{1}{2}|J|$. Furthermore, the number of non-zeroes in $\widetilde{\mathbf{S}}$, $\mathfrak{L}'$, and $\mathfrak{U}'$ is at most $\widetilde{O}(n\epsilon^{-6}\log^5(1/\delta))$ and the runtime is at most $\widetilde{O}(m + n\epsilon^{-8}\log^7(1/\delta))$.*

This means that with high probability we can bound the distortion within each phase by: $\|\mathbf{U}_{\mathbf{S}^{(i_p)}}^{\dagger/2}\left(\mathbf{L}^{(i_p)} - \mathbf{L}^{(i_{p+1})}\right)\mathbf{U}_{\mathbf{S}^{(i_p)}}^{\dagger/2}\| \leq \theta_p\epsilon$, while we also have $(n - i_{p+1}) \leq 0.99(n - i_p)$

We now define a PSD matrix which we use for measuring the accumulation of errors: $\mathbf{F}^{(p)} \overset{\mathrm{def}}{=} \sum_{p'\leq p}\theta_{p'}\mathbf{U}_{\mathbf{S}^{(i_p)}}$. For convenience, we define $\mathbf{F} \overset{\mathrm{def}}{=} \mathbf{F}^{(p_{\max})} = \sum_{0\leq p<p_{\max}}\theta_p\mathbf{U}_{\mathbf{S}^{(i_p)}}$. We will set $\theta_p = \frac{1}{p_{\max}}$ so that $\sum_{p=0}^{p_{\max}-1}\theta_p = 1$. We bound the final error in terms of $\mathbf{F}$ as stated in the following lemma.

Lemma II.3. *With high probability $\|\mathbf{F}^{\dagger/2}\left(\mathbf{L} - \mathbf{L}^{(i)}\right)\mathbf{F}^{\dagger/2}\|_2 \leq \epsilon$ for every $i = 0\ldots n$.*

This alone is insufficient for using $\mathbf{L}^{(n)}$ as a preconditioner. Therefore, we show the following as well.

Lemma II.4. *With high probability, the final norm that we use, $\mathbf{F}$, satisfies $\mathbf{L}^{(n)\top}\mathbf{F}^{\dagger}\mathbf{L}^{(n)} \succeq 1/O(\log^2 n)\cdot\mathbf{F}$.*

Lemma II.4 and the ability to solve linear systems in $\mathbf{L}^{(n)}$ enable us to solve linear systems in $\mathbf{L}$. To formalize this, we

Algorithm 2: $\textsc{SinglePhase}(\mathbf{L}, \delta, \epsilon)$

```
Input: Eulerian Laplacian $\mathbf{L} \in \mathbb{R}^{V \times V}$ and $\epsilon \in \mathbb{R}_{>0}$
Output: Set of vertices eliminated, $F$, $\mathbf{S}^{|F|}$ an Eulerian Laplacian on $V \setminus F$, matrices
        $\mathfrak{U}^{(|F|)}$, $\mathfrak{L}^{(|F|)}$ with non-zeros only in the rows/columns corresponding to $F$
        respectively that are upper/lower triangle upon rearranging the vertices in $F$, and
        $\mathbf{L} \approx \mathbf{S}^{(|F|)} + \mathfrak{L}^{(|F|)}\mathfrak{U}^{(|F|)}$
1  $P \leftarrow \Theta(\log^2(1/\delta)/\epsilon^2)$ and $\epsilon' \leftarrow \Theta(\epsilon/P)$ ;
2  $\mathbf{S}^{(0)} \leftarrow \textsc{SparsifyEulerian}(\mathbf{L}, \delta/P, \epsilon')$ ;
3  $T \leftarrow \widetilde{O}(n\epsilon^{-6}\log^5(1/\delta))$ (the upper bound on nnz$(\mathbf{S}^{(0)})$ from Theorem II.1);
4  Pick a 0.1-RCDD subset of vertices $F^{(0)}$ from $\mathbf{S}^{(0)}$ (See full version) ;
5  $\mathfrak{U}^{(0)} \leftarrow \mathbf{0}$, $\mathfrak{L}^{(0)} \leftarrow \mathbf{0}$, $k_{\max} \leftarrow \lfloor |F^{(0)}|/2 \rfloor$;
6  for $k = 1 \ldots k_{\max}$ do
7      Among vertices in $F^{(k-1)}$ with degree at most twice the average, pick a random $v_k$. ;
8      $F^{(k)} \leftarrow F^{(k-1)} \setminus \{v_k\}$, $d^{(k)} \leftarrow \mathbf{S}^{(k-1)}(v_k, v_k)$;
9      Set $\mathfrak{U}^{(k)}$ to $\mathfrak{U}^{(k-1)}$ with row $v_k$ replaced by $\frac{1}{d^{(k)}}\mathbf{S}^{(k-1)}(v_k,:)$ and $\mathfrak{L}^{(k)}$ to $\mathfrak{L}^{(k-1)}$ with
       column $v_k$ replaced by $\mathbf{S}^{(k-1)}(v_k,:)$. ;
10     Set $\boldsymbol{l}^{(k)}$, $\boldsymbol{r}^{(k)}$ to length $n$ vectors containing the the off-diagonal non-zeros in the column and
       row of $v_k$ in $\mathbf{S}^{(k-1)}$ respectively. ;
11     Initialize the first matrix of the inner loop to be the exact Schur complement of pivoting out $v_k$
       from $\mathbf{S}^{(k-1)}$: $\mathbf{S}^{(k,0)} \leftarrow \mathbf{S}^{(k-1)} - \frac{1}{d^{(k)}}\boldsymbol{l}^{(k)}\boldsymbol{r}^{(k)\top}$;
12     for $t = 1 \ldots P$ do
13         $\mathbf{S}^{(k,t)} \leftarrow \mathbf{S}^{(k,t-1)} - \frac{1}{P}\left(\textsc{SingleVertexElim}\left(d^{(k)}, \boldsymbol{l}^{(k)}, \boldsymbol{r}^{(k)}\right) - \frac{1}{d^{(k)}}\boldsymbol{l}^{(k)}\boldsymbol{r}^{(k)\top}\right)$, (see Algorithm 4)
14     if nnz$(\mathbf{S}^{(k,P)}) \geq 2T$ then
15         $\mathbf{S}^{(k)} \leftarrow \textsc{SparsifyEulerian}(\mathbf{S}^{(k,P)}, \delta/P, \epsilon')$
16     else
17         $\mathbf{S}^{(k)} \leftarrow \mathbf{S}^{(k,P)}$
18 Return $\mathbf{S}^{(k_{\max})}$, $\mathfrak{U}^{(k_{\max})}$, $\mathfrak{L}^{(k_{\max})}$, and $k_{\max}$
```

use the definition of approximate pseudoinverses from [29].

Definition II.5 (Approximate Pseudoinverse)**.** *Matrix* $\mathbf{Z}$ *is an* ϵ-approximate pseudoinverse of matrix $\mathbf{M}$ with respect to a symmetric PSD $\mathbf{F}$*, if* $\ker(\mathbf{F}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M}^\intercal) = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^\intercal)$*, and* $\|\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}\mathbf{M}\|_{\mathbf{F}\rightarrow\mathbf{F}} \leq \epsilon$.

The reason why approximate pseudoinverses are useful is that if one preconditions with a solver for an approximate pseudoinverse, one can quickly solve the original system.

Algorithm 3: $\textsc{EulerianLU}(\mathbf{L}, \delta, \epsilon)$

```
Input: Eulerian Laplacian $\mathbf{L}$ and $\epsilon \in (0, 1/2)$
Output: lower and upper triangular matrices $\mathfrak{L}, \mathfrak{U}$ whose product approximates $\mathbf{L}$
1  $\mathbf{L} \leftarrow \textsc{SparsifyEulerian}(\mathbf{L}, \delta/2, O(\epsilon/\log n))$
2  $\mathbf{S}^{(0)} \leftarrow \mathbf{L}$ and set $\mathfrak{L} \leftarrow \mathbf{0}$ $\mathfrak{U} \leftarrow \mathbf{0}$, $j \leftarrow 0$, $i_j \leftarrow 0$
3  while $i_j < n$   (i.e., for $p_{\max} = O(\log n)$ iterations) do
4      $(\mathbf{T}, \mathfrak{L}', \mathfrak{U}', k_{\max}) \leftarrow \textsc{SinglePhase}(\mathbf{S}^{(i_j)}, \frac{\delta}{2\log n}, O(\frac{\epsilon}{\log n}))$
5      $j \leftarrow j+1$,$i_j \leftarrow i_{j-1} + k_{\max}$, $\mathbf{S}^{(i_j)} \leftarrow \mathbf{T}$
6      Insert the nonzero vectors from the partial LU factorization $\mathfrak{L}', \mathfrak{U}'$ into their corresponding
       locations $\mathfrak{L}$ and $\mathfrak{U}$, respectively.
7  return $\mathfrak{L}, \mathfrak{U}$
```

Lemma II.6 (Preconditioned Richardson, [29] Lemma 4.2, pg. 30)**.** *Let* $\boldsymbol{b} \in \mathbb{R}^n$ *and* $\mathbf{M}, \mathbf{Z}, \mathbf{F} \in \mathbb{R}^{n \times n}$ *such that* $\mathbf{F}$ *is symmetric positive semidefinite,* $\ker(\mathbf{F}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M}^\intercal) = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^\intercal)$*, and* $b \in \mathrm{im}\,(\mathbf{M})$*. Then if one performs* $t \geq 0$ *iterative refinement steps with step size* $\eta > 0$*, one obtains a vector* $\boldsymbol{x}_t = \textsc{PreconRichardson}(\mathbf{M}, \mathbf{Z}, \boldsymbol{b}, \eta, t)$ *such that*

$$\left\|\boldsymbol{x}_t - \mathbf{M}^\dagger \boldsymbol{b}\right\|_{\mathbf{F}} \leq \left\|\mathbf{I}_{\mathrm{im}\,(\mathbf{M})} - \eta\mathbf{Z}\mathbf{M}\right\|_{\mathbf{F}\rightarrow\mathbf{F}}^t \left\|\mathbf{M}^\dagger \boldsymbol{b}\right\|_{\mathbf{F}}.$$

The properties of the approximate LU factorization produced by our algorithm imply that a solver for systems in it is an approximate pseudoinverse of the original Laplacian.

Lemma II.7. *Suppose we are given matrices* $\mathbf{L}$, $\widetilde{\mathbf{L}}$ *and a positive semi-definite matrix* $\mathbf{F}$ *such that* $\ker(\mathbf{F}) \subseteq \ker(\mathbf{L}) = \ker(\mathbf{L}^\intercal) = \ker(\widetilde{\mathbf{L}}) = \ker(\widetilde{\mathbf{L}}^\intercal)$*,* $\|\mathbf{F}^{\dagger/2}(\mathbf{L} - \widetilde{\mathbf{L}})\mathbf{F}^{\dagger/2}\|_2 \leq \epsilon$*, and* $\widetilde{\mathbf{L}}^T\mathbf{F}^\dagger\widetilde{\mathbf{L}} \succeq \gamma\mathbf{F}$*. Then* $\widetilde{\mathbf{L}}^\dagger$ *is an* $\sqrt{\epsilon^2\gamma^{-1}}$*-approximate pseudoinverse for* $\mathbf{L}$ *w.r.t. the norm* $\mathbf{F}$.

We would like for the error guarantees of our solver to be in terms of $\mathbf{U_L}$. In order to provide such guarantees, we need to relate this matrix to $\mathbf{F}$.

Lemma II.8. $\mathbf{U_L}/O(\log(n)) \preceq \mathbf{F} \preceq O(n^2\log^5 n) \cdot \mathbf{U_L}$

We have now stated the key theorems and lemmas needed to analyze correctness and prove Theorem I.1.

III. Unbiased Degree Preserving Vertex Elimination

In this section we provide and analyze $\textsc{SingleVertexElim}$, see Algorithm 4, which produces a sparse approximation of the clique created by Gaussian Elimination on an Eulerian directed Laplacian. It can be implemented to run in $O(\deg(v)\log\deg(v))$ time where deg is the combinatorial degree of the vertex v being eliminated, i.e. the number of vertices incident to it.

The algorithm has three key features. When including self-loops, it preserves the weighted in and out degree of each vertex. This ensures the graph created by replacing the clique with the sparse approximation is still Eulerian. Note that we may get self-loops which will cancel out and change the degree of vertices, but it won't change the fact that each vertex still has in-degree equal to out-degree. Secondly, it produces a *sparse* approximation of the biclique created by elimination. Thirdly, it achieves these guarantees while being an unbiased estimate of the biclique.

Algorithm 4: SINGLEVERTEXELIM$(\boldsymbol{l}, \boldsymbol{r})$

```
   Input: $\boldsymbol{l}, \boldsymbol{r} \in \mathbb{R}^n_{\geq 0}$ such that $\mathbf{1}^\top \boldsymbol{l} = \mathbf{1}^\top \boldsymbol{r}$.
   Output: $\mathbf{N} \in \mathbb{R}^{n \times n}$
1  if $\mathbf{1}^\top \boldsymbol{l} = 0$ then
2      return $\mathbf{0}$ ;
3  else if $\min(\boldsymbol{l}) \leq \min(\boldsymbol{r})$ then
4      $i \leftarrow \arg\min(\boldsymbol{l})$;
5      Pick index $j \leftarrow k$ with probability $\boldsymbol{r}(k)/s$;
6      return $\boldsymbol{l}(i)\boldsymbol{\chi}_i\boldsymbol{\chi}_j^\top$ + SINGLEVERTEXELIM($\boldsymbol{l} - \boldsymbol{l}(i)\boldsymbol{\chi}_i, \boldsymbol{r} - \boldsymbol{l}(i)\boldsymbol{\chi}_j$);
7  else
8      $i \leftarrow \arg\min(\boldsymbol{r})$;
9      Pick index $j \leftarrow k$ with probability $\boldsymbol{l}(k)/s$;
10     return $\boldsymbol{r}(i)\boldsymbol{\chi}_j\boldsymbol{\chi}_i^\top$ + SINGLEVERTEXELIM($\boldsymbol{l} - \boldsymbol{r}(i)\boldsymbol{\chi}_j, \boldsymbol{r} - \boldsymbol{r}(i)\boldsymbol{\chi}_i$);
```

A crucial matrix used in analyzing the elimination of a single vertex is the Schur complement of the star incident on the eliminated vertex in the Laplacian's undirectification.

Definition III.1. *Given an Eulerian Laplacian* $\mathbf{L}$ *and a vertex* v*, let* $\mathbf{U}_{local} = \textsc{St}[\mathbf{U_L}]_v - \frac{1}{\mathbf{U_L}(v,v)}\mathbf{U_L}(:,v)\mathbf{U_L}(v,:)$.

Note that $\mathbf{U}_{local} \preceq \textsc{St}[\mathbf{U_L}]_v$ and $\mathbb{E}_v\left[\mathbf{U}_{local}\right] \preceq \frac{2}{n}\mathbf{U_L}$ for a random choice of vertex v in a n vertex graph. Using this norm we show the following properties of Algorithm 4. .

Lemma III.2 (Single Vertex Elimination Routine)**.** *The routine* SINGLEVERTEXELIM *takes the in and out adjacency list vectors* $\boldsymbol{l}$ *and* $\boldsymbol{r}$ *of a vertex* u *in an Eulerian Laplacian with* d *non-zeros, and produces a non-negative matrix* $\mathbf{A}$ *with at most* d *non-zeros such that the error matrix* $\mathbf{X} = \frac{1}{\boldsymbol{r}^\top \mathbf{1}}\boldsymbol{l}\boldsymbol{r}^\top - \mathbf{A}$ *satisfies* $\mathbf{X1} = 0$*,* $\mathbf{X}^\top\mathbf{1} = 0$*,* $\mathbb{E}\left[\mathbf{X}\right] = 0$*, and for the local undirectification,* $\mathbf{U}_{local}$ *as given in Definition III.1, we have* $\|\mathbf{U}_{local}^{\dagger/2}\mathbf{X}\mathbf{U}_{local}^{\dagger/2}\|_2 \leq 4$.

IV. Robustly Bounded Schur Complement Sets

As we have discussed, one of the key difficulties in applying repeated vertex elimination to solve Eulerian Laplacian systems is that unlike with symmetric Laplacian systems the Schur complement of an Eulerian Laplacian may be much larger than that of the original Laplacian. Consequently, if we simply eliminate an arbitrary set of vertices the error we incur my too large to produce an effective preconditioner.

To circumvent this we eliminate vertices in phases where in each phase we only eliminate vertices that do not "blow up" the Schur complement, i.e. induce Schur complements that are not spectrally dominated by a small multiple of the current Laplacian. Formally, we only eliminate vertices from what we call *robustly bounded schur complement sets* defined below. These are sets, where even under a small amount of spectral error, the Schur complement is not too much larger than the original graph.

Definition IV.1 (Robustly Bounded Schur Complement Set)**.** *Given an Eulerian Laplacian* $\mathbf{L}$*, a* robustly bounded Schur complement set *vertex set* J*, is a subset of the vertices of* $\mathbf{L}$*, such that for any* $\widehat{J} \subseteq J$ *and any* $\widetilde{\mathbf{L}}$ *that* $1/2$*-approximates* $\mathbf{L}$ *we have* $\mathbf{U}_{\textsc{Sc}[\widetilde{\mathbf{L}}]_{\widehat{J}}} \preceq O(1)\mathbf{U_L}$.

In this section, we formally show that α-RCDD subsets of the vertices are robustly bounded Schur complement sets and therefore we can easily find such sets. The main result of this section is the following formalization of this claim.

Lemma IV.2 (α-RCDD Sets are Robustly Bounded)**.** *Given an Eulerian Laplacian* $\mathbf{L}$*, for any fixed constant* α*, an* α*-RCDD subset* J *of the vertices of* $\mathbf{L}$ *is robustly bounded.*

We prove this lemma in several pieces. First we provide a general lemma that upper bounds the symmeterization of the Schur complement of a general matrix spectrally via its symmetrization. This lemma is the main tool we use to reason about the Schur complement of subsets of a α-RCDD subset of a Eulerian Laplacian.

Lemma IV.3. *If* $\mathbf{N} \in \mathbb{R}^{n\times n}$ *satisfies* $\mathbf{U} \stackrel{\text{def}}{=} \mathbf{U_N} \succeq \mathbf{0}$ *and* $F, C \subseteq [n]$ *is a partition of* $[n]$ *where* $\mathbf{U}_{FF} \succ \mathbf{0}$ *then* $\mathbf{U}_{\textsc{Sc}[\mathbf{N}]_F} \preceq (1+\alpha)\mathbf{U}$ *provided the following condition holds*

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{bmatrix} [\mathbf{N}_{FF}^{-1}]^\top \mathbf{U}_{FF}\mathbf{N}_{FF}^{-1} & \mathbf{0}_{FC} \\ \mathbf{0}_{CF} & \mathbf{0}_{CC} \end{bmatrix} \preceq \alpha[\mathbf{N}^\dagger]^\top \mathbf{U}\mathbf{N}^\dagger \ .$$

Next, we show how to apply the Schur complement bounds of the previous subsection to bound the increase in Schur complements for Eulerian Laplacians. In particular we bound the blowup of the Schur complements as we pivot away α-RCDD subsets of vertices.

Lemma IV.4. *Suppose that* $\mathbf{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n\times n}$ *is an Eulerian Laplacian, and* $F \subseteq [n]$ *is an* α*-RCDD subset. Then* $\mathbf{U}_{\textsc{Sc}[\mathbf{L}]_F} \preceq (3 + \frac{2}{\alpha})\mathbf{U_L}$.

To prove this lemma, first we provide the following lemma, which is a self contained fact about Eulerian Laplacians that will allow us to leverage Lemma IV.3.

Lemma IV.5. *Suppose that* $\mathbf{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n\times n}$ *is an Eulerian Laplacian associated with directed graph* $G = (V, E, w)$ *and let* $\mathbf{U} = \frac{1}{2}(\mathbf{L} + \mathbf{L}^\top)$*. Then* $\mathbf{L}^\top\mathbf{D}^{-1}\mathbf{L} \preceq 2 \cdot \mathbf{U}$.

Using Lemma IV.3 and Lemma IV.5 we can bound the increase of Schur complements of Eulerian Laplacians.

Lemma IV.6. *Suppose that* $\mathbf{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n\times n}$ *is an Eulerian Laplacian, let* $\mathbf{U} \overset{\text{def}}{=} \mathbf{U}_{\mathbf{L}}$ *and let* $F \subseteq [n]$ *such that* $\mathbf{U}_{FF} \succeq \frac{1}{\alpha}\mathbf{D}_{FF}$ *then* $\mathbf{U}_{\text{Sc}[\mathbf{L}]_F} \preceq (1+2\alpha)\mathbf{U}$.

Finally, Lemma IV.2 follows from the following stability result Lemma IV.7 regarding Schur complements.

Lemma IV.7. *Suppose* $\mathbf{N} \in \mathbb{R}^{n\times n}$ *satisfies* $\mathbf{U} \overset{\text{def}}{=} \mathbf{U}_{\mathbf{N}} \succeq \mathbf{0}$ *and* $\ker(\mathbf{N}) = \ker(\mathbf{N}^\top)$ *and suppose* $F, C \subseteq [n]$ *is a partition of* $[n]$ *where* $\mathbf{U}_{FF} \succ \mathbf{0}$. *If* $\widetilde{\mathbf{N}}$ ϵ*-approximates* $\mathbf{N}$ *then* $\mathbf{U}_{\text{Sc}[\widetilde{\mathbf{N}}]_F} \preceq \left(\frac{1+\epsilon}{1-\epsilon}\right)^2 \mathbf{U}_{\text{Sc}[\mathbf{N}]_F}$.

V. Single Phase Analysis

Here we analyze SINGLEPHASE and prove Theorem II.2. We need to show that in SINGLEPHASE (Algorithm 2), we do not do a (very expensive) full sparsification of the entire graph too many times. Second, our runtime will have a term that is nearly linear in the sum of the degrees of the vertices we eliminate. To say that these aren't too big, we need to bound how much we can blow up the average degree of the graph and number of edges in it after doing one elimination. The following lemmas help us bound the running time of SINGLEPHASE, and the number of calls to SPARSIFYEULERIAN made inside it.

Lemma V.1. *If* $\mathbf{L}$ *is an Eulerian Laplacian on* n *vertices then in* SINGLEPHASE$(\mathbf{L}, \delta, \epsilon)$, *for all* k, $\text{nnz}(\mathbf{S}^{(k+1)}) \leq (1 + O(\frac{P}{n-k}))\,\text{nnz}(\mathbf{S}^{(k)})$.

Lemma V.2. *Suppose* $\mathbf{L}$ *is an Eulerian Laplacian on* n *vertices with* m *non-zeros. In the* for*-loop of* SINGLEPHASE *(Algorithm 2),* SPARSIFYEULERIAN$(\cdot)$ *is called* $O(P) = O(\epsilon^{-2}\log^2(1/\delta))$ *times, each time after the first on an Eulerian Laplacian with* $\widetilde{O}(n\epsilon^{-6}\log^5(1/\delta))$ *edges. The total running time for a call to* SINGLEPHASE$(\mathbf{L}, \delta, \epsilon)$ *is* $\widetilde{O}(m + n\epsilon^{-8}\log^7(1/\delta))$.

To prove Theorem II.2 we want to show

$$\Pr\left[\left\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}\left(\mathbf{L}^{(k_{\max})} - \mathbf{L}\right)\mathbf{U}_{\mathbf{L}}^{\dagger/2}\right\|_2 > \epsilon\right] \leq O(\delta) \tag{2}$$

where $\mathbf{L}$ is the input to SINGLEPHASE and $\mathbf{L}^{(k_{\max})}$ is the output. Secondly, we also need to prove the required bounds on the running time. We set up a matrix martingale to help us prove the theorem. Consider the inner loops (inside k, inside the t loops) of SINGLEPHASE. We can define the change in each step to be: $\mathbf{X}^{(k,t)} \overset{\text{def}}{=} \mathbf{S}^{(k,t)} - \mathbf{S}^{(k,t-1)}$. Since each $\mathbf{X}^{(k,t)}$ has zero expectation, we can define a zero-mean martingale sequence

$$\mathbf{M}^{(k,t)} = \sum_{\hat{k}=1}^{k} \sum_{\hat{t}=1}^{\substack{P \text{ if } \hat{k} < k \\ t \text{ if } \hat{k} = k}} \mathbf{X}^{(\hat{k},\hat{t})} = \sum_{(\hat{k},\hat{t}) \leq (k,t)} \mathbf{X}^{(\hat{k},\hat{t})}.$$

Here we overload the $\leq$ and $<$ notation to handle pair of variables in the lexicographical sense. The final step of this martingale is $\mathbf{M}^{(k_{\max},P)}$.Note that it is not the case that in general $\mathbf{L}^{(k)} \neq \mathbf{L} + \mathbf{M}^{(k,P)}$, because the martingale does not include the changes introduced by calls to SPARSIFYEULERIAN.

To track the changes caused by SPARSIFYEULERIAN, we further define changes from the sparsification steps: We define $\mathbf{Z}^{(0)} \overset{\text{def}}{=} \mathbf{S}^{(0)} - \mathbf{S}$, and for $k > 0$, we let $\mathbf{Z}^{(k)} \overset{\text{def}}{=} \mathbf{S}^{(k)} - \mathbf{S}^{(k,P)}$. We express the output matrix as $\mathbf{L}^{(k)} = \mathbf{L} + \mathbf{M}^{(k,P)} + \sum_{\hat{k}=0}^{k} \mathbf{Z}^{(\hat{k})}$. We also define the following intermediate matrices: $\mathbf{L}^{(k,t)} = \mathbf{L} + \mathbf{M}^{(k,t)} + \sum_{\hat{k}=0}^{k-1} \mathbf{Z}^{(\hat{k})}$. Next, we define stopped martingale variables.

Definition V.3. *We define the event of the martingale being* safe *until* (k,t), *for* $t \in \{1, \ldots, P+1\}$ *by* $\textbf{\textit{SAFE}}^{(k,t)}$ *to mean that both of the following conditions hold:*

1) *All calls to* SPARSIFYEULERIAN *strictly before the kth elimination have been successful, i.e.* $\|\mathbf{U}^{\dagger/2}_{\mathbf{S}^{(\hat{k},P)}} \mathbf{Z}^{(\hat{k})} \mathbf{U}^{\dagger/2}_{\mathbf{S}^{(\hat{k},P)}}\|_2 \leq \epsilon'$ *for all* $\hat{k} < k$.
2) *For all indices* $(\hat{k}, \hat{t})$ *strictly preceding* (k,t), *i.e.* $(\hat{k},\hat{t}) < (k,t)$, *we had* $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}\mathbf{M}^{(\hat{k},\hat{t})}\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 \leq \epsilon/2$.

Note that in addition to the events $\mathbf{SAFE}^{(k,1)}, \ldots, \mathbf{SAFE}^{(k,P)}$, we also consider event $\mathbf{SAFE}^{(k,P+1)}$. Index $(k, P+1)$ does not correspond to a step of the martingale, but the condition that the martingale has small norm for strictly preceding indices is still well-defined. This notation helps us prove statements by induction on the ordered indices $(k,1) < (k,2) < \ldots < (k,P) < (k,P+1) < (k+1,1) < \ldots$.

We then define a truncated Martingale as one where the steps incur no additional error once it fails. Its steps are:

$$\overline{\mathbf{X}}^{(k,t)} \overset{\text{def}}{=} \begin{cases} \mathbf{X}^{(k,t)} & \text{if } \mathbf{SAFE}^{(k,t)} \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{3}$$

The following sequence of sums of $\overline{\mathbf{X}}^{(k,t)}$ is another zero mean martingale: $\overline{\mathbf{M}}^{(k,t)} = \sum_{(\hat{k},\hat{t})\leq(k,t)} \overline{\mathbf{X}}^{(\hat{k},\hat{t})}$. Ultimately, we want to bound the probability of the event that $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}(\mathbf{L}^{(k_{\max})} - \mathbf{L})\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 > \epsilon$ occurs.

Lemma V.4. *If* $\textbf{\textit{SAFE}}^{(k,t)}$ *holds then (1)* $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}(\mathbf{L}^{(\hat{k},\hat{t})} - \mathbf{L})\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 \leq \epsilon$ *and* $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}(\mathbf{L}^{(\hat{k})} - \mathbf{L})\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 \leq \epsilon$ *for all* $(\hat{k},\hat{t}) < (k,t)$ *and (2)* $\mathbf{U}_{\mathbf{S}^{(\hat{k})}} \preceq O(1) \cdot \mathbf{U}_{\mathbf{L}}$ *for all* $\hat{k} < k$.

From this lemma, we see that if we can prove $\Pr[\neg\mathbf{SAFE}^{(n+1,1)}] \leq O(\delta)$, it will imply Equation (2). Note that $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}\mathbf{M}^{(k,t)}\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 > \epsilon/2$ implies $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}\overline{\mathbf{M}}^{(k,t)}\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 > \epsilon/2$. Hence, when upper bounding the probability of $\Pr[\neg\mathbf{SAFE}^{(k,t)}]$, we can instead consider the higher probability event that it is note the case that both $\|\mathbf{U}_{\mathbf{S}^{(\hat{k},P)}}^{\dagger/2}\mathbf{Z}^{(\hat{k})}\mathbf{U}_{\mathbf{S}^{(\hat{k},P)}}^{\dagger/2}\|_2 \leq \epsilon'$ for all $\hat{k} \leq k$ and $\|\mathbf{U}_{\mathbf{L}}^{\dagger/2}\overline{\mathbf{M}}^{(\hat{k},\hat{t})}\mathbf{U}_{\mathbf{L}}^{\dagger/2}\|_2 \leq \epsilon/2$ for all $(\hat{k},\hat{t}) \leq (k,P)$.

The matrix Martingale inequality that we use the rectangular matrix Martingale from Cor 1.3. of [32].

Lemma V.5 (Matrix Freedman). *Let $\mathbf{E}^{(1)}\ldots\mathbf{E}^{(N)}$ be a sequence of matrices, and use $\mathbb{E}_{j-1}\left[\mathbf{E}^{(j)}\right]$ to denote the expectation of $\mathbf{E}^{(j)}$ conditioned on $\mathbf{E}^{(j-1)}, \mathbf{E}^{(j-2)}, \ldots, \mathbf{E}^{(1)}$ such that for any i $\mathbb{E}_i\left[\mathbf{E}^{(i)}\right] = 0$ and $\|\mathbf{E}^{(i)}\|_2 \leq \rho$ over its entire support. Then for any error t we have that the probability that there exists $k \geq 0$ such that $\|\sum_{j\leq k}\mathbf{E}^{(j)}\|_2 \geq t$ and $\|\sum_{j\leq k}\mathbb{E}_{j-1}\left[\mathbf{E}^{(j)}\mathbf{E}^{(j)\top} + \mathbf{E}^{(j)\top}\mathbf{E}^{(j)}\right]\|_2 \leq \sigma^2$ is at most $n \cdot \exp\left(\frac{-t^2}{100(\sigma^2+t\rho)}\right)$.*

As we are always normalizing by $\mathbf{U_L}$, we can define rescaled versions of the martingale, as well as variances: by $\widehat{\mathbf{M}}^{(k,t)} \overset{\text{def}}{=} \mathbf{U}_{\mathbf{L}}^{\dagger/2}\overline{\mathbf{M}}^{(k,t)}\mathbf{U}_{\mathbf{L}}^{\dagger/2}$ and $\widehat{\mathbf{X}}^{(k,t)} \overset{\text{def}}{=} \mathbf{U}_{\mathbf{L}}^{\dagger/2}\overline{\mathbf{X}}^{(k,t)}\mathbf{U}_{\mathbf{L}}^{\dagger/2}$. Together with a union bound, this allows us to bound $\Pr[\neg\mathbf{SAFE}^{(n+1,1)}]$ by the sum of

$$\sum_k \Pr\left[\left\|\mathbf{U}_{\mathbf{S}^{(k,P)}}^{\dagger/2}\mathbf{Z}^{(k)}\mathbf{U}_{\mathbf{S}^{(k,P)}}^{\dagger/2}\right\|_2 > \epsilon'\right] \tag{4}$$

the probability there exists (k,t) with $\|\widehat{\mathbf{M}}^{(k,t)}\|_2 \geq s$ and

$$\left\|\sum_{(\hat{k},\hat{t})\leq(k,t)} \mathbb{E}_{<(\hat{k},\hat{t})}\left[\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top} + \widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\right]\right\|_2 \leq \sigma^2$$

and the probability that there exists (k,t) with

$$\left\|\sum_{(\hat{k},\hat{t})\leq(k,t)} \mathbb{E}_{<(\hat{k},\hat{t})}\left[\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top} + \widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\right]\right\|_2 > \sigma^2 .$$

Each call to SPARSIFYEULERIAN is made with error probability parameter δ/P and by Lemma V.2, we call the routine at most $O(P)$ times, so by a union bound and Theorem II.1, the probability at some call fails is at most $O(\delta)$, which bounds the term (4). The other two conditions rely on truncated Martingales, which rely on the condition $\mathbf{SAFE}^{(k,t)}$. Lemma V.4 allows us to bound the $\widehat{\mathbf{X}}^{(k,t)}$.

Lemma V.6. *We have $\|\widehat{\mathbf{X}}^{(k,t)}\|_2 \leq O(1/P)$ over the entire support of $\widehat{\mathbf{X}}^{(k,t)}$ unconditionally.*

Lemma V.6 says that the steps of $\widehat{\mathbf{M}}^{(k,t)}$ have norm bounded by $O(1/P)$. This means that Lemma V.5 gives that for $\sigma^2 = \frac{\Theta(\log(1/\delta))}{P}$ and $s = \epsilon^2$, using $P = \Theta(\epsilon^{-2}\log^2(1/\delta))$ and $\log(1/\delta) \geq \Omega(\log n)$, the last term in the sum is upper bounded by $n\exp(\frac{-t^2}{100(\sigma^2+t\rho)}) = O(\delta)$.

Lemma V.7. *The probability that there exists (k,t) such that $\|\sum_{(\hat{k},\hat{t})\leq(k,t)}\mathbb{E}_{<(\hat{k},\hat{t})}\left[\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top} + \widehat{\mathbf{X}}^{(\hat{k},\hat{t})\top}\widehat{\mathbf{X}}^{(\hat{k},\hat{t})}\right]\|_2 > \sigma^2$ is at most $O(\delta)$.*

The proof of Theorem II.2 follows from the above reasoning and these lemmas.

ACKNOWLEDGMENTS

M.B.C. and J.K: This material is based upon work supported by the National Science Foundation under Grant No. 1111109. R.K: This material is based upon work supported by ONR Award N00014-16-1-2374. J.P.: This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1122374 and by the National Science Foundation under Grant No. 1065125. R.P.: This material is based upon work supported by the National Science Foundation under Grant No. 1637566.

REFERENCES

[1] D. A. Spielman and S.-H. Teng, “Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems,” in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 81–90.

[2] Y. T. Lee and A. Sidford, “Path finding methods for linear programming: Solving linear programs in $\tilde{O}\sqrt{rank}$ iterations and faster algorithms for maximum flow,” in *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014*, 2014, pp. 424–433.

[3] A. Madry, “Computing maximum flow with augmenting electrical flows,” in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016, pp. 593–602.

[4] ——, “Navigating central path with electrical flows: From flows to matchings, and back,” in *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, 2013, pp. 253–262.

[5] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng, “Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs,” in *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, ser. STOC ’11, 2011, pp. 273–282.

[6] A. Madry, D. Straszak, and J. Tarnawski, “Fast generation of random spanning trees and the effective resistance metric,” in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2015, pp. 2019–2036.

[7] M. B. Cohen, A. Mądry, P. Sankowski, and A. Vladu, “Negative-weight shortest paths and unit capacity minimum cost flow in Õ(m10/7 log w) time: (extended abstract),” in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’17, 2017, pp. 752–771.

[8] J. A. Kelner, G. L. Miller, and R. Peng, “Faster approximate multicommodity flow using quadratically coupled flows,” in *Proceedings of the 44th symposium on Theory of Computing*, ser. STOC ’12, 2012, pp. 1–18.

[9] J. A. Kelner and A. Madry, “Faster generation of random spanning trees,” in *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, 2009, pp. 13–21.

[10] D. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.

[11] S. I. Daitch and D. A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008*, 2008, pp. 451–460.

[12] J. Sherman, "Breaking the multicommodity flow barrier for o(vlog n)-approximations to sparsest cut," in *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, 2009, pp. 363–372.

[13] J. Kelner and P. Maymounkov, "Electric routing and concurrent flow cutting," *Theor. Comput. Sci.*, vol. 412, no. 32, pp. 4123–4135, Jul. 2011.

[14] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the lanczos method and an $\tilde{o}(m)$-time spectral algorithm for balanced separator," in *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012*, 2012, pp. 1141–1160.

[15] D. A. Spielman and S.-H. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014, available at http://arxiv.org/abs/cs/0607105.

[16] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving SDD linear systems," in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, ser. FOCS '10, 2010, pp. 235–244.

[17] ——, "A nearly-m log n time solver for SDD linear systems," in *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, ser. FOCS '11, 2011, pp. 590–598.

[18] 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 45th Annual Symposium on Theory of Computing*, ser. STOC '13. New York, NY, USA: ACM, 2013, pp. 911–920.

[19] Y. T. Lee and A. Sidford, "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 147–156.

[20] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu, "Solving SDD linear systems in nearly $m\log^{1/2}n$ time," in *Symposium on Theory of Computing, STOC 2014*, 2014, pp. 343–352.

[21] R. Peng and D. A. Spielman, "An efficient parallel solver for SDD linear systems," in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, ser. STOC '14, 2014, pp. 333–342.

[22] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 2016, pp. 842–850, available at http://arxiv.org/abs/1512.01892.

[23] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for laplacians - fast, sparse, and simple," in *Proceedings of the 57th annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016, available at https://arxiv.org/pdf/1605.02353v1.pdf.

[24] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014*, 2014, pp. 217–226.

[25] D. Durfee, R. Kyng, J. Peebles, A. B. Rao, and S. Sachdeva, "Sampling random spanning trees faster than matrix multiplication," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017*, 2017, pp. 730–742.

[26] A. Madry, "Fast approximation algorithms for cut-based problems in undirected graphs," in *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, 2010, pp. 245–254.

[27] I. Abraham, D. Durfee, I. Koutis, S. Krinninger, and R. Peng, "On fully dynamic graph sparsifiers," in *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016, pp. 335–344.

[28] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, Oct 2016, pp. 583–592.

[29] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, "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*, 2017, pp. 410–419.

[30] Y. T. Lee, R. Peng, and D. A. Spielman, "Sparsified cholesky solvers for SDD linear systems," *CoRR*, vol. abs/1506.08204, 2015. [Online]. Available: http://arxiv.org/abs/1506.08204

[31] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, 2016, pp. 842–850.

[32] J. A. Tropp, "Freedman's inequality for matrix martingales," *arXiv preprint arXiv:1101.3039*, 2011, available at: https://arxiv.org/abs/1101.3039.