

Almost-Linear-Time Algorithms for Markov Chains and New Spectral Primitives for Directed Graphs

Michael B. Cohen*	Jonathan Kelner*	John Peebles ⁺	Richard Peng [‡]
MIT	MIT	MIT	Georgia Tech
micohen@mit.edu	kelner@mit.edu	jpeebles@mit.edu	rpeng@cc.gatech.edu
Anup B. Rao	Aaron Sidford	Adrian Vladu*	
Georgia Tech	Stanford University	MIT	
anup.rao@gatech.edu	sidford@stanford.edu	avladu@mit.edu	

Abstract

In this paper we introduce a notion of spectral approximation for directed graphs. While there are many potential ways one might define approximation for directed graphs, most of them are too strong to allow sparse approximations in general. In contrast, we prove that for our notion of approximation, such sparsifiers do exist, and we show how to compute them in almost linear time.

Using this notion of approximation, we provide a general framework for solving asymmetric linear systems that is broadly inspired by the work of [Peng-Spielman, STOC'14]. Applying this framework in conjunction with our sparsification algorithm, we obtain an almost-linear-time algorithm for solving directed Laplacian systems associated with Eulerian Graphs. Using this solver in the recent framework of [Cohen-Kelner-Peebles-Peng-Sidford-Vladu, FOCS'16], we obtain almost linear time algorithms for solving a directed Laplacian linear system, computing the stationary distribution of a Markov chain, computing expected commute times in a directed graph, and more.

For each of these problems, our algorithms improves the previous best running times of $O((nm^{3/4} + n^{2/3}m) \log^{O(1)}(n\kappa\epsilon^{-1}))$ to $O((m + n2^{O(\sqrt{\log n \log \log n})}) \log^{O(1)}(n\kappa\epsilon^{-1}))$ where n is the number of vertices in the graph, m is the number of edges, κ is a natural condition number associated with the problem, and ϵ is the desired accuracy. We hope these results open the door for further studies into directed spectral graph theory, and that they will serve as a stepping stone for designing a new generation of fast algorithms for directed graphs.

*This material is based upon work supported by the National Science Foundation under Grant No. 1111109.

⁺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.

[‡]This material is based upon work supported by the National Science Foundation under Grant No. 1637566.

1 Introduction

In the analysis of Markov chains, there has been a longstanding algorithmic gap between the general case, corresponding to random walks on directed graphs, and the special case of reversible chains, for which the corresponding graph can be taken to be undirected. This gap begins with the most basic computational task—computing the stationary distribution—and persists for many of the fundamental problems associated with random walks, such as computing hitting and commute times, escape probabilities, and personalized PageRank vectors. In the undirected case, there are algorithms for all of these problems that run in linear or nearly-linear time. In the directed case, however, the best algorithms have historically been much slower. Specifically, the best running times were given by a recent precursor to the present paper [12], which showed that one could solve these problems on a graph with n vertices and m edges in time $\tilde{O}(nm^{3/4} + n^{2/3}m)$.¹ Prior to that work, it was unknown whether one could solve any of them faster than the time needed to solve an arbitrary linear system with the given size and sparsity, i.e. $\Theta(\min(mn, n^\omega))$ time, where $\omega < 2.3729$ is the exponent for matrix multiplication.

This gap has its origins in a broader discrepancy between the state of algorithmic spectral graph theory in undirected and directed settings. While the undirected case has a richly developed theory and a powerful collection of algorithmic tools, similar results have remained somewhat elusive for directed graphs. In particular, the problems mentioned above can be expressed in terms of the linear algebraic properties of the Laplacian matrix of a graph, and it was shown in [12] how to reduce all these problems to the solution of a small number of Laplacian linear systems. In the undirected case, there has been a tremendously successful line of research on how to use the combinatorial properties of graphs to accelerate the solution of such systems, culminating in algorithms that can solve them in nearly-linear time [37, 23, 24, 22, 27, 13, 33, 25, 26]. Unfortunately, these solvers relied heavily on several features that seemed intrinsic to the undirected case and did not appear to be available for directed graphs, thereby precluding an analogous solver for directed Laplacians. In particular, the undirected solvers relied on:

Knowledge of the kernel/stationary distribution: Up to a simple rescaling by the vertex degrees, vectors in the kernel of a Laplacian correspond to stationary distributions of the corresponding random walk. For undirected graphs, the kernel is spanned by the all-ones vector on each of the connected components, so it and the space of stationary distributions can be easily computed in linear time. For directed graphs, however, this is no longer the case, and finding the stationary distribution does not seem to be any easier than the original problem of solving Laplacian linear systems. In fact, while stationary distributions of random walks on directed graphs have been studied for over 100 years [3], and computing them has been extensively investigated in both theory and practice (see e.g. [38, 34]), the $\tilde{O}(nm^{3/4} + n^{2/3}m)$ result in [12] was the first to find them in less time than is required to find the kernel of a general matrix.

Symmetry and positive semidefiniteness: Undirected Laplacians are symmetric and positive semidefinite. Essentially every aspect of algorithmic spectral graph theory uses this symmetry to treat the Laplacian as a quadratic form and relies on its expression as a sum of positive semidefinite contributions from each of the edges to analyze its properties. This includes the Laplacians’ connection to the graph’s cut structure, their relationship to electrical circuits and effective resistances, the notion of graph inequalities and spectral approximation, the combinatorial

¹ We use \tilde{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” to refer to algorithms whose running time is $\tilde{O}(m) = m \log^{O(1)}(n\kappa\epsilon^{-1})$ and “almost linear” to refer to algorithms that are linear up to sub-polynomial (but possibly super-logarithmic) factors, i.e., whose running time is $O(m(n\kappa\epsilon^{-1})^{o(1)})$.

construction of preconditioners, and the iterative methods used to solve Laplacian systems. On the other hand, directed Laplacians are asymmetric matrices, and their naive symmetrizations are not typically positive semidefinite.

Sparsification One of the most powerful algorithmic tools in the undirected setting is the ability to construct *sparsifiers* [6, 36, 5]. These allow one to approximate an arbitrarily dense graph by a sparse graph that has only a slightly super-linear number of edges. The classical notion of cut sparsification requires that the value of every cut in the original graph be approximately preserved in the sparsifier; the more recent notion of spectral sparsification is stronger, and also implies the former property. For directed graphs, it can be shown that, even for the weaker notion of cut sparsification, such sparsifiers do not generally exist. One simple example is the complete bipartite graph. (See Section 1.1.2.) In fact, it was not known how to define any other useful notion of sparsification for which this would not be the case.

In this paper, we show how to cope with these fundamental differences, and begin to address the algorithmic gap between general and reversible Markov chains. Our core technical result is the first almost-linear-time solver for directed Laplacian systems. Using the work from [12], this yields the first almost-linear-time algorithms for computing a host of fundamental objects and quantities associated with a random walk on a directed graph, including the stationary distribution, hitting and commute times, escape probabilities, and personalized PageRank vectors.

More broadly, constructing our solver required the development of directed versions of several foundational tools and techniques from undirected algorithmic spectral graph theory. Most notably, and perhaps surprisingly, we show that it is possible to develop a useful notion of spectral approximation and sparsification of directed graphs, and that sparsifiers under this definition exist and can be constructed efficiently.

In addition to their direct application to the analysis of Markov chains, we hope that both the solver itself and the sparsification machinery will prove to be useful tools in the further development of fast graph algorithms. In the undirected case, sparsifiers have been a core algorithmic tool since the early 1990s [6, 19, 20, 16, 1], and fast solvers for undirected Laplacian solvers have recently led to an explosion of algorithms operating in the so-called “Laplacian Paradigm” [39], in both cases leading to asymptotic improvements for many of the core algorithmic problems for undirected graphs. Given the success these methods have enjoyed in the case of undirected graphs, we hope that their directed analogues will spark similar progress in the directed setting.

1.1 Previous Work

In this section, we briefly review some of the previous work related to our results and techniques. Given the extensive prior research on Markov chains, spectral graph theory, sparsification, solving general and Laplacian linear systems, and computing PageRank, we do not attempt to give a comprehensive overview of the literature; instead we simply describe the work that most directly relates to or motivates this paper.

1.1.1 Directed Laplacian Systems, Stationary Distributions, and PageRanks

The most direct precursor to this work is a recent paper by a subset of the authors [12]. As mentioned above, it showed that, by exploiting linear algebraic properties of directed Laplacians, one could obtain faster algorithms for a wide range of problems involving directed random walks. Prior to this paper, it seemed quite possible that the similarities between directed and undirected Laplacians were largely syntactic, and that there was no way to use the structure of directed

Laplacians or random walks to obtain asymptotically faster algorithms. In particular, despite extensive theoretical and applied work in computer science, mathematics, statistics, and numerical scientific computing, all algorithms that we are aware of prior to [12] for obtaining high-quality² solutions for directed Laplacian systems, stationary distributions, or personalized PageRank vectors either have a polynomial dependence on a condition number or related parameter (such as a random walk’s mixing time or PageRank’s restart probability), or they apply a general-purpose linear algebra routine and thus run in at least the $\Omega(\min(mn, n^\omega))$ time currently required to solve arbitrary linear systems.

By showing that this was not the case, [12] provided the first indication that one could actually use the structure of directed Laplacian systems to accelerate their solution, which provided a strong motivation to see how much of an improvement was possible. It also created hope that the recently successful research program in building and applying fast algorithms for solving (symmetric) Laplacian systems [37, 23, 24, 22, 27, 33, 25] could be applied to give more direct improvements to running times for solving combinatorial optimization problems on directed graphs.

In addition to motivating the search for faster Laplacian solvers, [12] provided a set of reductions that we will directly apply in this paper. In order to prove its results, [12] showed how to reduce a range of algorithmic questions about directed walks, such as computing the stationary distribution, hitting and commute times, escape probabilities, and personalized PageRank vectors, to solving a small number of linear systems in directed Laplacians.

It turns out that it is easier to work with such systems in the special case where the graph is Eulerian. One of the main technical tools in [12] is a reduction to this special case. They did this by giving an iterative method that solved a general Laplacian system by solving a small number of systems in which the graph is Eulerian. Together, this showed that to solve the aforementioned problems, it suffices to give a solver for Eulerian graphs, and that this only incurs a factor of $\tilde{O}(1)$ overhead. It then obtained all of its results by constructing an Eulerian solver that runs in time $\tilde{O}(m^{3/4}n + mn^{2/3})$. In this paper we construct an Eulerian solver that runs in time $m^{1+o(1)}$ and then just directly apply these reductions to obtain our other results.

However, while [12] opened the door for further algorithmic improvements in analyzing Markov chains, the arguments in it provided little evidence that the running time could be improved to anything approaching what was known in the undirected case. Indeed, while the techniques in it suggested that it might be possible to obtain further improvements, even the most optimistic interpretations of the structural results in [12] only gave hope for achieving running times of roughly $\tilde{O}(m\sqrt{n})$. This would make it no faster than some of the existing algorithms that use undirected Laplacian solvers to solve problems on directed graphs, such as the $\tilde{O}(m^{10/7})$ algorithms for unit cost maximum flow [30, 31] and shortest path with negative edge lengths [14], or the $\tilde{O}(m\sqrt{n})$ type bounds for minimum cost flow [28]. As such, while this would provide better results for the applications to Markov chains, it would rule out the hope of obtaining improved results for these directed problems by replacing the undirected solver with a directed one.

Intuitively, the solver in [12] worked by showing how one could use the existence of a fast undirected solver to solve directed Laplacians. For a directed Eulerian Laplacian \mathcal{L} , it showed that the symmetrized matrix $\mathbf{U} = (\mathcal{L} + \mathcal{L}^\top)/2$ is the Laplacian of an undirected graph, and that the symmetric matrix $\mathcal{L}^\top \mathbf{U}^+ \mathcal{L}$ was, in a certain sense, reasonably well approximated by \mathbf{U} . Given a linear system $\mathcal{L}\vec{x} = \vec{b}$, one could then form the equivalent system $\mathcal{L}^\top \mathbf{U}^+ \mathcal{L}\vec{x} = \mathcal{L}^\top \mathbf{U}^+ \vec{b}$ and use a fast undirected Laplacian solver to apply \mathbf{U}^+ . One could then hope that the fact that the matrix on the left is somewhat well-approximated by \mathbf{U} would imply that \mathbf{U}^+ is a sufficiently good preconditioner

²By high-quality, we mean that the algorithm should be able to find a solution with error ϵ in time that is sub-polynomial in $1/\epsilon$, i.e. $(1/\epsilon)^{o(1)}$. For PageRank there were some known techniques for achieving better dependence on n and m at the expense of a polynomial dependence on $1/\epsilon$ [2, 11, 9, 10].

for it to yield an improved running time. It turned out that, while this would actually be the case in exact arithmetic, numerical issues provided a legitimate obstruction. This necessitated a more involved scheme, which gave a slightly slower running time of $\tilde{O}(m^{3/4}n + mn^{2/3})$, rather than the roughly $\tilde{O}(n^{2/3}m)$ running time that what would have been achieved by exact arithmetic.

The way this algorithm works provides a good intuitive explanation for why one would not expect it to give a solver yielding substantial improvements for combinatorial ‘‘Laplacian Paradigm’’ algorithms that rely on undirected solvers. At its root, the solver from [12] works by trying to find the right way to ignore the directed structure and solve the system with an undirected solver; thus it is on essentially the same footing as the algorithms it would hope to improve. The obstructions it faces are rooted in the fact that directed Laplacians are fundamentally not very well-approximated by undirected ones. In essence, the difference between the solver in this paper and the one presented in [12] is that, instead of figuring out how to properly neglect the directed structure, the solver we present here intrinsically works with asymmetric (directed) objects, and redevelops the theory from the ground up to properly capture them.

1.1.2 Directed Graph Sparsification and Approximation

While sparsification of undirected graphs has been extensively studied [6, 36, 16, 35, 4, 5, 42, 29], there has been very little success extending the notion to directed graphs. In fact, it was not even clear that there should exist a useful definition under which directed graphs should have sparsifiers with a subquadratic number of edges, and for many of the natural definitions one would propose, sparsification is provably impossible.

For instance, a natural first attempt would be to try to generalize the classical notion of cut sparsification for undirected graphs [18, 6]. Given any weighted undirected graph G , Benczur and Karger showed that one could construct a new graph H with at most $O(n \log n / \epsilon^2)$ edges such that the value of every cut in G is within a multiplicative factor of $1 \pm \epsilon$ of its value in H . While this definition makes sense for directed graphs as well, there is no analogous existence result. Indeed it is not hard to construct directed graphs for which any such approximation must have $\Omega(n^2)$ edges.

For example, consider the directed complete bipartite graph K on the vertex set $A \cup B$ with all edges directed from A to B . For each pair of $a \in A$ and $b \in B$, the directed cut

$$E(\{a\} \cup B \setminus \{b\}, \{b\} \cup A \setminus \{a\}) \tag{1.1}$$

contains only the edge $a \rightarrow b$. (See Figure 1.1.) Removing this edge from the graph would change the value of this cut from 1 to 0, resulting in an infinite multiplicative error.

Any graph that multiplicatively approximates the cuts in K must have $|E(B, A)| = 0$, so it must be supported on a subset of the edges of K , and the above then shows that such a graph must contain the edge $a \rightarrow b$ for every $a \in A$ and $b \in B$. It thus follows that any graph that approximately preserves every cut in K must contain all $|A||B|$ potential edges, so K has no nontrivial sparsifier under this definition.

It would therefore seem that any attempt at reducing the number of edges in a directed graph while preserving the combinatorial structure is doomed to fail. However, Eulerian graphs present a natural setting that circumvents this because cuts in Eulerian graphs have the same amount of edge weight going in each direction, the bipartite graph counterexamples above are precluded. This balancedness allows one to incorporate sparsification based tools for flows and routings in this setting to solve combinatorial flow and cut problems quickly on Eulerian graphs [15].

Most closely related to our notion of sparsification of directed graphs is the work by Chung on Cheeger’s inequality for directed graphs [8]. This result transforms the graph into an Eulerian graph G in a way identical to how we obtain Eulerian graphs [12]: by rescaling each edge weight by the

probability of its source vertex in a stationary distribution. It then relates the convergence rate of random walks on G to the eigenvalues of the undirected graph obtained by removing directions on all edges. Specifically if the Eulerian directed Laplacian is \mathcal{L} , this symmetrization is $(\mathcal{L} + \mathcal{L}^\top)/2$.

Since the eigenvalues of the symmetrization of an Eulerian graph give information about random walks on the original graph, it might be tempting to define approximation for Eulerian graphs in terms of whether their symmetrizations approximate each other in the conventional positive semidefinite sense. For our purposes, we require (and obtain) a substantially stronger notion of approximation that preserves much of the directed structure that would be erased by symmetrizing. The reason why we need a stronger notion of approximation is that we want graphs that approximate each other under this notion to be good preconditioners of one another. In contrast, if one defines approximation according to whether the symmetrizations approximate one another, one would have to say that the length n undirected cycle and the length n directed cycle approximate each other, since they are both Eulerian and have the same undirected symmetrization. However, they are not good preconditioners of one another, and using one as a substitute for the other would incur very large losses in our applications. Under the notion of approximation we introduce in this paper, these graphs differ by a factor of $\Omega(n^2)$.

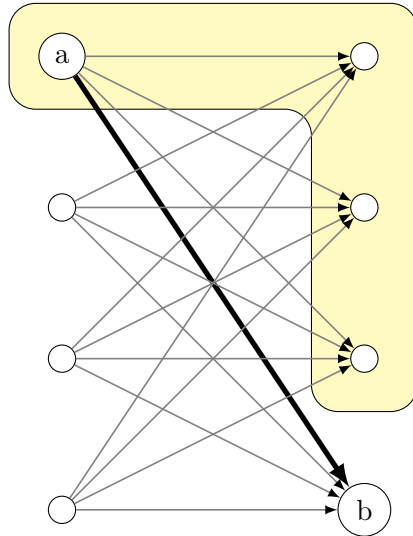


Figure 1.1: An example of the family of cuts described in Equation (1.1). The only edge leaving the highlighted set is $a \rightarrow b$, so any sparsifier that omits it will fail to approximate the corresponding cut.

1.1.3 Laplacian System Solvers

Our algorithms build heavily on the literature for solving undirected Laplacian systems. Since undirected Laplacians are special cases of directed Laplacians, any directed solver will yield an undirected solver when given a symmetric input. It is thus helpful to consider what undirected solver we would like our method to resemble in this case.

There are now a fairly large number of reasonably distinct algorithms for solving such systems, and we believe that several of them provide a template that could be turned into a working directed solver. Of these, the one that our solver most closely resembles is the parallel solver by Peng and Spielman [33], which we will briefly summarize here.

To simplify the notation and avoid having to keep track of degree normalizations, we only consider regular graphs when giving the intuition behind the algorithm. Suppose that we are given a d -regular undirected graph G with Laplacian $\mathcal{L} = d\mathbf{I} - \mathbf{A} = d(\mathbf{I} - \mathcal{A})$, where $\mathcal{A} = \mathbf{A}/d$ has $\|\mathcal{A}\| < 1$ on $\ker(\mathcal{L})^\perp$. For simplicity, in the equations that follow, we restrict our attention to the space perpendicular to the kernel of \mathcal{L} . With this convention, the algorithm of [33] is then motivated by the series expansion

$$(\mathbf{I} - \mathcal{A})^{-1} = \sum_{i \geq 0} \mathcal{A}^i = \prod_{k \geq 0} (\mathbf{I} + \mathcal{A}^{2^k}), \quad (1.2)$$

which is a matrix version of the standard scalar identity $1/(1-x) = 1 + x + x^2 + x^3 + \dots = 1(1+x)(1+x^2)(1+x^4)\dots$. If λ is the smallest nonzero eigenvalue of $\mathbf{I} - \mathcal{A}$, then truncating this product at $k = \Theta(\log 1/\lambda)$ yields a constant relative error, which can be made arbitrarily small

by further increasing k . Hence if $\lambda > 1/\text{poly}(n)$, we obtain a small error by multiplying the first $O(\log n)$ terms of the product. This seems to suggest a good algorithm for solving a system $\mathcal{L}\vec{x} = \vec{b}$: simply compute $\mathbf{I} + \mathcal{A}^{2^k}$ for $k = 0, \dots, t = O(\log n)$ and then return $\frac{1}{d}(\mathbf{I} + \mathcal{A}^{2^0}) \cdots (\mathbf{I} + \mathcal{A}^{2^t})\vec{b}$.

Unfortunately, this algorithm (implemented naively) would be too slow. As k grows, \mathcal{A}^k quickly becomes dense, so computing it requires repeatedly squaring dense matrices, which takes time $O(n^\omega)$. To deal with this, their algorithm instead replaces these matrices with sparse approximations of them. Peng and Spielman showed that given a graph with n vertices and m edges, one can compute a sparse approximation of the requisite squared matrix in nearly-linear-time.

Making this idea work requires care, since in general it is not true that the product of two matrices will be well approximated by the product of their approximations. For positive semidefinite matrices, however, there is a variant of this statement that holds if one takes the products symmetrically: if \mathbf{A} and \mathbf{B} are PSD and \mathbf{A} is a good approximation of \mathbf{B} , then for any matrix \mathbf{V} , $\mathbf{V}^\top \mathbf{A} \mathbf{V}$ is a good approximation of $\mathbf{V}^\top \mathbf{B} \mathbf{V}$. This led the authors of [33] to work with a more stable symmetric version of the series described above, which allowed them to obtain their result.

This turns out to be a reasonably convenient template for our directed solver. In particular, it has fewer moving parts than many of the other methods, and it does not require constructing combinatorial objects, like low-stretch spanning trees. Instead it directly relies on sparsification, which is our main new technical tool for directed graphs.

Unfortunately, we cannot directly apply the methods described above, since the symmetric product constructions that are used to control the error are no longer available for the (asymmetric) Laplacians of directed graphs. Moreover, the strong notions of graph approximation and positive semidefinite inequalities that facilitate the analysis for the undirected solver are unavailable in the directed setting. As such, we end up having to work with weaker error guarantees, and correct the extra error they introduce using a more involved iterative method.

1.2 Our Results

In this paper, we show that, in spite of these seemingly fundamental differences between the directed and undirected settings, we can develop directed analogues of several of the core spectral primitives that have been deployed to great effect on undirected graphs, and we use them to obtain the first almost-linear-time algorithms for many of the central problems in the analysis of non-reversible Markov chains. The main new theoretical tools and algorithmic primitives we introduce are:

- **Directed graph approximation:** We develop a well-behaved notion of spectral approximation for directed graphs, despite the fact that the corresponding Laplacians lack the symmetry and positive semidefiniteness properties that the undirected version crucially relies on. Our definition specializes to the standard version based on PSD matrix inequalities when applied to undirected graphs, and it retains many of the useful features of the undirected definition. For example, our notion of graph approximations roughly preserve the behavior of random walks, behave well under composition and change of basis, retain certain key aspects of the combinatorial structure, and provide good preconditioners for iterative methods.
- **Directed sparsification:** We show that, under our notion of approximation, any strongly-connected directed graph can be approximated by a *sparsifier* with only $\tilde{O}(n/\epsilon^2)$ edges, and we give an algorithm to compute such a sparsifier in almost-linear time. To our knowledge, this is the first time that directed sparsifiers with $o(n^2)$ edges have been proven to exist, even non-algorithmically, for any computationally useful definition that retains the directed structure of a graph.

- **Almost-linear-time solvers for directed Laplacian systems:** Given the Laplacian $\mathcal{L} = \mathbf{D} - ma^\top$ of a directed graph with n vertices and m edges, we provide an algorithm that leverages our sparsifier construction to solve the linear system $\mathcal{L}\vec{x} = \vec{b}$ in time

$$\mathcal{T} = O\left(m + n2^{O(\sqrt{\log n \log \log n})}\right) \log^{O(1)}(n\kappa\epsilon^{-1}) = O\left(m + n^{1+o(1)}\right) \log^{O(1)}(n\kappa\epsilon^{-1}), \quad (1.3)$$

where $\kappa = \max(\kappa(\mathcal{L}), \kappa(\mathbf{D}))$ is the maximum of the condition numbers of \mathcal{L} and \mathbf{D} , improving on the best previous running time of $O(nm^{3/4} + n^{2/3}m) \log^{O(1)}(n\kappa\epsilon^{-1})$. (See Theorem 4.1.) To do so, we introduce a novel iterative scheme and analysis that allows us to mitigate the accumulation of errors from multiplying sparse approximations without having access to the more stable constructions and bounds available for symmetric matrices.

In [12], we provided a suite of reductions that used a solver for directed Laplacians to solve a variety of other problems. Plugging our new solver’s running time into these reductions immediately gives the following almost-linear-time algorithms:³

- **Computing stationary distributions:** We can compute a vector within ℓ_2 distance ϵ of the stationary distribution of a random walk on a strongly connected directed graph in time \mathcal{T} .
- **Computing Personalized PageRank vectors:** We can compute a vector within ℓ_2 distance ϵ of the Personalized PageRank vector with restart probability β for a directed graph in time $\mathcal{T} \log^2(1/\beta)$.
- **Simulating random walks:** We can compute the escape probabilities, commute times, and hitting times for a random walk on a directed graph and estimate the mixing time of a lazy random walk up to a polynomial factor in time \mathcal{T} .
- **Estimating all-pairs commute times:** We can build a data structure of size $\tilde{O}(n\epsilon^{-2} \log n)$ in time \mathcal{T}/ϵ^2 that, when queried with any two vertices a and b , outputs a $1 \pm \epsilon$ multiplicative approximation to the expected commute time between a and b .
- **Solving row- and column-diagonally dominant linear systems:** We can solve linear systems that are row- or column-diagonally dominant in time $\mathcal{T} \log K$, where K denotes the ratio of the largest and smallest diagonal entries.

This gives the first almost-linear-time algorithm for each of these problems. For all of them, the best previous running time for obtaining high-quality solutions was what is obtained by replacing \mathcal{T} with $O(nm^{3/4} + n^{2/3}m) \log^{O(1)}(n\kappa\epsilon^{-1})$ and was proven in [12].

1.3 Paper Overview

The rest of this paper is organized as follows.

- Section 2 – we cover preliminaries such as notation, facts about directed Laplacians that we use throughout the paper, and an overview of our approach.

³We use \mathcal{T} to denote anything of the form given in equation 1.3, not the time required for one call to the solver. Some of the reductions call the solver a logarithmic number of times, so precise value of the $\log^{O(1)}(n\kappa\epsilon^{-1})$ term varies among the applications. Also, note that in this paper we give solving running times in terms of the condition number of symmetric Laplacian whereas in [12] they are often given in terms of the condition number of the corresponding diagonal matrix. However it is well-known that these differ only by a $O(\text{poly}(n))$ factor and as they are in the logarithmic terms, this does not affect the running times.

- Section 3 – we introduce a notion of asymmetric approximation, and prove that we can, in nearly linear time, produce sparsifiers which are good approximations under this notion.
- Section 4 – we show how to employ the sparsification routines from the previous section in order to obtain our fast Eulerian Laplacian system solver.
- Appendix A – we prove a matrix concentration result concerning entrywise sampling, which is the basic building block for the results from Section 3.
- Appendix B – we provide various general linear algebra facts used throughout the paper.
- Appendix C – we show how to obtain the graph decompositions required for sparsifying arbitrary Eulerian Laplacians using a decomposition of undirected graphs into expanders.
- Appendix D – we provide some details on the full algorithm for computing stationary distributions and solving directed Laplacians using a Eulerian Laplacian system solver.
- Appendix E – we relate our sparsification results to certain systems considered in [12].
- Appendix F – we provide a general reduction that improves the dependence of our Eulerian solver on the condition number from $\exp(\sqrt{\log \kappa})$ to $\log \kappa$.

2 Preliminaries

First we give notation in Section 2.1 and then we give basic information about directed Laplacians in Section 2.2. Much of this is inherited from [12]. With this notation in place we give an overview of our approach in Section 2.3.

2.1 Notation

Matrices: We use bold to denote matrices and let $\mathbf{I}, \mathbf{0} \in \mathbb{R}^{n \times n}$ denote the identity matrix and zero matrix respectively. For a matrix \mathbf{A} we use $\text{nnz}(\mathbf{A})$ to denote the number of non-zero entries in \mathbf{A} . When $\mathbf{A} \in \mathbb{R}^{n \times n}$ we use $\text{supp}(\mathbf{A})$ to denote the subset of $[n]$ corresponding to the indices for which at least one of the corresponding row or column in \mathbf{A} is non-zero.

Vectors: We use the vector notation when we wish to highlight that we are representing a vector. We let $\mathbf{0}, \mathbf{1} \in \mathbb{R}^n$ denote the all zeros and ones vectors, respectively. We use $\vec{\mathbf{1}}_i \in \mathbb{R}^n$ to denote the i -th basis vector, i.e. $(\vec{\mathbf{1}}_i)_j = 0$ for $j \neq i$ and $(\vec{\mathbf{1}}_i)_i = 1$. Occasionally, when it is obvious from the context, we apply scalar operations to vectors with the interpretation that they be applied coordinate-wise. As with matrices, we use $\text{supp}(\vec{x})$ to denote the indices of \vec{x} with non-zero entries.

Positive Semidefinite Ordering: For symmetric matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ we use $\mathbf{A} \preceq \mathbf{B}$ to denote the condition that $x^\top \mathbf{A} x \leq x^\top \mathbf{B} x$, for all x . We define $\succeq, \prec,$ and \succ analogously. We call a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ positive semidefinite (PSD) if $\mathbf{A} \succeq \mathbf{0}$. For vectors x , we let $\|x\|_{\mathbf{A}} \stackrel{\text{def}}{=} \sqrt{x^\top \mathbf{A} x}$. For asymmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ we let $\mathbf{U}_{\mathbf{A}} \stackrel{\text{def}}{=} \frac{1}{2}(\mathbf{A} + \mathbf{A}^\top)$ and note that $x^\top \mathbf{A} x = x^\top \mathbf{A}^\top x = x^\top \mathbf{U}_{\mathbf{A}} x$ for all $x \in \mathbb{R}^n$.

Operator Norms: For any norm $\|\cdot\|$ defined on vectors in \mathbb{R}^n we define the *seminorm* it induces on $\mathbb{R}^{n \times n}$ by $\|\mathbf{A}\| = \max_{x \neq 0} \frac{\|\mathbf{A}x\|}{\|x\|}$ 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., $\|\mathbf{A}\|_{\mathbf{H} \rightarrow \mathbf{H}} = \max_{x \neq 0} \frac{\|\mathbf{A}x\|_{\mathbf{H}}}{\|x\|_{\mathbf{H}}}$, but we may

also simply write $\|\mathbf{A}\|_{\mathbf{H}} \stackrel{\text{def}}{=} \|\mathbf{A}\|_{\mathbf{H} \rightarrow \mathbf{H}}$ in this case. For symmetric positive definite \mathbf{H} we have that $\|\mathbf{A}\|_{\mathbf{H} \rightarrow \mathbf{H}}$ can be equivalently expressed in terms of $\|\cdot\|_2$ as $\|\mathbf{A}\|_{\mathbf{H} \rightarrow \mathbf{H}} = \|\mathbf{H}^{1/2} \mathbf{A} \mathbf{H}^{-1/2}\|_2$. Also note that $\|\mathbf{A}\|_1$ is the maximum ℓ_1 norm of a column of \mathbf{A} , and $\|\mathbf{A}\|_\infty$ is the maximum ℓ_1 norm of a row of \mathbf{A} .

Diagonals For $x \in \mathbb{R}^n$ we let $\mathbf{diag}(x) \in \mathbb{R}^{n \times n}$ denote the diagonal matrix with $\mathbf{diag}(x)_{ii} = x_i$ and typically use $\mathbf{X} \stackrel{\text{def}}{=} \mathbf{diag}(x)$. For $\mathbf{A} \in \mathbb{R}^{n \times n}$ we let $\mathbf{diag}(\mathbf{A}) \in \mathbb{R}^n$ denote the vector corresponding to the diagonal of \mathbf{A} , i.e. $\mathbf{diag}(\mathbf{A})_i = \mathbf{A}_{ii}$ and we let $\mathbf{diag}(\mathbf{A})$ denote the diagonal matrix having the same diagonal as \mathbf{A} .

Linear Algebra For a matrix \mathbf{A} , we let \mathbf{A}^+ denote the (Moore-Penrose) pseudoinverse of \mathbf{A} . For a symmetric positive semidefinite matrix \mathbf{B} , we let $\mathbf{B}^{1/2}$ denote the square root of \mathbf{B} , that is the unique symmetric positive semidefinite matrix such that $\mathbf{B}^{1/2} \mathbf{B}^{1/2} = \mathbf{B}$. Furthermore, we let $\mathbf{B}^{+/2}$ denote the pseudoinverse of the square root of \mathbf{B} . We use $\ker(\mathbf{A})$ to denote nullspace (kernel) of \mathbf{A} . We use $\text{span}(x_1, x_2, \dots, x_k)$ to denote the subspace spanned by x_1, \dots, x_k . For a symmetric PSD matrix \mathbf{A} we let $\lambda_*(\mathbf{A})$ denote the smallest non-zero eigenvalue of \mathbf{A} .

Misc: We let $[n] \stackrel{\text{def}}{=} \{1, \dots, n\}$. For $\mathbf{A} \in \mathbb{R}^{n \times n}$, let $\kappa(\mathbf{A}) \stackrel{\text{def}}{=} \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^+\|_2$ denote the condition number of \mathbf{A} . For symmetric PSD matrices \mathbf{A} and \mathbf{B} with the same kernel, let $\kappa(\mathbf{A}, \mathbf{B}) \stackrel{\text{def}}{=} \kappa(\mathbf{A}^{+/2} \mathbf{B} \mathbf{A}^{+/2})$ denote their relative condition number (e.g. if $\alpha \mathbf{B} \preceq \mathbf{A} \preceq \beta \mathbf{B}$ then $\kappa(\mathbf{A}, \mathbf{B}) \leq \beta/\alpha$). Note that our use of pseudoinverse rather than inverse in these definitions is non-standard but convenient.

2.2 Directed Laplacians

Here we provide notation regarding directed Laplacians and review basic facts regarding these matrices that were proved in [12]. We begin with some basic definitions and notation regarding Laplacians:

Directed Laplacian: A matrix $\mathcal{L} \in \mathbb{R}^{n \times n}$ is called a *directed Laplacian* if (1) its off diagonal entries are non-positive, i.e. $\mathcal{L}_{i,j} \leq 0$ for all $i \neq j$, and (2) it satisfies $\mathbf{1}^\top \mathcal{L} = \mathbf{0}$, i.e. $\mathcal{L}_{ii} = -\sum_{j \neq i} \mathcal{L}_{ji}$ for all i .

Associated Graph: To every directed Laplacian $\mathcal{L} \in \mathbb{R}^{n \times n}$ we associate a graph $G_{\mathcal{L}} = (V, E, w)$ with vertices $V = [n]$, and edges (i, j) of weight $w_{ij} = -\mathcal{L}_{ji}$, for all $i \neq j \in [n]$ with $\mathcal{L}_{ji} \neq 0$. Occasionally we write $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$ to denote that we decompose \mathcal{L} into the diagonal matrix \mathbf{D} (where $\mathbf{D}_{ii} = \mathcal{L}_{ii}$ is the out degree of vertex i in $G_{\mathcal{L}}$) and non-negative matrix \mathbf{A} (which is weighted adjacency matrix of $G_{\mathcal{L}}$, with $\mathbf{A}_{ij} = w_{ij}$ if $(i, j) \in E$, and $\mathbf{A}_{ij} = 0$ otherwise).

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

(Symmetric) Laplacian: A matrix $\mathbf{U} \in \mathbb{R}^{n \times n}$ is called a *symmetric* or *undirected Laplacian* or just a *Laplacian* if it is symmetric and a directed Laplacian. Note that the graph associated with an undirected Laplacian is undirected, i.e. for every forward edge there is a backward edge of the same weight. Given a symmetric Laplacian $\mathbf{U} = \mathbf{D} - \mathbf{A}$, we let its *spectral gap* be defined as the smallest nonzero eigenvalue of $\mathbf{D}^{-1/2} \mathbf{U} \mathbf{D}^{-1/2}$, i.e. $\lambda_2(\mathbf{D}^{-1/2} \mathbf{U} \mathbf{D}^{-1/2}) = \min_{x \perp \ker(\mathbf{D}^{-1/2} \mathbf{U} \mathbf{D}^{-1/2}), \|x\|=1} x^\top \mathbf{D}^{-1/2} \mathbf{U} \mathbf{D}^{-1/2} x$.

Running Times: Our central object is almost always a directed Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{n \times n}$, where $m = \text{nnz}(\mathbf{A})$, $U \stackrel{\text{def}}{=} \max_{i,j} |\mathbf{A}_{ij}| / \min_{i,j: \mathbf{A}_{ij} \neq 0} |\mathbf{A}_{ij}|$. We use $\tilde{O}(\cdot)$ notation to suppress factors

polylogarithmic in n , m , U , and κ , the natural condition number of the particular problem.

2.3 Overview of Approach

Here we provide an overview of our approach for solving linear systems in directed Laplacians. We split it into three parts. In the first part, Section 2.3.1, we describe how to reduce the problem to the special case of solving Eulerian Laplacians with polynomial condition number. In the second part, Section 2.3.2 we cover the efficient construction of sparsifiers. Finally, and in the third part, Section 2.3.3, we discuss how to use the sparsifier construction to build an almost-linear-time solver for polynomially well-conditioned Eulerian Laplacian systems.

2.3.1 Reductions

We begin by applying two reductions. The first is a result from [12], which states that one can solve row- and column-diagonally dominant linear systems, which include general directed Laplacian systems, by solving a small number of Laplacian systems in which the graphs are Eulerian:

Theorem 2.1 (Theorem 42 from [12]). *Let \mathbf{M} be an arbitrary $n \times n$ column-diagonally-dominant or row-diagonally-dominant matrix with diagonal \mathbf{D} . Let $b \in \text{im}(\mathbf{M})$. Then for any $0 < \epsilon \leq 1$, one can compute, with high probability and in time*

$$O\left(\mathcal{T}_{\text{solve}} \log^2\left(\frac{n \cdot \kappa(\mathbf{D}) \cdot \kappa(\mathbf{M})}{\epsilon}\right)\right)$$

a vector x' satisfying $\|\mathbf{M}x' - b\|_2 \leq \epsilon \|b\|_2$.

Furthermore, all the intermediate Eulerian Laplacian solves required to produce the approximate solution involve only matrices \mathbf{R} for which $\kappa(\mathbf{R} + \mathbf{R}^\top)$, $\kappa(\text{diag}(\mathbf{R})) \leq (n\kappa(\mathbf{D})\kappa(\mathbf{M})/\epsilon)^{O(1)}$.

If we were to combine this directly with the algorithm from Section 4, it would give a running time of $\tilde{O}\left((m + n \exp O(\sqrt{\log \kappa \cdot \log \log \kappa})) \log(1/\epsilon)\right)$ to solve linear systems in a directed Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$, where κ is the condition number of the normalized Laplacian $\mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{-1/2}$. While κ is typically polynomial in n , it is possible for it to be exponential, so we would like our running time to depend on it logarithmically, instead of just sub-polynomially. We show how to do this in Appendix F, where we give an algorithm to solve an arbitrarily ill-conditioned Eulerian Laplacian systems by solving $O(\log(n\kappa))$ Eulerian Laplacians whose condition numbers are polynomial in n . This allows us to restrict our attention for the rest of the paper to the case where κ is polynomial in n and, when applied to the algorithm from Section 4, gives our final running time of $\log^{O(1)}(n\kappa\epsilon^{-1})$.

2.3.2 Sparsification

Our primary new graph theoretic tool is a directed notion of spectral sparsifiers, along with efficient techniques for constructing them for an Eulerian graph and its square. As discussed in the introduction, there are seemingly intrinsic problems with many of the notions of directed sparsification that one would propose based on analogies to the undirected case. In particular, both the cut-based and spectral notions have seemingly fatal issues that preclude their use in directed graphs. For the cut-based notion, as shown in Section 1.1.2, good sparsifiers provably don't exist for some graphs.

If one instead seeks to generalize the undirected definition of spectral sparsifiers, which requires a sparsifier H of a graph G to obey $(1 - \epsilon)\vec{x}^\top \mathcal{L}_H \vec{x} \leq \vec{x}^\top \mathcal{L}_G \vec{x} \leq (1 + \epsilon)\vec{x}^\top \mathcal{L}_H \vec{x}$, the problems are perhaps even more severe. For instance, when G is directed \mathcal{L}_G is no longer symmetric, so it's not clear that it makes sense to use it as a quadratic form $\vec{x}^\top \mathcal{L}_G \vec{x}$, and doing so essentially symmetrizes

it and discards the directed structure, since $\vec{x}^\top \mathcal{L}_G \vec{x} = \vec{x}^\top \mathcal{L}_G^\top \vec{x} = \vec{x}^\top \left(\frac{\mathcal{L}_G + \mathcal{L}_G^\top}{2} \right) \vec{x}$. In addition, the resulting quadratic form is not typically PSD, i.e. there often exist \vec{x} for which $\vec{x}^\top \mathcal{L}_G \vec{x} < 0$, in which case G would not approximate itself under the definition given for $\epsilon > 0$.

One also has to deal with the fact that, unlike in the undirected case, the kernels of directed graph Laplacians are rather subtle objects: for a strongly-connected graph G , the kernel of $\mathcal{L}_G = \mathbf{D} - \mathbf{A}^\top$ is given by $\mathbf{D}^{-1}\phi$, where ϕ is the stationary distribution of the random walk on G . This carries various problematic consequences, including the fact that \mathcal{L} and \mathcal{L}^\top typically have different kernels, and even small changes in the graph can change whether $\mathcal{L}\vec{x} = 0$ for a given vector \vec{x} .

Our approach to this is based on the fact that many of these problems do not occur for Eulerian graphs. In particular, if \mathcal{L} is the Laplacian of an Eulerian directed graph G , $\mathbf{U}_\mathcal{L} = (\mathcal{L} + \mathcal{L}^\top)/2$ is the Laplacian of an undirected graph and thus positive semidefinite, and cuts in the corresponding undirected graph are the same as those in G . In addition, the kernel of \mathcal{L} is spanned by the all-ones vector and is the same as the kernel of \mathcal{L}^\top . In addition, the following was shown in [12], which says that the Laplacian of any strongly connected graph can be turned into an Eulerian Laplacian by applying a diagonal scaling:

Lemma 2.2 (Lemma 1 from [12], abridged). *Given a directed Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n \times n}$ whose associated graph is strongly connected, there exists a positive vector $\vec{x} \in \mathbb{R}_{>0}^n$ (unique up to scaling) such that $\mathcal{L} \cdot \mathbf{diag}(\vec{x})$ is an Eulerian Laplacian. Furthermore, $\ker(\mathcal{L}) = \text{span}(\vec{x})$, and $\ker(\mathcal{L}^\top) = \text{span}(\mathbf{1})$.*

Moreover, it was shown in [12] that one could find a high-precision approximation to this scaling efficiently given access to an Eulerian solver.

Intuitively, we define our notion of sparsification and approximation for Eulerian graphs, and we show that this notion induces a well-behaved definition for other strongly-connected graphs through the Eulerian scaling. As we do not want to neglect the directed structure, we will think of Laplacians as linear operators, not quadratic forms, and we study their sizes through various operator norms.

For Laplacians of Eulerian graphs, we use the fact that their symmetrizations are PSD, and our definition of approximation will demand that the difference between the two operators be small relative to the corresponding quadratic form. More precisely, we say that an Eulerian Laplacian \mathcal{L}_H ϵ -approximates another Eulerian Laplacian \mathcal{L}_G if $\|\mathbf{U}_{\mathcal{L}_G}^{+/2}(\mathcal{L}_H - \mathcal{L}_G)\mathbf{U}_{\mathcal{L}_G}^{+/2}\|_2 \leq \epsilon$. We note that this use of $\mathbf{U}_{\mathcal{L}_G}$ is closely related to the $\mathcal{L}_G^\top \mathbf{U}_{\mathcal{L}_G}^+ \mathcal{L}_G$ matrix that appeared in [12]. The difference, however, is that we are not trying to directly use this matrix as a symmetric stand-in for our Laplacian; we are working directly with the original (asymmetric) Laplacians and are just using it to help define a matrix norm.

To construct sparsifiers of Eulerian graphs with respect to this notion, we follow a similar approach to the one originally used by Spielman and Teng for spectral sparsification, but carefully tailored to the directed setting. The idea is to first partition our graph into well-connected components. Because the cuts in an Eulerian graph match those in its symmetrization, it makes sense to do this partitioning by simply partitioning the corresponding undirected graph into clusters with good expansion. We use existing decomposition techniques to argue that one can find such a partition with a significant fraction of the edges contained in the clusters. We then show a concentration result for asymmetric matrices that says that appropriately sub-sampling within these clusters preserves the relevant structure reasonably well while only keeping a small number of edges relative to the cluster size.

In the undirected case, one would just repeat this procedure until the graph is sparse. Where our procedure differs, however, is that we keep track of the directed structure along the way, and “patch” the subsampled object to keep it from diverging from what it should be. In particular,

the sampling procedure, when applied to an Eulerian graph will produce a non-Eulerian graph. However, we add additional edges to fix this after every sampling step and use our concentration bounds to show that the patches we add are sufficiently small to not decrease the quality of our approximation.

Carefully, analyzing this procedure allows us to produce a sparsifier in nearly linear time. However, in order to use our sparsification routine to produce a solver, we also need to sparsify the Laplacian of the square of a graph. To do this, we could just explicitly form the square and then sparsify it. However, we would like to perform this procedure in time that is nearly-linear in the number of edges of the original graph, whereas explicitly forming the square would cause the running time to grow with the number of edges of the square, which could be substantially larger. To prevent this, we instead show how to work with an implicit representation of the square that we can manipulate more efficiently, similar to [33].

2.3.3 Linear System Solving

In Section 4, we describe our algorithm for solving Eulerian Laplacian systems of equations. It begins with a similar template to the Peng-Spielman solver [33] described in Section 1.1.3, but with modifications to accommodate our non-symmetric setting. Given a linear system in an Eulerian Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$, we write $\mathcal{L} = \mathbf{D}^{1/2} (\mathbf{I} - \mathcal{A}) \mathbf{D}^{1/2}$, where $\mathcal{A} = \mathbf{D}^{-1/2} \mathbf{A}^\top \mathbf{D}^{-1/2}$. This reduces the problem to solving linear systems in $\mathbf{L} = \mathbf{I} - \mathcal{A}$ where we can show that $\|\mathcal{A}\|_2 = 1$. We then apply the expansion in Equation (1.2), but with some slight modifications:

- We find it convenient to build up the product expansion in Equation (1.2) recursively. We do so using the identity

$$(\mathbf{I} - \mathcal{A})^+ = (\mathbf{I} - \mathcal{A}^2)^+ (\mathbf{I} + \mathcal{A}), \quad (2.1)$$

which can be thought of as a matrix analogue of the rational function identity

$$\frac{1}{1-z} = \frac{1+z}{1-z^2}.$$

Applying this identity repeatedly gives

$$(\mathbf{I} - \mathcal{A})^+ = (\mathbf{I} - \mathcal{A}^2)^+ (\mathbf{I} + \mathcal{A}) = (\mathbf{I} - \mathcal{A}^4)^+ (\mathbf{I} + \mathcal{A}^2) (\mathbf{I} + \mathcal{A}) = (\mathbf{I} - \mathcal{A}^8)^+ (\mathbf{I} + \mathcal{A}^4) (\mathbf{I} + \mathcal{A}^2) (\mathbf{I} + \mathcal{A}) = \dots$$

After k applications of the identity, this yields the first k terms of the product expansion in (1.2) times $(\mathbf{I} - \mathcal{A}^{2^k})^+$, which converges to the identity as k gets large if $\|\mathcal{A}\|_2 < 1$. Some advantages of this compared to the infinite product expansion are that it gives an exact expression rather than an asymptotic result, which will be more convenient to work with when analyzing the growth of errors, and that the pseudoinverses in the expression gives a correct answer when $\|\mathcal{A}\| = 1$, which decreases the extent to which we need to explicitly handle the kernel of \mathcal{L} as a special case.

- If $z \neq 1$ is a complex number with $|z| = 1$, $1/(1-z)$ exists but the series $1/(1-z) = 1+z+z^2+\dots$ does not converge, and our matrix expansion will exhibit similar behavior. Graph theoretically, this case corresponds periodic behavior in the random walk, and we deal with it, as usual, by adding self-loops and working with a lazy random walk. Algebraically, we work with a convex combination with the identity,

$$\mathcal{A}^{(\alpha)} = \alpha \mathbf{I} + (1-\alpha) \mathcal{A},$$

and we note that $\mathbf{I} - \mathcal{A}^{(\alpha)} = (1-\alpha)(\mathbf{I} - \mathcal{A})$. We then replace the identity in Equation (2.1) with the modified identity

$$(\mathbf{I} - \mathcal{A})^+ = (1-\alpha) \left(\mathbf{I} - \mathcal{A}^{(\alpha)} \right)^+ = (1-\alpha) \left(\mathbf{I} - \mathcal{A}^{(\alpha)^2} \right)^+ \left(\mathbf{I} + \mathcal{A}^{(\alpha)} \right), \quad (2.2)$$

which leads to better convergence behavior. This step insures that each application of the identity causes a change that is more gradual than squaring. Moreover, our analysis takes advantage of the fact that taking a linear combination with the identity makes it easier to relate $\mathbf{I} - \mathcal{A}_{j+1}^{(\alpha)}$ to $\mathbf{I} - \mathcal{A}_j^{(\alpha)}$. While it may not be necessary to do at every step, it is used to simplify the current analysis. Note, that this algebraic simplification through ‘lazy’ random walks is also present in other works involving squaring [7, 17].

Similarly to the approach in [33], our strategy is to repeatedly apply (2.2), but to replace $(\mathcal{A}^{(\alpha)})^2$ with a sparsifier in each step to allow us to decrease the computational costs. More precisely, we show how to efficiently construct a sequence of matrices $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ and associated matrices $\mathcal{A}_i^{(\alpha)} = \alpha \mathbf{I} + (1 - \alpha) \mathcal{A}_i$ such that each matrix in the sequence has $\tilde{O}(n/\epsilon^2)$ nonzero entries, $\mathbf{I} - \mathcal{A}_0$ is an ϵ -approximation of $\mathbf{I} - \mathcal{A}$, and $\mathbf{I} - \mathcal{A}_i$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_{i-1}^{(\alpha)})^2$ for each $i \geq 1$ (note that we set \mathcal{A}_0 by sparsifying the original Laplacian). We call this a *square-sparsification chain*. In Section 4.2, we show how to compute all of the matrices in such a chain in time $\tilde{O}(\text{nnz}(\mathcal{L}) + n\epsilon^{-2}d)$, which we note is within logarithmic factors of the total number of nonzero entries.

The length of the chain is then dictated by the condition number $\kappa = \kappa(\mathbf{U}_{\mathbf{I}-\mathcal{A}})$, the condition number of the symmetric Laplacian associated with the input Eulerian Laplacian. Note that $\kappa = O(\text{poly}(nU))$ where $U \stackrel{\text{def}}{=} \max_{i,j} |\mathbf{A}_{ij}| / \min_{i,j: \mathbf{A}_{ij} \neq 0} |\mathbf{A}_{ij}|$ and may be smaller. If we set $d = \Omega(\log \kappa)$, we show that $\mathbf{I} - \mathcal{A}_d^{(\alpha)}$ well-conditioned. We can thus stop our recursion at this point and (approximately) apply $(\mathbf{I} - \mathcal{A}_d^{(\alpha)})^+$ using a small number of iterations of a standard iterative method, Richardson iteration.

Expanding the recurrence in (2.2) gives

$$(\mathbf{I} - \mathcal{A}_i)^+ \approx (1 - \alpha)^{j-i} \left(\mathbf{I} - \mathcal{A}_j^{(\alpha)} \right)^+ \left(\mathbf{I} + \mathcal{A}_{j-1}^{(\alpha)} \right) \left(\mathbf{I} + \mathcal{A}_{j-2}^{(\alpha)} \right) \cdots \left(\mathbf{I} + \mathcal{A}_i^{(\alpha)} \right). \quad (2.3)$$

If we have already computed the matrices in the chain, we can apply the right-hand side to a vector \vec{b} by performing $(j - i)$ matrix-vector multiplications and solving a linear system in $\mathbf{I} - \mathcal{A}_j^{(\alpha)}$. It is useful to think of this as an approximate reduction from applying $(\mathbf{I} - \mathcal{A}_i)^+$ to applying $(\mathbf{I} - \mathcal{A}_j^{(\alpha)})^+$. The matrices in (2.3) have at most $\tilde{O}(n/\epsilon^2)$ nonzero entries, so the total time for the matrix vector multiplications is then at most $\tilde{O}((j - i)n\epsilon^{-2})$.

Because of the errors introduced by the sparsification steps, the right-hand side of (2.3) is only an approximation of $(\mathbf{I} - \mathcal{A}_i)^+$, so applying it directly to \vec{b} only yields a (typically somewhat crude) approximation to solution to $(\mathbf{I} - \mathcal{A}_i)\vec{x} = \vec{b}$. To obtain a better solution, we instead use it as a preconditioner inside an iterative method for the linear system. This allows us to obtain an arbitrarily good solution to the system, and the quality of the approximation in (2.3) then determines the number of iterations required.

This suggests that we quantify the error in our approximations using a notion that directly bounds the convergence rate of such a preconditioned iterative method. We do so with the notion of an ϵ -approximate pseudoinverse (defined with respect to some PSD matrix \mathbf{U}), which we introduce in Section 4.1. Roughly speaking, solving a linear system with an appropriate iterative method using such a matrix as a preconditioner will guarantee the \mathbf{U} -norm of the error to decrease by a factor of ϵ in each iteration. We note that this is only useful for $\epsilon < 1$. For technical reasons, we measure the quality of approximate pseudoinverses with respect to different \mathbf{U} matrices at different stages of the algorithm and translate between them. For simplicity, we just refer to an “ ϵ -approximate pseudoinverse” in this overview, but in our algorithm we set the value of ϵ in our sparsification routines and apply iterative methods, again Richardson iterations, to appropriately pay for the costs of translating between norms.

To analyze the errors introduced by sparsification, we therefore need to:

1. Relate our notion of graph approximation to approximate pseudoinverses, and
2. Bound the rate at which the quality of the approximate pseudoinverse we produce decreases as we increase the number of terms in (2.3). We use (2.3) recursively, so it is also useful to bound how this is affected if we use an approximate pseudoinverse instead of the exact operator $(\mathbf{I} - \mathcal{A}_j^{(\alpha)})^+$.

For the former, we show in Theorem 4.11 that our notion of an ϵ -sparsifier leads to an $O(\epsilon)$ -approximate pseudoinverse. For the latter, we show in Lemma 4.13 that using a square-sparsifier chain of length d with some given ϵ , and using an ϵ' -approximate pseudoinverse of $(\mathbf{I} - \mathcal{A}_j^{(\alpha)})$ in place of $(\mathbf{I} - \mathcal{A}_j^{(\alpha)})^+$, produces an $(\epsilon + \epsilon') \cdot 2^{O(d)}$ -approximate pseudoinverse for $\mathbf{I} - \mathcal{A}_j^{(\alpha)}$.

The exponential dependence of the error on length of the chain is a key difference between our analysis and the undirected case, and it is what prevents us from having a simpler and more efficient algorithm. If the dependence on the chain length were polynomial, applying (2.3) with $i = 0$ and $j = d$ would provide an $\epsilon \cdot \text{polylog}(n)$ -approximate pseudoinverse. We could thus set $\epsilon = 1/\text{polylog}(\kappa)$ in our sparsifier chain and get an $O(1)$ -approximate pseudoinverse in $\tilde{O}(n)$ time. An iterative method could then call this $\log(1/\delta)$ times to obtain a solution with error δ . However, because of the exponential dependence on the chain length, we would only get an $\epsilon \cdot \text{poly}(\kappa)$ -approximate pseudoinverse. We would thus need to set $\epsilon = 1/\text{poly}(\kappa)$ to get a value less than 1, which would lead to “sparsifiers” with $\tilde{\Omega}(n \cdot \text{poly}(\kappa))$ edges. In the typical case where $\kappa = \text{poly}(n)$, simply writing these down would exceed the desired almost-linear time bound.

To prevent this, we do not wait until the end to apply an iterative method to reduce the error. Instead, we break our sparsification and squaring steps into $\lceil d/\Delta \rceil$ blocks of size $\Delta \ll d$, each of which we will wrap in several steps of Richardson iteration (which we review in Section 4.1), in order to keep the error under control.

Our algorithm first computes (once, not recursively) a square-sparsifier chain of length $d = O(\log \kappa)$ in which the sparsifiers are ϵ_{spar} -approximations. It then recursively combines two types of steps that are suggested by the discussion above:

- **High error $(\mathbf{I} - \mathcal{A}_i^{(\alpha)})^+$ from low error $(\mathbf{I} - \mathcal{A}_{i+\Delta}^{(\alpha)})^+$:** Given a routine to apply an ϵ_{lo} -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_{i+\Delta}^{(\alpha)}$ in time $\mathcal{T}_{i+\Delta, \epsilon_{\text{lo}}}$, we can use the expansion in (2.3) to apply an ϵ_{hi} -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i^{(\alpha)}$ in time $\mathcal{T}_{i, \epsilon_{\text{hi}}} = \mathcal{T}_{i+\Delta, \epsilon_{\text{lo}}} + \tilde{O}(\Delta n \epsilon_{\text{spar}}^{-2})$, where $\epsilon_{\text{hi}} = (\epsilon_{\text{spar}} + \epsilon_{\text{lo}})2^{O(\Delta)}$.
- **Low error $(\mathbf{I} - \mathcal{A}_i^{(\alpha)})^+$ from high error $(\mathbf{I} - \mathcal{A}_i^{(\alpha)})^+$:** By running Richardson iteration for $O(\log \epsilon_{\text{lo}}/\log \epsilon_{\text{hi}})$ steps, we can turn an ϵ_{hi} -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i^{(\alpha)}$ into a ϵ_{lo} -approximate pseudoinverse. This applies the former once in each iteration, so it takes time

$$\mathcal{T}_{i, \epsilon_{\text{lo}}} = O\left(\frac{\log \epsilon_{\text{lo}}}{\log \epsilon_{\text{hi}}}\right) \mathcal{T}_{i, \epsilon_{\text{hi}}} = O\left(\frac{\log \epsilon_{\text{lo}}}{\log \epsilon_{\text{hi}}}\right) \left(\mathcal{T}_{i+\Delta, \epsilon_{\text{lo}}} + \tilde{O}(\Delta n \epsilon_{\text{spar}}^{-2})\right). \quad (2.4)$$

If we set ϵ_{hi} to be a constant (say, $1/10$), we get $\epsilon_{\text{spar}} + \epsilon_{\text{lo}} = 2^{-\Omega(\Delta)}$, so we set $\epsilon_{\text{spar}} = \epsilon_{\text{lo}} = 2^{-\Theta(\Delta)}$, and (2.4) simplifies to

$$\mathcal{T}_{i, \epsilon_{\text{lo}}} = O(\Delta) \left(\mathcal{T}_{i+\Delta, \epsilon_{\text{lo}}} + \tilde{O}(\Delta n 2^{\Theta(\Delta)})\right) = O(\Delta) \mathcal{T}_{i+\Delta, \epsilon_{\text{lo}}} + \tilde{O}(n 2^{\Theta(\Delta)}).$$

For the base case of our recurrence, $\mathbf{I} - \mathcal{A}_d^{(\alpha)}$ is well-conditioned, so we can approximately apply its pseudoinverse using a standard iterative method in time $\mathcal{T}_{d, \epsilon_{\text{lo}}} = \tilde{O}(\text{nnz}(\mathcal{A}_d^{(\alpha)}) \log \epsilon_{\text{lo}}^{-1}) = \tilde{O}(n \epsilon_{\text{spar}}^{-2} \log \epsilon_{\text{lo}}^{-1}) = \tilde{O}(n 2^{\Theta(\Delta)})$. This can be folded into the additive $\tilde{O}(n 2^{\Theta(\Delta)})$ term in the recurrence, so it does not significantly affect the time bound.

To estimate the solution to the recurrence, we note that depth of the recursion is $\lceil d/\Delta \rceil$, and at each stage we multiply by $O(\Delta)$. We can think of this as producing a recursion tree with $O(\Delta)^{\lceil d/\Delta \rceil}$ nodes, and we add $\tilde{O}(n 2^{\Theta(\Delta)})$ at each, so we get that

$$\mathcal{T}_{0, \epsilon_{\text{lo}}} = O(\Delta)^{\lceil d/\Delta \rceil} \tilde{O}(n 2^{O(\Delta)}) = n O(\Delta)^{O(d/\Delta)} 2^{O(\Delta)} = n 2^{O(\Delta + \frac{d \log \Delta}{\Delta})}.$$

Setting $\Delta = \sqrt{d \log d} = \sqrt{\log \kappa \log \log \kappa}$ approximately balances the two terms in the exponent. Plugging this in and adding the $\tilde{O}(m)$ for the overhead from the non-recursive parts of the algorithm gives our running time bound of $\tilde{O}(m) + n 2^{O(\sqrt{\log \kappa \log \log \kappa})}$.

3 Sparsification of Directed Laplacians

In this section, we define what it means for one strongly connected directed graph to approximate another, and we use this to define directed sparsifiers. We show that such sparsifiers exist for any directed graph and give efficient algorithms to construct them. For our almost linear time directed Laplacian system solver (Section 4), we only need to be able to sparsify Eulerian graphs. However, we have included the more general case of non-Eulerian graphs because we believe it is of independent interest and may be useful in other settings.

As discussed in Section 2.3.2, we define our notion of graph approximation by first giving a notion of approximation for matrices and then applying it to (possibly rescaled versions of) directed graph Laplacians. This notion will be qualitatively better-behaved for Laplacians of Eulerian graphs than for general directed Laplacians. As such, our definition will make use of the existence of a scaling that makes any strongly connected graph Eulerian.

Our notion of approximation for asymmetric matrices is defined as follows:

Definition 3.1 (Asymmetric Matrix Approximation). A (possibly asymmetric) matrix $\tilde{\mathbf{A}}$ is said to be an ϵ -approximation of \mathbf{A} if:

1. $\mathbf{U}_{\mathbf{A}}$ is a symmetric PSD matrix, with $\ker(\mathbf{U}_{\mathbf{A}}) \subseteq \ker(\tilde{\mathbf{A}} - \mathbf{A}) \cap \ker((\tilde{\mathbf{A}} - \mathbf{A})^\top)$, and
2. $\|\mathbf{U}_{\mathbf{A}}^{+1/2}(\tilde{\mathbf{A}} - \mathbf{A})\mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq \epsilon$.

When these properties hold for some constant $\epsilon \in (0, 1)$ we simply say $\tilde{\mathbf{A}}$ approximates \mathbf{A} .

In Section 3.1 we provide an equivalent definition and several facts which justify this choice of matrix approximation. In particular, we prove the following facts regarding Definition 3.1

- it generalizes spectral approximation (ie., small relative condition number) of symmetric matrices, and behaves predictably under perturbations;
- it implies the symmetrizations $\mathbf{U}_{\mathbf{A}}$ and $\mathbf{U}_{\tilde{\mathbf{A}}}$ of \mathbf{A} and $\tilde{\mathbf{A}}$, spectrally approximate each other;
- its behavior under composition is natural.

Furthermore, in Appendix E we show that our notion of approximation also yields approximations of the symmetric systems solved in [12], known as harmonic symmetrizations.

We extend this notion of approximation from asymmetric matrices to directed graphs as follows:

Definition 3.2 (Directed Graph Approximation). Let $\mathcal{L}, \tilde{\mathcal{L}} \in \mathbb{R}^{n \times n}$ be the Laplacians of strongly-connected directed graphs G and \tilde{G} respectively, and let $\mathbf{X} = \text{diag}(\vec{x})$ and $\tilde{\mathbf{X}} = \text{diag}(\vec{\tilde{x}})$ be the diagonal matrices for which $\mathcal{L}\mathbf{X}$ and $\tilde{\mathcal{L}}\tilde{\mathbf{X}}$ are Eulerian Laplacians that are guaranteed to exist by Lemma 2.2, normalized to have $\text{Tr}(\mathbf{X}) = \text{Tr}(\tilde{\mathbf{X}}) = n$.

We say that \tilde{G} is an ϵ -approximation of G if:

1. $(1 - \epsilon)\mathbf{X} \leq \tilde{\mathbf{X}} \leq (1 + \epsilon)\mathbf{X}$, and
2. $\tilde{\mathcal{L}}\tilde{\mathbf{X}}$ is an ϵ -approximation of $\mathcal{L}\mathbf{X}$.

If $\tilde{\mathbf{X}} = \mathbf{X}$, we say that \tilde{G} is a *strict* ϵ -approximation of G .

In words, we say that a graph approximates another graph if their Eulerian scalings are within small multiplicative factors of one another and the resulting Eulerian graphs obey our definition of asymmetric matrix approximation. We call the approximation “strict” if their Eulerian scalings are not just within small multiplicative factors of one another but are actually identical.

Our main use of this notion is to define *sparsifiers*, which are approximations that have a small number of nonzero entries.

Definition 3.3. (Graph Sparsifier) Let $\mathcal{L}, \tilde{\mathcal{L}} \in \mathbb{R}^{n \times n}$ be the Laplacians of strongly-connected directed graphs G and \tilde{G} , respectively. We say that \tilde{G} is a (strict) ϵ -sparsifier of G if:

1. \tilde{G} is a (strict) ϵ -approximation of G , and
2. $\text{nnz}(\tilde{\mathcal{L}}) \leq \tilde{O}(n\epsilon^{-2})$, where n is the number of vertices in G .

We note that, if we show that strict sparsifiers exist for Eulerian graphs, this will imply that they exist for general strongly connected graphs as well. One can simply apply the graph’s Eulerian scaling, find a sparsifier for the resulting Eulerian graph, and then “unscale” the graph by applying the inverse of the Eulerian scaling. The results of this paper allow us to compute an Eulerian scaling for any graph in almost-linear time. Thus, the almost linear time sparsification procedure we will give for Eulerian graphs will imply an almost linear time sparsification procedure for all graphs.⁴ As such, we will focus on the Eulerian case. In this case, graph approximation will be the same as matrix sparsification, and it will often be convenient to refer directly to the Laplacian, instead of to the graph. Moreover, we will seek to exactly preserve the fact that the graph is Eulerian, so we will exclusively consider strict approximations. We thus define:⁵

Definition 3.4 (Eulerian Sparsifier). $\tilde{\mathcal{L}} \in \mathbb{R}^{n \times n}$ is an ϵ -sparsifier of an Eulerian Laplacian \mathcal{L} if

1. $\tilde{\mathcal{L}}$ is a strict ϵ -approximation of \mathcal{L}
2. $\text{nnz}(\tilde{\mathcal{L}}) \leq \tilde{O}(n\epsilon^{-2})$.

In the remainder of the section, we show how to produce such sparsifiers of Eulerian Laplacians. I.E., given an Eulerian Laplacian, we will obtain an Eulerian Laplacian that approximates the original and has a small number of nonzero entries.

⁴One has to exercise some care with the numerics here, since we will only be able to compute a finite precision estimate of the Eulerian scaling of a graph. So, if we apply this scaling and then want to sparsify, the graph we want to sparsify will not be perfectly Eulerian. However, it is straightforward to show that—as long as the approximate Eulerian scaling being used is fairly precise—one can “patch” the rescaled graph to become Eulerian while only incurring a very small loss in the approximation quality.

⁵We could ask for sparsifiers under other notions of approximation, such as the weaker conditions required to obtain a preconditioner (see Section 4.1); as our algorithms always give this notion we use this terminology.

Moreover, we show how to construct these sparsifiers in nearly linear time. Specifically, we give an algorithm that produces an ϵ -sparsifier of an Eulerian Laplacian \mathcal{L} with high probability in $\tilde{O}(\text{nnz}(\mathcal{L})/\epsilon^2)$ time. Even without any additional work, this result alone immediately implies some improvement in the runtime for solving arbitrary directed Laplacian systems. Specifically, one can write down the harmonic symmetrization of the original matrix, the harmonic symmetrization of the sparsifier, and solve the original harmonic symmetrization preconditioned by the harmonic symmetrization of the sparsifier. We prove in Appendix E that these Harmonic symmetrizations have small relative condition number, so the runtime of this solver will be dominated by the time it takes to solve systems in the sparsified matrix plus the time to apply the unsparsified matrix to a vector. Using the solver in [12], this comes out to an $\tilde{O}(m + n^{7/4})$ time algorithm for solving directed Laplacian systems.

In order to construct a better, almost linear time solver, we'll also need to be able to sparsify a normalized version of the Laplacian of the square of graph. Specifically, we also show how to sparsify any matrix of the form $\mathbf{D} - \mathbf{A}^\top \mathbf{D}^{-1} \mathbf{A}^\top$, where we are given the adjacency matrix \mathbf{A} of some Eulerian graph G and the degrees \mathbf{D} of G . Note that if G is regular and has all (weighted) degrees equal to one, this formula simplifies to $I - (\mathbf{A}^\top)^2$. Thus, it corresponds to the Laplacian of the square of the graph in this special case, and to a normalized version of it in general. Our algorithm for sparsifying matrices of this form takes outputs a strict ϵ -sparsifier in $\tilde{O}(\text{nnz}(\mathcal{L})\epsilon^{-2})$ time. Combining these results, and using our properties regarding asymmetric approximations, we show that we can use this ϵ -approximation to efficiently obtain an ϵ -sparsifier of $\mathbf{D} - \mathbf{A}^\top \mathbf{D}^{-1} \mathbf{A}^\top$. Applying a closely related routine recursively we obtain a faster, almost linear time algorithm for solving Eulerian Laplacian systems in Section 4.

The remainder of this section is structured as follows. First, in Section 3.1, we provide various facts regarding our notion of asymmetric approximation. Then, in Section 3.2, we provide one of our main technical tools: an algorithm for crudely sparsifying an arbitrary (not necessarily Eulerian) directed Laplacian by randomly sampling its adjacency matrix. On its own, this algorithm achieves relatively weak guarantees. In Section 3.3 we then combine this tool with known decomposition results for undirected graphs to sparsify any Eulerian Laplacian. In Section 3.4, we build on this to sparsify the normalized square of any Eulerian Laplacian.

3.1 Approximation Facts

Here we provide various basic facts regarding the notion of approximation for asymmetric matrices given in Definition 3.4. These facts both motivate and justify our choice of definition and are used extensively throughout this section.

First we provide the following lemma, giving alternative definitions of approximation in terms of a quantity reminiscent of Rayleigh quotients.

Lemma 3.5 (Equivalent Approximation Definitions). *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be such that $\mathbf{U}_\mathbf{A}$ PSD. A matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ is an ϵ -approximation of \mathbf{A} if and only if*

$$\max_{\vec{x}, \vec{y} \neq 0} \frac{\vec{x}^\top (\tilde{\mathbf{A}} - \mathbf{A}) \vec{y}}{\sqrt{\vec{x}^\top \mathbf{U}_\mathbf{A} \vec{x} \cdot \vec{y}^\top \mathbf{U}_\mathbf{A} \vec{y}}} \leq \epsilon \quad \text{or equivalently} \quad \max_{\vec{x}, \vec{y} \neq 0} \frac{\vec{x}^\top (\tilde{\mathbf{A}} - \mathbf{A}) \vec{y}}{\vec{x}^\top \mathbf{U}_\mathbf{A} \vec{x} + \vec{y}^\top \mathbf{U}_\mathbf{A} \vec{y}} \leq \frac{\epsilon}{2},$$

under the convention that $0/0 = 0$.

Proof. This lemma follows from a more general result, Lemma B.2, which we prove in Appendix C, and by noting that if $\ker(\mathbf{U}_\mathbf{A})$ is not a subset of both $\tilde{\mathbf{A}} - \mathbf{A}$ and its transpose then the maximization problems are infinite in value. \square

This lemma will allow us to show that our notion of ϵ -approximation does coincide with the standard notion in the case of symmetric matrices, and is therefore a stronger notion. More generally, we prove that ϵ -approximation of asymmetric matrices implies that their symmetrizations are ϵ -approximations in the traditional spectral (small relative condition number) sense.

Lemma 3.6. *Suppose $\tilde{\mathbf{A}}$ is an ϵ -approximation of \mathbf{A} . Then*

$$(1 - \epsilon)\mathbf{U}_{\mathbf{A}} \preceq \mathbf{U}_{\tilde{\mathbf{A}}} \preceq (1 + \epsilon)\mathbf{U}_{\mathbf{A}} .$$

Proof. Suppose $\tilde{\mathbf{A}}$ is an ϵ -approximation of \mathbf{A} , and let $\vec{x} \in \mathbb{R}^n$ with $\vec{x} \neq 0$ be arbitrary. Applying Lemma 3.5 twice with $\vec{y} = \pm\vec{x}$ we have

$$\left| \frac{\vec{x}^\top (\tilde{\mathbf{A}} - \mathbf{A}) \vec{x}}{\vec{x}^\top \mathbf{U}_{\mathbf{A}} \vec{x}} \right| \leq \|\mathbf{U}_{\mathbf{A}}^{+1/2} (\tilde{\mathbf{A}} - \mathbf{A}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq \epsilon .$$

The desired result follows from the fact that $\vec{z}^\top \mathbf{A} \vec{z} = \vec{z}^\top \mathbf{U}_{\mathbf{A}} \vec{z}$ and $\vec{z}^\top \tilde{\mathbf{A}} \vec{z} = \vec{z}^\top \mathbf{U}_{\tilde{\mathbf{A}}} \vec{z}$ for all z . \square

Next we use Lemma 3.5 to show that just as in the symmetric case, asymmetric approximation is preserved when taking symmetric products.

Lemma 3.7. *If $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ is an ϵ -approximation of $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{M} \in \mathbb{R}^{n \times n}$ satisfies $\ker(\mathbf{M}^\top) \subseteq \ker(\mathbf{U}_{\mathbf{A}})$ then $\mathbf{M}^\top \tilde{\mathbf{A}} \mathbf{M}$ is an ϵ -approximation of $\mathbf{M}^\top \mathbf{A} \mathbf{M}$.*

Proof. Define $\mathbf{B} \stackrel{\text{def}}{=} \mathbf{M}^\top \mathbf{A} \mathbf{M}$ and $\tilde{\mathbf{B}} \stackrel{\text{def}}{=} \mathbf{M}^\top \tilde{\mathbf{A}} \mathbf{M}$. We first wish to show $\ker(\mathbf{U}_{\mathbf{B}}) \subseteq \ker(\tilde{\mathbf{B}} - \mathbf{B}) \cap \ker((\tilde{\mathbf{B}} - \mathbf{B})^\top)$. Suppose we have any $x \in \ker(\mathbf{U}_{\mathbf{B}})$. Then,

$$\mathbf{M}x \in \ker(\mathbf{M}^\top \mathbf{U}_{\mathbf{A}}) = \ker(\mathbf{U}_{\mathbf{A}}) \subseteq \ker(\tilde{\mathbf{A}} - \mathbf{A}) \cap \ker((\tilde{\mathbf{A}} - \mathbf{A})^\top) \subseteq \ker(\mathbf{M}^\top (\tilde{\mathbf{A}} - \mathbf{A})) \cap \ker(\mathbf{M}^\top (\tilde{\mathbf{A}} - \mathbf{A})^\top)$$

which implies the portion of the definition of approximation concerning kernels.

Then we have, using the convention that $0/0 = 0$ and by applying Lemma B.2 and Lemma 3.5, that

$$\begin{aligned} \|\mathbf{U}_{\mathbf{B}}^{+1/2} (\tilde{\mathbf{B}} - \mathbf{B}) \mathbf{U}_{\mathbf{B}}^{+1/2}\|_2 &= \max_{\vec{x}, \vec{y} \neq 0} \frac{\vec{x}^\top (\tilde{\mathbf{B}} - \mathbf{B}) \vec{y}}{\sqrt{\vec{x}^\top \mathbf{U}_{\mathbf{B}} \vec{x} \cdot \vec{y}^\top \mathbf{U}_{\mathbf{B}} \vec{y}}} v = \max_{\vec{x}, \vec{y} \neq 0} \frac{\vec{x}^\top \mathbf{M}^\top (\tilde{\mathbf{A}} - \mathbf{A}) \mathbf{M} \vec{y}}{\sqrt{\vec{x}^\top \mathbf{M}^\top \mathbf{U}_{\mathbf{A}} \mathbf{M} \vec{x} \cdot \vec{y}^\top \mathbf{M}^\top \mathbf{U}_{\mathbf{A}} \mathbf{M} \vec{y}}} \\ &\leq \max_{\vec{x}, \vec{y} \neq 0} \frac{\vec{x}^\top (\tilde{\mathbf{A}} - \mathbf{A}) \vec{y}}{\sqrt{\vec{x}^\top \mathbf{U}_{\mathbf{A}} \vec{x} \cdot \vec{y}^\top \mathbf{U}_{\mathbf{A}} \vec{y}}} = \|\mathbf{U}_{\mathbf{A}}^{+1/2} (\tilde{\mathbf{A}} - \mathbf{A}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq \epsilon . \end{aligned}$$

\square

Finally, we provide a transitivity result for strong-approximation.

Lemma 3.8 (Approximation Transitivity). *If \mathbf{C} is an ϵ -approximation of \mathbf{B} and \mathbf{B} is an ϵ -approximation of \mathbf{A} then \mathbf{C} is an $\epsilon(2 + \epsilon)$ -approximation of \mathbf{A} .*

Proof. Note that by triangle inequality

$$\|\mathbf{U}_{\mathbf{A}}^{+1/2} (\mathbf{C} - \mathbf{A}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq \|\mathbf{U}_{\mathbf{A}}^{+1/2} (\mathbf{C} - \mathbf{B}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 + \|\mathbf{U}_{\mathbf{A}}^{+1/2} (\mathbf{B} - \mathbf{A}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 .$$

Now, $\mathbf{U}_{\mathbf{B}} \preceq (1 + \epsilon)\mathbf{U}_{\mathbf{A}}$ by Lemma 3.6 and therefore $\mathbf{U}_{\mathbf{A}}^+ \preceq (1 + \epsilon)\mathbf{U}_{\mathbf{B}}^+$. Applying Lemma B.3 yields

$$\|\mathbf{U}_{\mathbf{A}}^{+1/2} (\mathbf{C} - \mathbf{B}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq (1 + \epsilon) \cdot \|\mathbf{U}_{\mathbf{B}}^{+1/2} (\mathbf{C} - \mathbf{B}) \mathbf{U}_{\mathbf{B}}^{+1/2}\|_2 .$$

The result follows as $\|\mathbf{U}_{\mathbf{A}}^{+1/2} (\mathbf{B} - \mathbf{A}) \mathbf{U}_{\mathbf{A}}^{+1/2}\|_2 \leq \epsilon$ and $\|\mathbf{U}_{\mathbf{B}}^{+1/2} (\mathbf{C} - \mathbf{B}) \mathbf{U}_{\mathbf{B}}^{+1/2}\|_2 \leq \epsilon$ by assumption. \square

3.2 Sampling a Directed Laplacian

Here we show how to compute a crude, sparse approximation to an arbitrary directed Laplacian by randomly sampling its entries. We provide both a general bound on the effect of such sampling for a directed Laplacian as well as a more specific result in the case where the directed Laplacian can be related to the symmetric Laplacian of an expander. The latter result (Lemma 3.13) and the terminology relevant to it, is all we use from this subsection in order to obtain and analyze our sparsification algorithms.

The main tool for our analysis is Theorem 3.9, a general bound on concentration when sampling the entries of an asymmetric matrix. Its proof follows directly from standard matrix concentration inequalities, so we defer its proof to Appendix A.

Theorem 3.9. *Let $\mathbf{A} \in \mathbb{R}_{\geq 0}^{d_1 \times d_2}$ be a matrix where no row or column is all zeros. Let $\epsilon, p \in (0, 1)$, $s = d_1 + d_2$, $\vec{r} = \mathbf{A}\mathbb{1}$, $\vec{c} = \mathbf{A}^\top \mathbb{1}$, and \mathcal{D} be a distribution over $\mathbb{R}^{d_1 \times d_2}$ such that $\mathbf{X} \sim \mathcal{D}$ takes value*

$$\mathbf{X} = \begin{pmatrix} \mathbf{A}_{ij} \\ p_{ij} \end{pmatrix} \vec{r}_i \vec{c}_j^\top \text{ with probability } p_{ij} = \frac{\mathbf{A}_{ij}}{s} \left[\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right] \text{ for all } \mathbf{A}_{ij} \neq 0 .$$

If $\mathbf{A}_1, \dots, \mathbf{A}_k$ are sampled independently from \mathcal{D} for $k \geq 128 \cdot \frac{s}{\epsilon^2} \log \frac{s}{p}$, $\mathbf{R} = \mathbf{diag}(\vec{r})$, and $\mathbf{C} = \mathbf{diag}(\vec{c})$ then the average $\tilde{\mathbf{A}} \stackrel{\text{def}}{=} \frac{1}{k} \sum_{i \in [k]} \mathbf{A}_i$, satisfies

$$\begin{aligned} \Pr \left[\|\mathbf{R}^{-1/2} (\tilde{\mathbf{A}} - \mathbf{A}) \mathbf{C}^{-1/2}\|_2 \geq \epsilon \right] &\leq p, \\ \Pr \left[\|\mathbf{R}^{-1} (\tilde{\mathbf{A}} - \mathbf{A}) \mathbb{1}\|_\infty \geq \epsilon \right] &\leq p, \text{ and} \\ \Pr \left[\|\mathbf{C}^{-1} (\tilde{\mathbf{A}} - \mathbf{A})^\top \mathbb{1}\|_\infty \geq \epsilon \right] &\leq p. \end{aligned}$$

Theorem 3.9 shows that by sampling the entries of a rectangular matrix we can compute a new matrix such that spectral norm of the differences is bounded and the row sums and column sums are approximately preserved. In the next lemma we should how we can use this procedure to obtain a matrix with the same bound on the difference in spectral norm, but the row and column norms preserved *exactly*. In short, we show how to add a matrix to the result of Theorem 3.9 preserving its properties while fixing the row norms, we call this procedure *patching*.

Lemma 3.10 (Sparsifying Non-negative Matrices). *Let $\mathbf{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ be a matrix with non-negative entries and having no all zeros row or column. Let $\epsilon, p \in (0, 1)$. In $O(\text{nnz}(\mathbf{A}) + n\epsilon^{-2} \log(n/p))$ time we can compute a matrix $\tilde{\mathbf{A}} \in \mathbb{R}_{\geq 0}^{n \times n}$ with non-negative entries such that for $\mathbf{R} \stackrel{\text{def}}{=} \mathbf{diag}(\mathbf{A}\mathbb{1})$ and $\mathbf{C} \stackrel{\text{def}}{=} \mathbf{diag}(\mathbf{A}^\top \mathbb{1})$ we have*

1. $\text{nnz}(\tilde{\mathbf{A}}) = O(n\epsilon^{-2} \log(n/p))$,⁶
2. the row and column sums of \mathbf{A} and $\tilde{\mathbf{A}}$ are the same, i.e. $\mathbf{A}\mathbb{1} = \tilde{\mathbf{A}}\mathbb{1}$ and $\mathbf{A}^\top \mathbb{1} = \tilde{\mathbf{A}}^\top \mathbb{1}$.
3. for $i \in [n]$, if $\mathbf{A}_{ii} = 0$ then $\tilde{\mathbf{A}}_{ii} = 0$, and
4. with probability at least $1 - p$, $\|\mathbf{R}^{-1/2} (\mathbf{A} - \tilde{\mathbf{A}}) \mathbf{C}^{-1/2}\|_2 \leq \epsilon$.

⁶Note that it is possible to remove the dependence in p from the sparsifier simply by increasing the running time by a constant factor and making it an expected running time. This can be achieved by using the power method to approximately compute the value of $\|\mathbf{R}^{-1/2} (\mathbf{A} - \tilde{\mathbf{A}}) \mathbf{C}^{-1/2}\|_2$ and resampling when this is large.

Proof. We prove this using Theorem 3.9. Let $\epsilon' = \epsilon/4$. By sampling as in Theorem 3.9 we can compute $\widehat{\mathbf{A}} \in \mathbb{R}^{n \times n}$ such that $\text{nnz}(\widehat{\mathbf{A}}) = O(n\epsilon^{-2} \log(n/p))$, $\|\mathbf{R}^{-1/2}(\mathbf{A} - \widehat{\mathbf{A}})\mathbf{C}^{-1/2}\|_2 \leq \epsilon'$,

$$\|\mathbf{R}^{-1}(\widehat{\mathbf{A}} - \mathbf{A})\mathbb{1}\|_\infty \leq \epsilon' \text{ and } \|\mathbf{C}^{-1}(\widehat{\mathbf{A}} - \mathbf{A})^\top \mathbb{1}\|_\infty \leq \epsilon'.$$

Note that the latter two conditions imply that entrywise the row sums and column sums of \mathbf{A} are approximately the same as $\widehat{\mathbf{A}}$. Formally, it implies that entrywise the following inequalities hold

$$(1 - \epsilon')\mathbf{A}\mathbb{1} \leq \widehat{\mathbf{A}}\mathbb{1} \leq (1 + \epsilon')\mathbf{A}\mathbb{1} \text{ and } (1 - \epsilon')\mathbf{A}^\top \mathbb{1} \leq \widehat{\mathbf{A}}^\top \mathbb{1} \leq (1 + \epsilon')\mathbf{A}^\top \mathbb{1}. \quad (3.1)$$

Therefore $(1 + \epsilon')^{-1} \cdot \widehat{\mathbf{A}}$ has row and column sums that are less than or equal to those of \mathbf{A} .

Next, we compute a matrix to make the row and column sums the same as in \mathbf{A} . Formally, we let $\mathbf{E} \in \mathbb{R}_{\geq 0}^{n \times n}$ be a matrix with $\text{nnz}(\mathbf{E}) = O(n)$ such that

$$((1 + \epsilon')^{-1}\widehat{\mathbf{A}} + \mathbf{E})\mathbb{1} = \mathbf{A}\mathbb{1}, \text{ and } ((1 + \epsilon')^{-1}\widehat{\mathbf{A}} + \mathbf{E})^\top = \mathbf{A}^\top \mathbb{1}$$

and the ℓ_1 norm of the entries of \mathbf{E} is at most $n\epsilon$.

We can compute such a matrix \mathbf{E} in $O(\text{nnz}(\widehat{\mathbf{A}}))$ time by greedily adding in values to $\widehat{\mathbf{A}}$ to make one of the row or column sums as large as that of \mathbf{A} , while maintaining the invariant that no row or column sum is larger than it is in \mathbf{A} . Note that if $\mathbf{A}_{ii} = 0$ for all $i \in [n]$, then $\widehat{\mathbf{A}}_{ii} = 0$ for all $i \in [n]$, and it can be ensured that $\mathbf{E}_{ii} = 0$ for all $i \in [n]$.

Finally, we output $\widetilde{\mathbf{A}} = (1 + \epsilon')^{-1}\widehat{\mathbf{A}} + \mathbf{E}$. By construction $\text{nnz}(\widetilde{\mathbf{A}}) = O(n\epsilon^{-2} \log(n/p))$ and \mathbf{A} and $\widetilde{\mathbf{A}}$ have the same row and column sums, i.e. $\widetilde{\mathbf{A}}\mathbb{1} = \mathbf{A}\mathbb{1}$, $\widetilde{\mathbf{A}}^\top \mathbb{1} = \mathbf{A}^\top \mathbb{1}$. All that remains, is to show that the last property holds, i.e., that $\widetilde{\mathbf{A}}$ is still a good approximation of \mathbf{A} . The previous conditions, together with (3.1) imply that

$$\|\mathbf{R}^{-1}\mathbf{E}\|_\infty = \|\mathbf{R}^{-1}\mathbf{E}\mathbb{1}\|_\infty = \|\mathbf{R}^{-1}(\mathbf{A} - (1 + \epsilon')^{-1}\widehat{\mathbf{A}})\mathbb{1}\|_\infty \leq \left(1 - \frac{1 - \epsilon'}{1 + \epsilon'}\right) \leq 2\epsilon'.$$

and similarly,

$$\|\mathbf{C}^{-1}\mathbf{E}\|_1 = \|\mathbf{E}^\top \mathbf{C}^{-1} \mathbb{1}\|_\infty = \|(\mathbf{A} - (1 + \epsilon')^{-1}\widehat{\mathbf{A}})^\top \mathbf{C}^{-1} \mathbb{1}\|_\infty \leq \left(1 - \frac{1 - \epsilon'}{1 + \epsilon'}\right) \leq 2\epsilon'.$$

Applying Lemma B.4 then yields:

$$\begin{aligned} & \|\mathbf{R}^{-1/2}(\mathbf{A} - \widetilde{\mathbf{A}})\mathbf{C}^{-1/2}\|_2 \\ & \leq \|\mathbf{R}^{-1/2}(\mathbf{A} - \widehat{\mathbf{A}})\mathbf{C}^{-1/2}\|_2 + \left(1 - \frac{1}{1 + \epsilon'}\right) \|\mathbf{R}^{-1/2}\widehat{\mathbf{A}}\mathbf{C}^{-1/2}\|_2 + \|\mathbf{R}^{-1/2}\mathbf{E}\mathbf{C}^{-1/2}\|_2 \\ & \leq \epsilon' + \frac{\epsilon'}{1 + \epsilon'} + 2\epsilon' \leq 4\epsilon' = \epsilon. \end{aligned}$$

□

This lemma immediately implies the following fact on providing sparse approximations to directed (not necessarily Eulerian) Laplacians.

Corollary 3.11 (Crude Sparsification of Directed Laplacian). *Let $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n \times n}$ be a directed Laplacian associated with a (not necessarily Eulerian) graph G that has edges incident to at most v vertices and let $\epsilon, p \in (0, 1)$. The routine `SPARSIFYSUBGRAPH`(\mathcal{L}, p, ϵ) computes a directed Laplacian $\widetilde{\mathcal{L}} = \mathbf{D} - \widetilde{\mathbf{A}}^\top$ in time $O(\text{nnz}(\mathcal{L}) + v\epsilon^{-2} \log(v/p))$ such that*

1. $\tilde{\mathcal{L}}$ is sparse, i.e. $\text{nnz}(\tilde{\mathcal{L}}) = O(v\epsilon^{-2} \log(v/p))$,
2. the in and out degrees of the graphs associated with \mathcal{L} and $\tilde{\mathcal{L}}$ are the same, i.e. $\mathbf{A}\mathbb{1} = \tilde{\mathbf{A}}\mathbb{1}$ and $\mathbf{A}^\top \mathbb{1} = \tilde{\mathbf{A}}^\top \mathbb{1}$,
3. $\|\mathbf{D}_{in}^{-1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{D}_{out}^{-1/2}\|_2 \leq \epsilon$ with probability at least $1 - p$ where $\mathbf{D}_{in} = \mathbf{diag}(\mathbf{A}^\top \mathbb{1})$ and $\mathbf{D}_{out} = \mathbf{diag}(\mathbf{A}\mathbb{1})$ are the diagonal matrices associated with the in and out degrees of G .

Proof. This follows by applying the sampling result from Lemma 3.10 to \mathbf{A} , after removing the rows and columns corresponding to isolated vertices. The guarantees still hold on the larger matrix after inserting the zero rows and columns back. We can then substitute the corresponding directed Laplacians matrices into the formulas, since the diagonal terms will cancel (note that this follows again from Lemma 3.10, since the sparsified Laplacian has the same out degrees as the original one). \square

Using this, we prove the main result of this section, how to sparsify a subgraph that is contained in an expander, or more formally a particular undirected graph with large spectral gap. For notational convenience, we first formally define the type of graph symmetrization we consider for these subgraph arguments. Using this notation we then provide the main result of this subsection, Lemma 3.13. Pseudocode of this routine is in Figure 3.1.

Definition 3.12 (Graph Symmetrization). For a directed Laplacian, $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$, its *graph symmetrization* $\mathbf{S}_{\mathcal{L}}$ is the symmetric Laplacian given by $\mathbf{S}_{\mathcal{L}} \stackrel{\text{def}}{=} \mathbf{diag}(\mathbf{U}\mathbf{A}\mathbb{1}) - \mathbf{U}\mathbf{A}$. Equivalently, if we considering the graph associated with \mathcal{L} and replace every directed edge with an undirected edge of half the weight, then $\mathbf{S}_{\mathcal{L}}$ is the symmetric Laplacian associated with this undirected graph.

Lemma 3.13 (Subgraph Sparsification). *Let \mathcal{L} be a directed Laplacian and \mathbf{U} be an undirected Laplacian with spectral gap at least α , support size v , and $\mathbf{diag}(\mathbf{S}_{\mathcal{L}}) \preceq \mathbf{diag}(\mathbf{U})$. For $\delta \leq \alpha\epsilon/2$ the routine $\text{SPARSIFYSUBGRAPH}(\mathcal{L}, p, \delta)$ in time $O(\text{nnz}(\mathcal{L}) + v\delta^{-2} \log(v/p))$ computes a Laplacian $\tilde{\mathcal{L}}$ with the same in and out degrees as \mathcal{L} , $\text{nnz}(\tilde{\mathcal{L}}) = O(v\delta^{-2} \log(v/p))$, and $\|\mathbf{U}^{+1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{U}^{+1/2}\|_2 \leq \epsilon$ with probability at least $1 - p$.*

Proof. Without loss of generality let $\mathcal{L} = \mathbf{D}_{out} - \mathbf{A}^\top$. Furthermore, let $\mathbf{D} = \mathbf{diag}(\mathbf{U})$, $\mathbf{D}_{in} = \mathbf{diag}(\mathbf{A}\mathbb{1})$, and $\mathbf{D}_{out} = \mathbf{diag}(\mathbf{A}^\top \mathbb{1})$. By Corollary 3.11, we can compute a directed Laplacian $\tilde{\mathcal{L}} = \mathbf{D}_{out} - \tilde{\mathbf{A}}$ with in and out degrees being the same as those of \mathcal{L} , $\text{nnz}(\tilde{\mathcal{L}}) = O(v\delta^{-2} \log(v/p))$, and with probability at least $1 - p$

$$\|\mathbf{D}_{in}^{-1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{D}_{out}^{-1/2}\|_2 = \|\mathbf{D}_{in}^{-1/2}(\mathbf{A} - \tilde{\mathbf{A}})\mathbf{D}_{out}^{-1/2}\|_2 \leq \delta.$$

If this happens, then since \mathcal{L} and $\tilde{\mathcal{L}}$ have the same in and out degrees and $\mathbf{diag}(\mathbf{S}_{\mathcal{L}}) \preceq \mathbf{diag}(\mathbf{U})$ we have that \mathbf{U} has larger support than \mathcal{L} and $\tilde{\mathcal{L}}$, and therefore $\ker(\mathbf{U}) \subseteq \ker(\mathcal{L} - \tilde{\mathcal{L}}) \cap \ker((\mathcal{L} - \tilde{\mathcal{L}})^\top)$. Consequently, by Lemma 3.5 we have

$$\|\mathbf{U}^{+1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{U}^{+1/2}\|_2 = \max_{\vec{x}, \vec{y} \neq \mathbf{0}} \frac{\vec{x}^\top (\mathcal{L} - \tilde{\mathcal{L}}) \vec{y}}{\sqrt{\vec{x}^\top \mathbf{U} \vec{x} \cdot \vec{y}^\top \mathbf{U} \vec{y}}}.$$

Now, clearly, there is a maximizing $\vec{x}, \vec{y} \perp \mathbb{1}$. Consequently $\vec{x}' = \vec{x} - \frac{\vec{x}^\top \mathbf{d}}{\|\vec{d}\|_1} \mathbb{1}$ and $\vec{y}' = \vec{y} - \frac{\vec{y}^\top \vec{d}}{\|\vec{d}\|_1} \mathbb{1}$ are nonzero and satisfy $(\vec{x}')^\top \vec{d} = 0$ and $(\vec{y}')^\top \vec{d} = 0$. The spectral gap of \mathbf{U} being at least α implies

$$\mathbf{U} \succeq \alpha \left(\mathbf{D} - \frac{1}{\|\vec{d}\|_1} \vec{d} \vec{d}^\top \right),$$

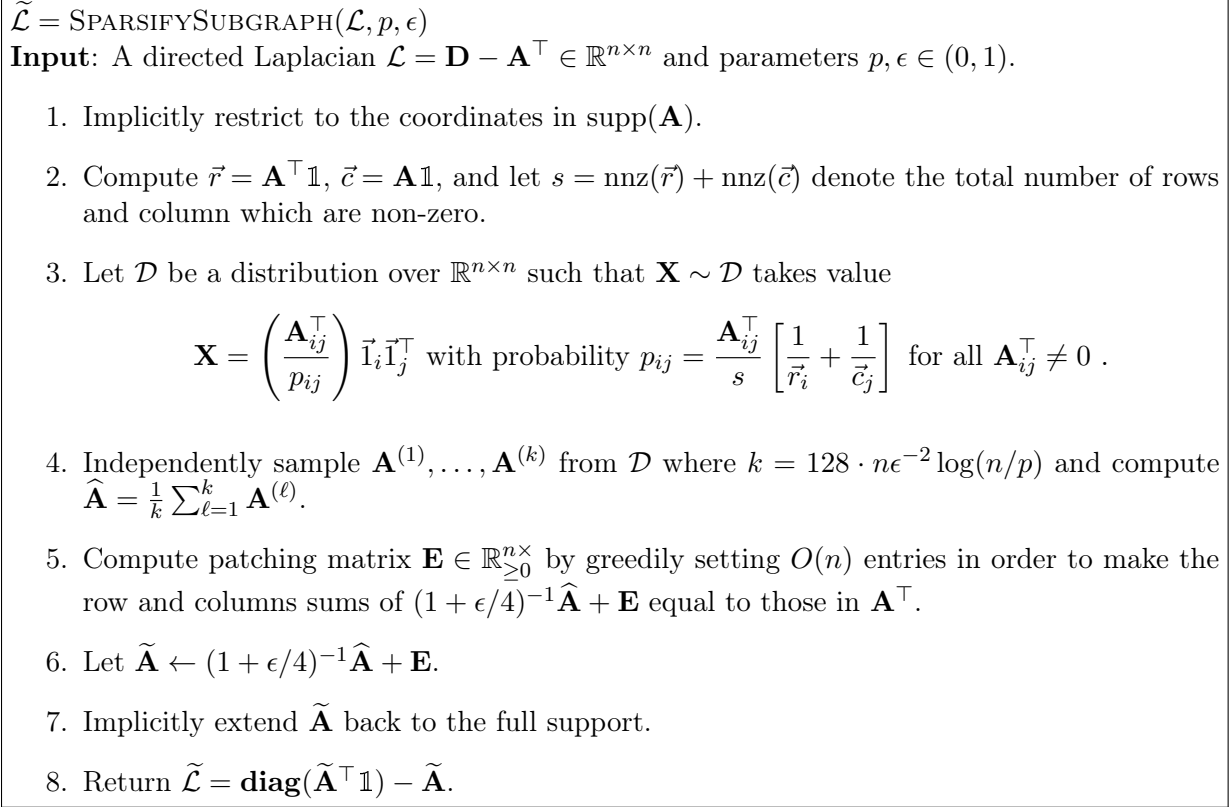


Figure 3.1: Pseudocode of the subgraph sparsification routine

which then gives

$$\|\mathbf{U}^{+1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{U}^{+1/2}\|_2 = \frac{(\vec{x}')^\top (\mathcal{L} - \tilde{\mathcal{L}}) \vec{y}'}{\sqrt{(\vec{x}')^\top \mathbf{U} \vec{x}' \cdot (\vec{y}')^\top \mathbf{U} \vec{y}'}} \leq \frac{1}{\alpha} \cdot \frac{(\vec{x}')^\top (\mathcal{L} - \tilde{\mathcal{L}}) \vec{y}'}{\sqrt{(\vec{x}')^\top \mathbf{D} \vec{x}' \cdot (\vec{y}')^\top \mathbf{D} \vec{y}'}} .$$

Since $\mathbf{D}_{in} \preceq 2 \cdot \mathbf{D}$ and $\mathbf{D}_{out} \preceq 2 \cdot \mathbf{D}$, applying Lemma 3.5 again yields

$$\|\mathbf{U}^{+1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{U}^{+1/2}\|_2 \leq \frac{2}{\alpha} \cdot \frac{(\vec{x}')^\top (\mathcal{L} - \tilde{\mathcal{L}}) \vec{y}'}{\sqrt{(\vec{x}')^\top \mathbf{D}_{in} \vec{x}' \cdot (\vec{y}')^\top \mathbf{D}_{out} \vec{y}'}} \leq \frac{2}{\alpha} \cdot \|\mathbf{D}_{in}^{-1/2}(\mathbf{A} - \tilde{\mathbf{A}})\mathbf{D}_{out}^{-1/2}\|_2 \leq \frac{2}{\alpha} \delta$$

and the result follows by our restriction on δ . □

3.3 Sparsifying an Eulerian Laplacian

Here we show how to produce an ϵ -sparsifier of an Eulerian Laplacian in nearly linear time. We achieve this by applying our result on sparsifying subgraphs (Lemma 3.13) proved in Section 3.2 on a decomposition of the Eulerian graph into well-connected pieces on an associated undirected graph.

The decomposition we use is essentially identical to the expander decomposition used in Spielman and Teng's work on graph sparsification [36]. Interestingly, the quality of our decomposition is measured only in terms of properties of the symmetrized graph, rather than of the original directed

graph. Ultimately, only the sampling probabilities that we use on the decomposition take into account edge direction.⁷

Below we formally define the type of decomposition we need, and provide a theorem about computing such decompositions. Finding these decompositions has been done in prior works [37, 21, 33, 32], and we defer the discussion of it to Appendix C.

Definition 3.14. An (s, α, β) -decomposition of a directed Laplacian \mathcal{L} is a decomposition of \mathcal{L} into directed Laplacians $\mathcal{L}^{(1)}, \dots, \mathcal{L}^{(k)} \in \mathbb{R}^{n \times n}$, i.e. $\mathcal{L} = \sum_{i \in [k]} \mathcal{L}^{(i)}$, such that $\sum_{i \in [k]} |\text{supp}(\mathcal{L}^{(i)})| \leq s$ and such that there exists undirected Laplacians $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(k)}$ such that:

1. $\text{diag}(\mathbf{S}_{\mathcal{L}^{(i)}}) \preceq \text{diag}(\mathbf{U}^{(i)})$, for all $i \in [k]$,
2. $\mathbf{U}^{(i)}$ has spectral gap at least α , for all $i \in [k]$, and
3. $\sum_{i \in [k]} \mathbf{U}^{(i)} \preceq \beta \mathbf{U}_{\mathcal{L}}$.

We call the $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(k)}$ with these properties an (α, β) undirected cover of $\mathcal{L}^{(1)}, \dots, \mathcal{L}^{(k)}$.

Theorem 3.15. Given a directed Laplacian, $\mathcal{L} \in \mathbb{R}^{n \times n}$, the routine $\text{FINDDECOMPOSITION}(\mathcal{L})$ returns an $(\tilde{O}(n), 1/\alpha, \beta)$ -decomposition, with $\alpha, \beta = \tilde{O}(1)$, in $\tilde{O}(\text{nnz}(\mathcal{L}))$ time.

We produce our sparsifiers by computing the decomposition using Theorem 3.15 and then applying Lemma 3.13 repeatedly to obtain the sparsifier. Pseudocode of this algorithm is given in Figure 3.2. and the analysis of this algorithm is given in the following theorem.

$\tilde{\mathcal{L}} = \text{SPARSIFYEULERIAN}(\mathcal{L}, p, \epsilon)$
Input: \mathcal{L} an $n \times n$ directed Laplacian, parameters $p, \epsilon \in (0, 1)$.

1. $(\mathcal{L}^{(1)}, \dots, \mathcal{L}^{(k)}, \alpha, \beta) \leftarrow \text{FINDDECOMPOSITION}(\mathcal{L})$.
2. For $i = 1, \dots, k$
 - (a) $\tilde{\mathcal{L}}^{(i)} \leftarrow \text{SPARSIFYSUBGRAPH}(\mathcal{L}^{(i)}, p/n^2, \epsilon/(2\alpha\beta))$.
3. Return $\tilde{\mathcal{L}} = \sum_{i=1}^k \tilde{\mathcal{L}}^{(i)}$.

Figure 3.2: Pseudocode of the Eulerian graph sparsification routine

Theorem 3.16. For Eulerian Laplacian $\mathcal{L} \in \mathbb{R}^{n \times n}$ and $\epsilon, p \in (0, 1)$ with probability at least $1 - p$ the routine $\text{SPARSIFYEULERIAN}(\mathcal{L}, p, \epsilon)$ computes in $\tilde{O}(\text{nnz}(\mathcal{L}) + n\epsilon^{-2} \log(1/p))$ time an Eulerian Laplacian $\tilde{\mathcal{L}} \in \mathbb{R}^{n \times n}$ such that

1. $\tilde{\mathcal{L}}$ is an ϵ -sparsifier of \mathcal{L} ,
2. the in and out degrees of the graphs associated with \mathcal{L} and $\tilde{\mathcal{L}}$ are identical.

⁷Even this can possibly be overcome by choosing different sampling probabilities. It is the patching of the graph, i.e. adding edges to preserve degree imbalance, where we truly use the directed structure of the graph.

Proof. Using the FINDDECOMPOSITION routine (Theorem 3.15) we compute Laplacians $\mathcal{L}^{(1)}, \dots, \mathcal{L}^{(k)} \in \mathbb{R}^{n \times n}$ that are a (s, α, β) partition of \mathcal{L} with (α, β) undirected cover $\mathbf{U}^{(1)}, \dots, \mathbf{U}^{(k)}$ for $s = \tilde{O}(n)$, $\alpha = 1/\tilde{O}(1)$, and $\beta = \tilde{O}(1)$.

We then apply the SPARSIFYSUBGRAPH routine (Lemma 3.13) to each $\mathcal{L}^{(i)}$ to compute $\tilde{\mathcal{L}}^{(i)}$ in $O(\text{nnz}(\mathcal{L}) + s\epsilon^{-2}\beta^{-2}\alpha^{-2}\log(n/p))$ time such that each $\tilde{\mathcal{L}}^{(i)}$ has the same in and out degree as $\mathcal{L}^{(i)}$ and $\|(\mathbf{U}^{(i)})^{+1/2}(\mathcal{L}^{(i)} - \tilde{\mathcal{L}}^{(i)})(\mathbf{U}^{(i)})^{+1/2}\|_2 \leq \epsilon/\beta$. The running time follows from the fact that $\sum_{i \in [k]} |\text{supp}(\mathcal{L}^{(i)})| \leq s$ and that happens with probability at least $1 - p$ by union bounding over the success probability of each call to SPARSIFYSUBGRAPH.

Finally, considering $\tilde{\mathcal{L}} = \sum_{i \in [k]} \tilde{\mathcal{L}}^{(i)}$, Lemma 3.5 yields that for all $\vec{x}, \vec{y} \neq 0$ it is the case that

$$\vec{x}^\top (\mathcal{L} - \tilde{\mathcal{L}}) \vec{y} = \sum_{i \in [k]} \vec{x}^\top (\mathcal{L}^{(i)} - \tilde{\mathcal{L}}^{(i)}) \vec{y} \leq \sum_{i \in [k]} \frac{\epsilon}{2\beta} \left[\vec{x}^\top \mathbf{U}^{(i)} \vec{x} + \vec{y}^\top \mathbf{U}^{(i)} \vec{y} \right] \leq \frac{\epsilon\beta}{2\beta} \left[\vec{x}^\top \mathbf{U}_{\mathcal{L}} \vec{x} + \vec{y}^\top \mathbf{U}_{\mathcal{L}} \vec{y} \right]$$

The result follows from Lemma 3.5 applied above and the the bounds on s, α , and β . Note that the fact that in and out degrees are preserved is guaranteed by the fact that for each component in the decomposition the degrees are preserved, according to Lemma 3.13. \square

3.4 Sparsifying a Squared Eulerian Laplacian

Here we build upon Section 3.3 and show how to sparsify certain implicitly represented Eulerian Laplacians. In particular, given an Eulerian Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$ associated with a strongly connected graph we show how to compute a sparsifier for the Eulerian Laplacian $\mathcal{M} = \mathbf{D} - \mathbf{A}^\top \mathbf{D}^{-1} \mathbf{A}^\top$ in nearly-linear time with respect to \mathcal{L} , i.e. without explicitly constructing \mathcal{M} . Note that running time we achieve may be sublinear in size of the matrix we are sparsifying, i.e \mathcal{M} .

Our approach is a natural directed extension of the approach taken by Peng and Spielman [33] for solving the same problem in the case when \mathcal{L} is symmetric. Broadly speaking, we decompose \mathcal{M} into a directed Laplacian for each vertex. Each of these directed Laplacians may be dense but we show that they have a compact representation that allows us to efficiently implement a sampling scheme analogous to SPARSIFYSUBGRAPH for each of these Laplacians such that adding these approximation yields an Eulerian approximation to \mathcal{M} that has $\text{nnz}(\mathcal{L})$ non-zero entries. Applying SPARSIFYEULERIAN from Section 3.3 to the result then yields our desired sparsifier.

Formally, we consider the slightly more general setting where we have a square matrix with non-negative entries, $\mathcal{A} \in \mathbb{R}_{\geq 0}^{n \times n}$, that has the same row and column sums, i.e. $\mathcal{A}\mathbf{1} = \mathcal{A}^\top \mathbf{1}$. We show how to compute a sparsifier of $\mathcal{M} = \mathbf{D} - \mathcal{A}\mathbf{D}^{-1}\mathcal{A}$ for $\mathbf{D} = \text{diag}(\mathcal{A}\mathbf{1})$ in time nearly linear in $\text{nnz}(\mathcal{A})$. This setting is more general as we allow entries on the diagonal of the squared matrix. We consider this case as it simplifies our analysis in Section 4.

As discussed, we first decompose \mathbf{M} into a directed Laplacian for each vertex. For $i \in [n]$ we let

$$\mathcal{L}^{(i)} = \text{diag}(\mathcal{A}_{i,:}) - \frac{1}{\mathbf{D}_{i,i}} \mathcal{A}_{:,i} \mathcal{A}_{i,:}^\top$$

where $\mathcal{A}_{i,:}, \mathcal{A}_{:,i} \in \mathbb{R}^n$ are the vectors corresponding to the row and column i of \mathcal{A} respectively. In Theorem 3.19 we show that \mathcal{M} and each $\mathcal{L}^{(i)}$ are directed Laplacians such that $\mathcal{M} = \sum_{i \in [n]} \mathcal{L}^{(i)}$.

Note that while \mathcal{M} may be dense and forming each of the $\mathcal{L}^{(i)}$ explicitly maybe expensive, we have a compact representation of each $\mathcal{L}^{(i)}$ in terms of a single row and column of \mathcal{A} . Moreover, if we look at the total support of $\mathcal{L}^{(i)}$ then since each row and column only appears once we see that the total support is just $O(\text{nnz}(\mathcal{A}))$. Furthermore, we show that due to the low rank structure of the graph symmetrization, $\mathbf{S}_{\mathcal{L}^{(i)}}$, of each $\mathcal{L}^{(i)}$ has spectral gap of at least a constant (See Lemma 3.17).

Consequently, if we apply SPARSIFYSUBGRAPH to each $\mathcal{L}^{(i)}$ and sum the results, the analysis in Section 3.3 would imply that this matrix would be an approximation to \mathcal{M} with $O(\text{nnz}(\mathcal{A}))$ non-zero entries.

The only difficulty in following this approach is to show that we can apply SPARSIFYSUBGRAPH to each $\mathcal{L}^{(i)}$ efficiently. We show that we can perform this operation in time proportional to the number of non-zeros in each row and column of \mathcal{A} , rather than the naive $O(\text{nnz}(\mathcal{L}^{(i)}))$ running time which could be much larger. To do this, as in Peng and Spielman [33], we exploit the simple product structure of $\mathcal{L}^{(i)}$. Since \mathcal{A} has the same row and column sums this means that $\|\mathcal{A}_{:,i}\|_1 = \|\mathcal{A}_{i,:}\|_1 = \mathbf{D}_{i,i}$. Consequently, each $\mathcal{L}^{(i)}$ is of the form $\mathbf{N} = \mathbf{diag}(\vec{y}) - \frac{1}{\|\vec{y}\|_1} \vec{x} \vec{y}^\top$ for some $\vec{x}, \vec{y} \in \mathbb{R}_{\geq 0}^n$ with $\|\vec{x}\|_1 = \|\vec{y}\|_1$ that depend on i . Since, the off-diagonals and their row and column sums have simple closed form expressions we can show that with $O(\text{nnz}(\vec{x}) + \text{nnz}(\vec{y}))$ preprocessing time, we can sample from the distribution required to apply SPARSIFYSUBGRAPH on this matrix in $O(1)$ time. Consequently, we can implement the approach in our desired running time.

In the remainder of this section we provide pseudocode for these routines and prove their correctness. First, in Figure 3.3 we provide SPARSIFYPRODUCT the pseudocode for sparsifying matrices of the form \mathbf{N} given above. In Lemma 3.17 we prove that the graph symmetrizations, $\mathbf{S}_{\mathbf{N}}$, of these matrices have spectral gap of at least 1 and using this fact in Lemma 3.18 we prove that SPARSIFYPRODUCT does provide sparse approximations to these matrices. In Figure 3.4 we provide SPARSIFY SQUARE the pseudocode for producing sparsifiers of \mathcal{M} by invoking SPARSIFYPRODUCT on each $\mathcal{L}^{(i)}$ and then invoking SPARSIFYEULERIAN on the result. Finally, we conclude the section with Theorem 3.19 which proves the correctness and analyzes the running time of SPARSIFYPRODUCT.

Lemma 3.17. *For $\vec{x}, \vec{y} \in \mathbb{R}_{\geq 0}^n$ with $r = \|\vec{x}\|_1 = \|\vec{y}\|_1 > 0$ and $\mathbf{Y} = \mathbf{diag}(\vec{y})$, the matrix $\mathcal{L} = \mathbf{Y} - \frac{1}{r} \vec{x} \vec{y}^\top$ is a directed Laplacian and the spectral gap of its symmetrization $\mathbf{S}_{\mathcal{L}}$ is at least 1.*

Proof. Note that $\mathcal{L}_{ij} = -\frac{1}{r} x_i y_j \leq 0$ for $i \neq j$ and $\mathbf{1}^\top \mathcal{L} = \mathbf{1}^\top \mathbf{Y} - \frac{1}{r} \mathbf{1}^\top \vec{x} \vec{y}^\top = \vec{y}^\top - \vec{y}^\top = \mathbf{0}^\top$. Consequently, \mathcal{L} is a directed Laplacian. All that remains is to lower bound the spectral gap of $\mathbf{S}_{\mathcal{L}}$.

Letting $\mathbf{X} = \mathbf{diag}(\vec{x})$, we see that the graph symmetrization $\mathbf{S}_{\mathcal{L}}$ of \mathcal{L} is the undirected Laplacian

$$\mathbf{S}_{\mathcal{L}} = \frac{1}{2} \left(\mathbf{X} + \mathbf{Y} - \frac{1}{r} \left(\vec{x} \vec{y}^\top + \vec{y} \vec{x}^\top \right) \right).$$

Furthermore, the diagonal entries of $\mathbf{S}_{\mathcal{L}}$, denoted $\vec{d} = \mathbf{diag}(\mathbf{S}_{\mathcal{L}})$, are given by

$$\vec{d}_i = [\mathbf{S}_{\mathcal{L}}]_{ii} = \frac{1}{2} (\vec{x}_i + \vec{y}_i) - \frac{1}{r} \vec{x}_i \vec{y}_i \leq \frac{1}{2} (\vec{x}_i + \vec{y}_i).$$

Now recall that the spectral gap of $\mathbf{S}_{\mathcal{L}}$ is defined to be the smallest nonzero eigenvalue of the normalized Laplacian $\mathbf{D}^{-1/2} \mathbf{S}_{\mathcal{L}} \mathbf{D}^{-1/2}$, where $\mathbf{D} = \mathbf{diag}(\vec{d})$. Since $d_i \leq \frac{1}{2} (\vec{x}_i + \vec{y}_i)$, we have

$$\mathbf{D}^{-1/2} \mathbf{S}_{\mathcal{L}} \mathbf{D}^{-1/2} \succeq \left(\frac{1}{2} (\mathbf{X} + \mathbf{Y}) \right)^{-1/2} \mathbf{S}_{\mathcal{L}} \left(\frac{1}{2} (\mathbf{X} + \mathbf{Y}) \right)^{-1/2} = 2 (\mathbf{X} + \mathbf{Y})^{-1/2} \mathbf{S}_{\mathcal{L}} (\mathbf{X} + \mathbf{Y})^{-1/2} =: \mathbf{M}.$$

This implies that the eigenvalues of $\mathbf{D}^{-1/2} \mathbf{S}_{\mathcal{L}} \mathbf{D}^{-1/2}$ dominate those of \mathbf{M} . The multiplicity of zero as an eigenvalue is the same for the two matrices, so it thus suffices to show that the smallest nonzero eigenvalue of \mathbf{M} is at least 1. Plugging the definition of $\mathbf{S}_{\mathcal{L}}$ in our expression for \mathbf{M} gives

$$\begin{aligned} \mathbf{M} &= 2 (\mathbf{X} + \mathbf{Y})^{-1/2} \mathbf{S}_{\mathcal{L}} (\mathbf{X} + \mathbf{Y})^{-1/2} \\ &= (\mathbf{X} + \mathbf{Y})^{-1/2} \left(\mathbf{X} + \mathbf{Y} - \frac{1}{r} \left(\vec{x} \vec{y}^\top + \vec{y} \vec{x}^\top \right) \right) (\mathbf{X} + \mathbf{Y})^{-1/2} = \mathbf{I} - \mathbf{N}, \end{aligned}$$

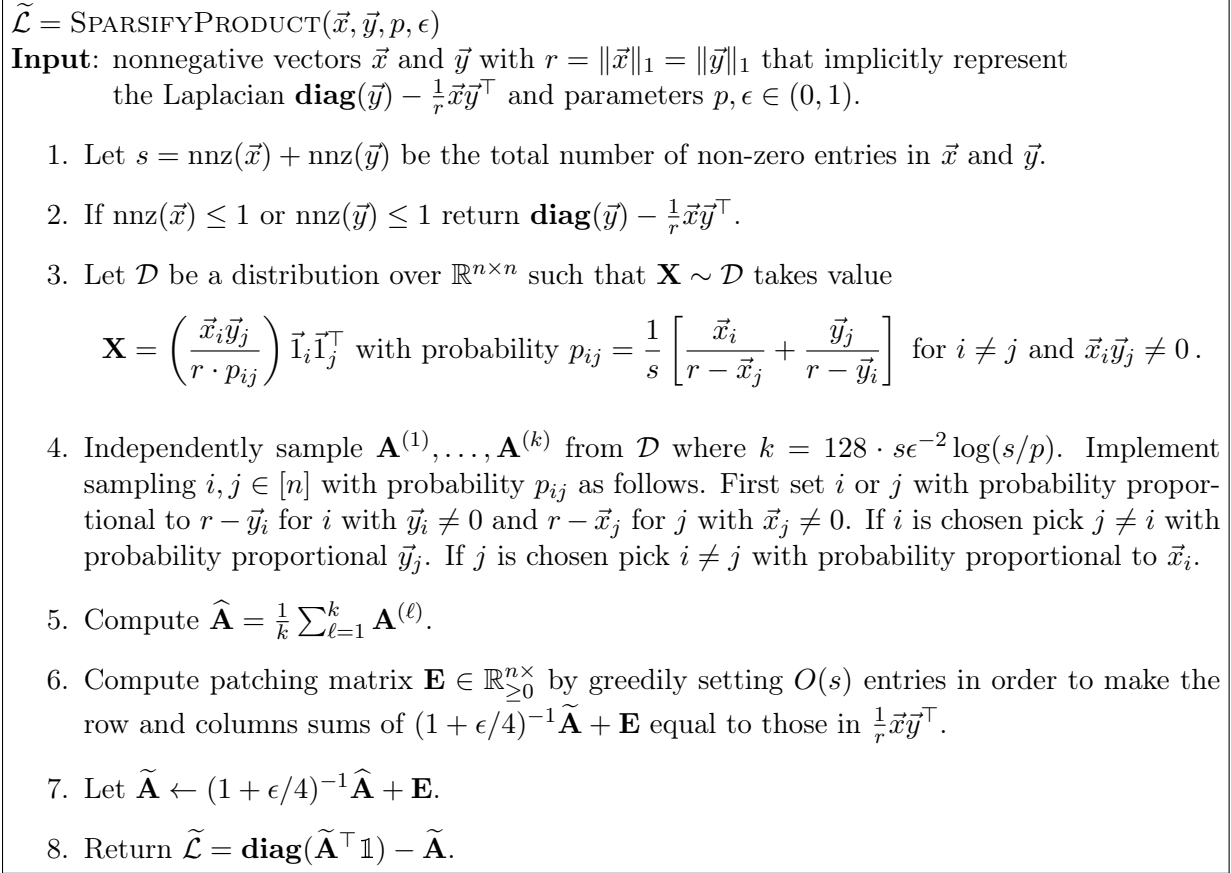


Figure 3.3: Pseudocode for sparsifying a single product graph

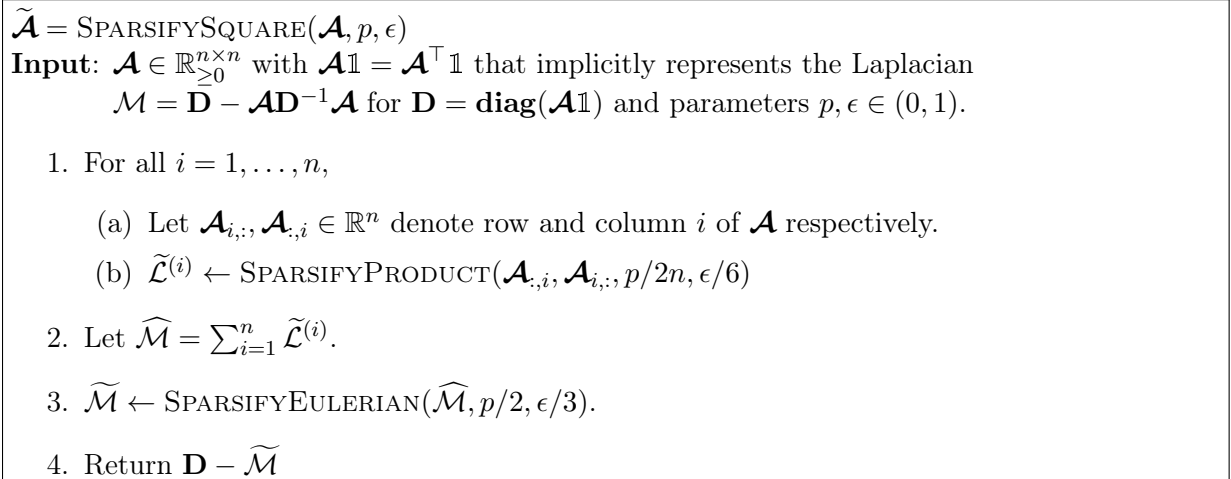


Figure 3.4: Pseudocode for producing an ϵ -sparsifier of $\mathcal{M} = \mathbf{D} - \mathcal{A}^\top \mathbf{D}^{-1} \mathcal{A}^\top$,

where $\mathbf{N} = \frac{1}{r}(\mathbf{X} + \mathbf{Y})^{-1/2}(\vec{x}\vec{y}^\top + \vec{y}\vec{x}^\top)(\mathbf{X} + \mathbf{Y})^{-1/2}$. The matrix \mathbf{N} has rank 2, so $\mathbf{I} - \mathbf{N}$ has at most 2 eigenvalues that are not equal to 1. Furthermore, we know that \mathbf{M} has a nontrivial kernel, so at least one of these eigenvalues is 0. Let λ be the one remaining eigenvalue. Since $\text{tr}(\mathbf{M})$ equals the sum of these eigenvalues, we have $\text{tr}(\mathbf{M}) = (n - 2) \cdot 1 + 0 + \lambda = n - 2 + \lambda$, so

$$\lambda = \text{tr}(\mathbf{M}) - n + 2 = \text{tr}(\mathbf{I}) - \text{tr}(\mathbf{N}) - n + 2 = 2 - \text{tr}(\mathbf{N}).$$

The inequality $\frac{2ab}{a+b} \leq \frac{a+b}{2}$ between the harmonic and arithmetic means then gives

$$\text{tr}(\mathbf{N}) = \sum_{i \in [n]} \mathbf{N}_{ii} = \frac{1}{r} \sum_{i \in [n]} \frac{2\vec{x}_i\vec{y}_i}{\vec{x}_i + \vec{y}_i} \leq \frac{1}{r} \sum_{i \in [n]} \frac{\vec{x}_i + \vec{y}_i}{2} = 1,$$

so $\lambda = 2 - \text{tr}(\mathbf{N}) \geq 1$. The nonzero eigenvalues of \mathbf{M} are thus all at least 1, as desired. \square

Lemma 3.18 (Product Sparsification). *Let $\vec{x}, \vec{y} \in \mathbb{R}_{\geq 0}^n$ be non-negative vectors with $\|\vec{x}\|_1 = \|\vec{y}\|_1 = r$ and let $\epsilon, p \in (0, 1)$. Furthermore, let s denote the total number of non-zero entries in \vec{x} and \vec{y} , i.e. $s = \text{nnz}(\vec{x}) + \text{nnz}(\vec{y})$ and let $\mathcal{L} \stackrel{\text{def}}{=} \mathbf{diag}(\vec{y}) - \frac{1}{r}\vec{x}\vec{y}^\top$. The routine SPARSIFYPRODUCT($\vec{x}, \vec{y}, p, \epsilon/2$) in time $O(s\epsilon^{-2} \log(s/p))$ computes with probability at least $1 - p$ a Laplacian $\tilde{\mathcal{L}}$ with the same in and out degrees as \mathcal{L} , $\text{nnz}(\tilde{\mathcal{L}}) = O(s\epsilon^{-2} \log(s/p))$, and $\|\mathbf{S}_{\mathcal{L}}^{+1/2}(\mathcal{L} - \tilde{\mathcal{L}})\mathbf{S}_{\mathcal{L}}^{+1/2}\|_2 \leq \epsilon$.*

Proof. Note that if $\text{nnz}(\vec{x}) \leq 1$ or $\text{nnz}(\vec{y}) \leq 1$. Then clearly $\frac{1}{r}\vec{x}\vec{y}^\top$ has at most s non-zero entries and \mathcal{L} has $O(s)$ non-zero entries. Furthermore, we can clearly compute \mathcal{L} in $O(s)$ time and therefore the result follows. In the remainder of the proof we therefore assume that $\text{nnz}(\vec{x}) \geq 2$ and $\text{nnz}(\vec{y}) \geq 2$.

First, we show that $\tilde{\mathcal{L}}$ is precisely the output of an execution of SPARSIFYSUBGRAPH($\mathcal{L}, p, \epsilon/2$). Let $\mathbf{X} = \mathbf{diag}(\vec{x})$, $\mathbf{Y} = \mathbf{diag}(\vec{y})$, $\mathbf{A}^\top \stackrel{\text{def}}{=} \frac{1}{r}[\vec{x}\vec{y}^\top - \mathbf{X}\mathbf{Y}]$, and $\mathbf{D} = \mathbf{Y} - \frac{1}{r}\mathbf{X}\mathbf{Y}$. Clearly, $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$. Furthermore, we see that \mathbf{A}^\top is non-negative with a zero-diagonal $\mathbf{diag}(\mathbf{A}^\top) = \mathbf{0}$ and therefore \mathbf{D} is diagonal and this is the standard decomposition of \mathcal{L} .

Now as in SPARSIFYSUBGRAPH(\mathcal{L}, p, ϵ) let $\vec{r} = \mathbf{A}^\top \mathbf{1} = \vec{x} - \frac{1}{r}\mathbf{X}\mathbf{Y}\mathbf{1}$ and $\vec{c} = \mathbf{A}\mathbf{1} = \vec{y} - \frac{1}{r}\mathbf{X}\mathbf{Y}\mathbf{1}$. Furthermore, since $\text{nnz}(\vec{x}) \geq 2$ and $\text{nnz}(\vec{y}) \geq 2$ we see that a row or column of $\frac{1}{r}\vec{x}\vec{y}^\top$ is non-zero if and only if the corresponding row and column in \mathbf{A}^\top is non-zero and thus s is the same in SPARSIFYSUBGRAPH(\mathcal{L}, p, ϵ) and SPARSIFYPRODUCT($\vec{x}, \vec{y}, p, \epsilon$). Now, for all $i \neq j$ with $\mathbf{A}_{ij}^\top = \vec{x}_i\vec{y}_j \neq 0$ we have

$$\frac{\mathbf{A}_{ij}^\top}{s} \left[\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right] = \frac{\vec{x}_i\vec{y}_j}{r \cdot s} \left[\frac{1}{x_i - \frac{1}{r}\vec{x}_i\vec{y}_i} + \frac{1}{y_j - \frac{1}{r}\vec{x}_j\vec{y}_j} \right] = \frac{1}{s} \left[\frac{\vec{x}_i}{r - \vec{x}_j} + \frac{\vec{y}_j}{r - \vec{y}_i} \right].$$

Consequently, we see that $\tilde{\mathcal{L}}$ is precisely the output of an execution of SPARSIFYSUBGRAPH($\mathcal{L}, p, \epsilon/2$). Furthermore, since $\mathbf{S}_{\mathcal{L}}$ has spectral gap at least 1 by Lemma 3.17 we have by Lemma 3.13 which analyzed SPARSIFYSUBGRAPH that $\tilde{\mathcal{L}}$ has the desired properties.

All that remains is to bound the running time of SPARSIFYPRODUCT($\vec{x}, \vec{y}, p, \epsilon/2$). Note that computing s takes $O(s)$ time and computing all the $r - \vec{y}_i$ and $r - \vec{x}_j$ can be done in $O(s)$ time. Consequently, with $O(s)$ preprocessing we can build a table so that each sample from \mathcal{D} takes $O(1)$ time and computing $\tilde{\mathbf{A}}$ takes $O(s + k) = O(s\epsilon^{-2} \log(s/p))$ time. Furthermore, since computing the patching \mathbf{E} also takes $O(s + k)$ time we have that the total running time is as desired. \square

Theorem 3.19 (Sparsifying Squares). *Let $\mathcal{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ have the same row and column sums, i.e. $\mathcal{A}\mathbf{1} = \mathcal{A}^\top \mathbf{1}$, and let $\epsilon, p \in (0, 1)$. Let $\mathbf{D} = \mathbf{diag}(\mathcal{A}\mathbf{1})$, $\mathcal{L} = \mathbf{D} - \mathcal{A}$, and $\mathcal{M} = \mathbf{D} - \mathcal{A}\mathbf{D}^{-1}\mathcal{A}$. Both \mathcal{L} and \mathcal{M} are Eulerian Laplacians and in $O(\text{nnz}(\mathcal{L})\epsilon^{-2} \log(n/p))$ time the routine SPARSIFYSQUARE(\mathcal{L}, p, ϵ) computes $\tilde{\mathcal{A}}$ such that with probability at least $1 - p$ the matrix $\tilde{\mathcal{M}} = \mathbf{D} - \tilde{\mathcal{A}}$ is an ϵ -sparsifier of \mathcal{M} .*

Proof. Clearly, both \mathcal{A} and $\mathcal{A}\mathbf{D}^{-1}\mathcal{A}$ are entrywise non-negative and therefore the off diagonals of \mathcal{L} and \mathcal{M} are non-positive. Furthermore, since clearly $\mathcal{A}\mathbf{1} = \mathcal{A}^\top \mathbf{1} = \mathcal{A}\mathbf{D}^{-1}\mathcal{A}\mathbf{1} = [\mathcal{A}\mathbf{D}^{-1}\mathcal{A}]^\top \mathbf{1} = \mathbf{D}\mathbf{1}$ we have that $\mathcal{L}\mathbf{1} = \mathcal{L}^\top \mathbf{1} = \mathcal{M}\mathbf{1} = \mathcal{M}^\top \mathbf{1} = \mathbf{0}$ and both \mathcal{L} and \mathcal{M} are Eulerian Laplacians.

Next, for all $i \in [n]$ let $s_i = \text{nnz}(\mathcal{A}_{:,i}) + \text{nnz}(\mathcal{A}_{i,:})$ and

$$\mathcal{L}^{(i)} = \mathbf{diag}(\mathcal{A}_{i,:}) - \frac{1}{\mathbf{D}_{i,i}} \mathcal{A}_{:,i} \mathcal{A}_{i,:}^\top.$$

Note that since $\mathcal{A}\mathbf{1} = \mathcal{A}^\top \mathbf{1}$ and \mathcal{A} is entrywise non-negative we have that $\mathbf{D}_{i,i} = \|\mathcal{A}_{i,:}\|_1 = \|\mathcal{A}_{:,i}\|_1$. Consequently, by Lemma 3.18 and union bound with probability at least $1 - \frac{1}{2p}$ it is the case that each $\tilde{\mathcal{L}}^{(i)}$ is a directed Laplacian with the same in and out degrees as $\mathcal{L}^{(i)}$, $\text{nnz}(\tilde{\mathcal{L}}^{(i)}) = O(s_i \epsilon^{-2} \log(sn/p))$, and

$$\left\| \mathbf{S}_{\mathcal{L}^{(i)}}^{+/2} (\mathcal{L}^{(i)} - \tilde{\mathcal{L}}^{(i)}) \mathbf{S}_{\mathcal{L}^{(i)}}^{+/2} \right\|_2 \leq \epsilon/3. \quad (3.2)$$

Furthermore, since clearly $\sum_i s_i = 2\text{nnz}(\mathcal{A})$ we have that

$$\text{nnz}(\widehat{\mathcal{M}}) \leq \sum_{i \in [n]} \text{nnz}(\tilde{\mathcal{L}}^{(i)}) \leq \sum_{i \in [n]} s_i \epsilon^{-2} \log(ns_i/p) \leq O(\text{nnz}(\mathcal{A}) \epsilon^{-2} \log(n/p))$$

and therefore the total running time for computing $\widehat{\mathcal{M}}$ is $O(\text{nnz}(\mathcal{L}) \epsilon^{-2} \log(n/p))$. Using Theorem 3.16 to reason about the effect of SPARSIFYEULERIAN then completes our running time analysis and union bounding yields that $\widehat{\mathcal{M}}$ has the desired degrees and sparsity. Consequently, $\tilde{\mathcal{A}}$ also has the desired sparsity and since the degrees of the graph associated with \mathcal{M} are at most the degrees of the graph associated with \mathcal{L} we see that $\tilde{\mathcal{A}} \in \mathbb{R}_{\geq 0}^{n \times n}$.

All that remains is to verify that $\widehat{\mathcal{M}}$ is an ϵ -approximation of \mathcal{M} . By Lemma 3.5, (3.2) implies that for all $\vec{x}, \vec{y} \neq \mathbf{0}$ it is the case that

$$\vec{x}^\top (\widehat{\mathcal{M}} - \mathcal{M}) \vec{y} = \sum_{i=1}^n \vec{x}^\top (\tilde{\mathcal{L}}^{(i)} - \mathcal{L}^{(i)}) \vec{y} \leq \sum_{i=1}^n \frac{\epsilon}{3} \left[\vec{x}^\top \mathbf{S}_{\mathcal{L}^{(i)}} \vec{x} + \vec{y}^\top \mathbf{S}_{\mathcal{L}^{(i)}} \vec{y} \right] = \frac{\epsilon}{3} \left[\vec{x}^\top \mathbf{U}_{\mathcal{M}} \vec{x} + \vec{y}^\top \mathbf{U}_{\mathcal{M}} \vec{y} \right]$$

where in the last identity we used that $\sum_{i=1}^n \mathbf{S}_{\mathcal{L}^{(i)}} = \mathbf{U}_{\mathcal{M}}$ from the fact that $\sum_{i \in [n]} \mathcal{L}^{(i)} = \mathcal{M}$. Applying Lemma 3.5 again on the above bound, we obtain that $\widehat{\mathcal{M}}$ is an $\epsilon/3$ approximation of \mathcal{M} . Since, Theorem 3.16 implies that $\tilde{\mathcal{M}}$ is an $\epsilon/3$ -approximation of $\widehat{\mathcal{M}}$, invoking the transitivity bound, Lemma 3.8, yields that $\tilde{\mathcal{M}}$ is an $(\epsilon/3)(2 + \epsilon/3) \leq \epsilon$ -approximation of \mathcal{M} as desired. \square

4 Solving Directed Laplacian Systems

In this section, we show how to solve directed Laplacian systems in almost-linear time. Our main result is as follows:

Theorem 4.1. *Let \mathbf{M} be an arbitrary $n \times n$ column-diagonally-dominant or row-diagonally-dominant matrix with diagonal \mathbf{D} and m non-zero entries. Let $\kappa(\mathbf{D})$ be the ratio between the maximum and minimum diagonal entries of \mathbf{D} . Then for any $\vec{b} \in \text{im}(\mathbf{M})$ and $0 < \epsilon \leq 1$, one can compute, with high probability and in time*

$$\tilde{O} \left(\left(m + n 2^{O(\sqrt{\log n \log \log n})} \right) \log^3 \left(\frac{\kappa(\mathbf{D}) \cdot \kappa(\mathbf{M})}{\epsilon} \right) \right)$$

a vector \vec{x}' satisfying $\|\mathbf{M}\vec{x}' - \vec{b}\|_2 \leq \epsilon \|\vec{b}\|_2$.

Note that column-diagonally-dominant matrices include Laplacians of directed graphs. This bound follows from combining the reduction to solving linear systems in Eulerian Laplacians stated in Theorem 42 of [12] with our main solver result. The condition number of the undirected Laplacians that arise can be bounded by $O(\kappa(\mathbf{D}) \cdot \kappa(\mathbf{M}))$ by the preceding Theorem 41 in [12]. This condition number becomes a logarithmic overhead by the condition number reductions in Appendix F, which allows us to focus on solving $\text{poly}(n)$ conditioned Eulerian systems.

The result that we will focus on in this section is an algorithm that given an Eulerian Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n \times n}$ with m non-zero entries computes an ϵ -approximate solution to $\mathcal{L}\vec{x} = \vec{b}$ in time $\tilde{O}((m + n \exp(O(\sqrt{\log \kappa \log \log \kappa}))) \log(1/\epsilon))$ where $\kappa = \kappa(\mathbf{U}_{\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}})$. Note that $\exp(O(\sqrt{\log(\kappa) \log \log \kappa}))$ is a term that is $\kappa^{o(1)}$, i.e. it grows less than $O(\kappa^\epsilon)$ for any constant $\epsilon > 0$, whereas iterative methods for solving such systems typically have a dependence of $\kappa^{1/2}$ or higher in their running time. An overview of the main components of this algorithm is in Section 2.3.3.

4.1 Preconditioned Richardson Iteration and Approximate Pseudoinverse

The key iterative method that we use to build our solver is the preconditioned Richardson iteration. It can be thought of as a general-purpose tool that boosts the quality of a linear system solver by iteratively applying the solver on the residual.⁸ In this section we describe the preconditioned Richardson iteration, and based on it, we derive a measure of quality of a linear system solver in terms of how well it functions as an approximate pseudoinverse for the matrix involved in the system we want to solve. In Section 4.4 we analyze our solver chain in terms of this notion.

The Richardson iteration refers to perhaps one of the simplest methods for solving a linear system $\mathbf{M}\vec{x} = \vec{b}$: start with $\vec{x}_0 = \mathbf{0}$, then repeatedly move in the direction of the residual, i.e. $\vec{x}_{k+1} := \vec{x}_k + \eta(\vec{b} - \mathbf{M}\vec{x}_k)$, for some step size η . The preconditioned Richardson iteration refers to applying the same method, with the aid of a matrix \mathbf{Z} whose purpose is to improve the quality of the iterations by producing better approximations to the matrix inverse applied to the residual: $\vec{x}_{k+1} = \vec{x}_k + \eta\mathbf{Z}(\vec{b} - \mathbf{M}\vec{x}_k)$. Note that whenever $\mathbf{Z} = \mathbf{M}^+$, preconditioned Richardson adds in every step an η fraction of the true solution to our current iterate; therefore, in that case, we obtain the exact solution in one iteration by setting $\eta = 1$. Intuitively, the quality of the preconditioner \mathbf{Z} dictates the size of the steps we are allowed to take, and therefore how long it takes to get close to optimum. This motivates the notion of approximation we introduce in this section.

The preconditioned Richardson iteration is very well studied and fundamental to numerical methods (see e.g. Section 13.2.1 of [34]). However, we are not aware of an operator-based analysis involving asymmetric matrices in the different matrix norms required in our algorithm. Therefore, we provide the algorithm (see Figure 4.1) and its short analysis in Lemma 4.2 below.

Lemma 4.2 (Preconditioned Richardson). *Let $\vec{b} \in \mathbb{R}^n$ and $\mathbf{M}, \mathbf{Z}, \mathbf{U} \in \mathbb{R}^{n \times n}$ such that \mathbf{U} is symmetric positive semidefinite, $\ker(\mathbf{U}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M}^\top) = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^\top)$, and $b \in \text{im}(\mathbf{M})$. Then $N \geq 0$ iterations of preconditioned Richardson with step size $\eta > 0$, results in a vector $\vec{x}_N = \text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \vec{b}, \eta, N)$ such that*

$$\left\| \vec{x}_N - \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U}} \leq \left\| \mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M} \right\|_{\mathbf{U} \rightarrow \mathbf{U}}^N \left\| \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U}}.$$

Furthermore, preconditioned Richardson implements a linear operator, in the sense that $\vec{x}_N = \mathbf{Z}_N \vec{b}$, for some matrix \mathbf{Z}_N only depending on $\mathbf{Z}, \mathbf{M}, \eta$ and N .

⁸One should note that being able to boost a solver using preconditioned Richardson relies on the fact that the solver is a linear operator, which is precisely the case in our algorithm.

$\vec{x} = \text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \vec{b}, \eta, N)$

Input: $n \times n$ matrix \mathbf{M} ,

preconditioning linear operator \mathbf{Z} (in the unpreconditioned case, $\mathbf{Z} = \mathbf{I}$),
right hand side vector $\vec{b} \in \text{im}(\mathbf{M})$, step size η , iteration count N .

1. Initialize $\vec{x}_0 \leftarrow 0$.

2. For $k = 0, \dots, N - 1$

(a) $\vec{x}_{k+1} \leftarrow \vec{x}_k + \eta \mathbf{Z} (\vec{b} - \mathbf{M} \vec{x}_k)$.

3. Return \vec{x}_N .

Figure 4.1: Pseudocode for the (preconditioned) Richardson Iteration

Proof. Let $\vec{x}^* \stackrel{\text{def}}{=} \mathbf{M}^+ \vec{b}$. The iteration on Line 2a, together with the fact that \vec{b} lives inside the image of \mathbf{M} , implies

$$\begin{aligned} \vec{x}_{k+1} - \vec{x}^* &= ((\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}) \vec{x}_k + \eta \mathbf{Z} \vec{b}) - \vec{x}^* = ((\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}) \vec{x}_k + \eta \mathbf{Z} \mathbf{M} \vec{x}^*) - \vec{x}^* \\ &= (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M})(\vec{x}_k - \vec{x}^*), \end{aligned}$$

and therefore,

$$\|\vec{x}_{k+1} - \vec{x}^*\|_{\mathbf{U}} = \|(\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M})(\vec{x}_k - \vec{x}^*)\|_{\mathbf{U}} \leq \|\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} \|\vec{x}_k - \vec{x}^*\|_{\mathbf{U}}.$$

By induction, this shows that

$$\|\vec{x}_N - \vec{x}^*\|_{\mathbf{U}} \leq \|\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}}^N \|\vec{x}_0 - \vec{x}^*\|_{\mathbf{U}} = \|\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}}^N \|\vec{x}^*\|_{\mathbf{U}}.$$

Now, by writing the iteration as $\vec{x}_{k+1} = (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M}) \vec{x}_k + \eta \mathbf{Z} \vec{b}$, and expanding, we see by induction that $\vec{x}_N = \sum_{k=1}^N (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M})^{k-1} \eta \mathbf{Z} \vec{b}$, and therefore

$$\mathbf{Z}_N = \eta \sum_{k=1}^N (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{Z} \mathbf{M})^{k-1} \mathbf{Z}.$$

□

Lemma 4.2 shows that if $\eta \mathbf{Z} \mathbf{M}$ is sufficiently close to the identity, then preconditioned Richardson converges quickly when solving a linear system. This highlights a precise way to quantify how good a matrix is as a preconditioner for the Richardson iteration.

Definition 4.3 (Approximate Pseudoinverse). Matrix \mathbf{Z} is an ϵ -approximate pseudoinverse of matrix \mathbf{M} with respect to a symmetric positive semidefinite matrix \mathbf{U} , if $\ker(\mathbf{U}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M}^\top) = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^\top)$, and

$$\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z} \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} \leq \epsilon. \quad \text{⁹}$$

⁹Note that the ordering of \mathbf{Z} and \mathbf{M} is crucial: this definition is not equivalent to $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{M} \mathbf{Z}\|_{\mathbf{U} \rightarrow \mathbf{U}}$ being small.

With this definition and Lemma 4.2, we see that our problem of solving a linear system can be reduced to producing an approximate pseudoinverse that we can apply efficiently. This is the approach we take in the rest of the paper. In the remainder of this section we give two tools towards producing such pseudoinverses. We show how to use preconditioned Richardson to improve the quality of an approximate pseudoinverse (Lemma 4.4), and how to produce one for a matrix whose symmetrization is well conditioned (Lemma 4.5).

Lemma 4.4 (Pseudoinverse Improvement). *If \mathbf{Z} is an ϵ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{U} , for $\epsilon \in (0, 1)$, $\vec{b} \in \text{im}(\mathbf{M})$, and $N \geq 0$, then $\text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \vec{b}, 1, N)$ computes $\vec{x}_N = \mathbf{Z}_N \vec{b}$, for some matrix \mathbf{Z}_N only depending on \mathbf{Z} , \mathbf{M} and N , such that \mathbf{Z}_N is an ϵ^N -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{U} .*

Proof. By Lemma 4.2 we know that $\vec{x}_N = \mathbf{Z}_N \vec{b}$, for some \mathbf{Z}_N that only depends on \mathbf{Z} , \mathbf{M} , and N ; furthermore, we know that:

$$\left\| (\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N \mathbf{M}) \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U} \rightarrow \mathbf{U}} = \left\| \vec{x}_N - \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U}} \leq \left\| \mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z} \mathbf{M} \right\|_{\mathbf{U} \rightarrow \mathbf{U}}^N \left\| \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U}} \leq \epsilon^N \left\| \mathbf{M}^+ \vec{b} \right\|_{\mathbf{U}},$$

and that this holds for any vector $\vec{b} \in \text{im}(\mathbf{M})$ (since \mathbf{Z}_N does not depend on \vec{b}). Equivalently, this means that

$$\left\| \mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N \mathbf{M} \right\|_{\mathbf{U} \rightarrow \mathbf{U}} \leq \epsilon^N.$$

In order to complete the proof we need to show that $\ker(\mathbf{Z}) = \ker(\mathbf{Z}^\top) = \ker(\mathbf{Z})$.

As we saw in the proof of Lemma 4.2, over $\mathbf{I}_{\text{im}(\mathbf{M})}$, \mathbf{Z}_N is a polynomial in \mathbf{Z} and \mathbf{M} with no constant term. Also, since all the kernels and cokernels of \mathbf{Z} and \mathbf{M} are identical by definition, $\ker \mathbf{Z}_N \supseteq \ker \mathbf{Z}$, and similarly $\ker \mathbf{Z}^\top \supseteq \ker \mathbf{Z}$.

Now suppose the first inclusion is strict, i.e. there exists $\vec{x} \perp \ker(\mathbf{Z})$ such that $\mathbf{Z}_N \vec{x} = 0$. Then, $\|\mathbf{M}^+ \vec{x}\|_{\mathbf{U}} > 0$ since $\mathbf{M}^+ \vec{x} \perp \ker(\mathbf{U})$, as by definition, the kernel of \mathbf{U} is a subset of that of \mathbf{Z} . This implies that

$$\left\| (\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N \mathbf{M}) \mathbf{M}^+ \vec{x} \right\|_{\mathbf{U}} = \left\| \mathbf{M}^+ \vec{x} \right\|_{\mathbf{U}}$$

This shows that $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} \geq 1$, which contradicts the fact that it is at most ϵ^N .

Similarly, if there exists $\vec{x} \perp \ker(\mathbf{Z}^\top)$ such that $\mathbf{Z}^\top \vec{x} = 0$, then we obtain

$$\left\| (\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N^\top \mathbf{M}^\top) (\mathbf{M}^\top)^+ \vec{x} \right\|_{\mathbf{U}} = \left\| (\mathbf{M}^\top)^+ \vec{x} \right\|_{\mathbf{U}},$$

and thus $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N^\top \mathbf{M}^\top\|_{\mathbf{U} \rightarrow \mathbf{U}} \geq 1$. Equivalently, this shows that $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}_N \mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} \geq 1$, which yields a contradiction. The fact that the norm is the same when taking transposes follows from writing it in terms of the ℓ_2 norm, and using the fact that $\ker(\mathbf{U})$ is a subset of both the kernel of the matrix and that of its transpose. □

In addition, we show that preconditioned Richardson converges quickly whenever the matrix \mathbf{M} is well conditioned (as a matter of fact, for our purposes we only care about the case when the ratio between $\|\mathbf{M}\|$ and $\lambda_*(\mathbf{U}_\mathbf{M})$ is at most a constant). The lemma below gives precise bounds on the number of iterations required to obtain a good approximate pseudoinverse with respect to \mathbf{I} . Such an approximate pseudoinverse is the standard object that can be used as a preconditioner in order to obtain a small number of preconditioned Richardson iterations when measuring error with respect to the ℓ_2 norm.

Lemma 4.5 (Building a Pseudoinverse). *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ such that $\mathbf{U}_{\mathbf{M}}$ is positive semidefinite, and $\ker(\mathbf{M}) = \ker(\mathbf{M}^\top)$. Let the vector $\vec{b} \in \text{im}(\mathbf{M})$, the step size $\eta \leq \lambda_*(\mathbf{U}_{\mathbf{M}})/\|\mathbf{M}\|_2^2$, and the number of iterations N . Then $\text{PRECONRICHARDSON}(\mathbf{M}, \eta \mathbf{I}_{\text{im}(\mathbf{M})}, \vec{b}, 1, N)$ computes $\vec{x}_N = \mathbf{Z}_N \vec{b}$, for some matrix \mathbf{Z}_N only depending on \mathbf{Z}, \mathbf{M} and N , such that \mathbf{Z}_N is an $\exp(-N\eta\lambda_*(\mathbf{U}_{\mathbf{M}})/2)$ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{I} .*

Proof. We begin by showing that $\eta \mathbf{I}_{\text{im}(\mathbf{M})}$ is a $(1 - \eta\lambda_*(\mathbf{U}_{\mathbf{M}}))^{1/2}$ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{I} . First we notice that the kernel conditions for $\mathbf{I}_{\text{im}(\mathbf{M})}$ being an approximate pseudoinverse for \mathbf{M} are trivially satisfied. Second, we bound the matrix induced norm of $\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{M}$:

$$\begin{aligned} \|\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{M}\|_2^2 &= \max_{\vec{x} \in \text{im}(\mathbf{M}), \|\vec{x}\|_2=1} \vec{x}^\top (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{M})^\top (\mathbf{I}_{\text{im}(\mathbf{M})} - \eta \mathbf{M}) \vec{x} \\ &= \max_{\vec{x} \in \text{im}(\mathbf{M}), \|\vec{x}\|_2=1} \left(\vec{x}^\top \mathbf{I}_{\text{im}(\mathbf{M})} \vec{x} - \eta \vec{x}^\top (\mathbf{M} + \mathbf{M}^\top) \vec{x} + \eta^2 \vec{x}^\top \mathbf{M}^\top \mathbf{M} \vec{x} \right) \\ &\leq 1 - 2\eta \min_{\vec{x} \in \text{im}(\mathbf{M}), \|\vec{x}\|_2=1} \vec{x}^\top \mathbf{U}_{\mathbf{M}} \vec{x} + \eta^2 \max_{\vec{x} \in \text{im}(\mathbf{M}), \|\vec{x}\|_2=1} \vec{x}^\top \mathbf{M}^\top \mathbf{M} \vec{x}. \\ &\leq 1 - 2\eta\lambda_*(\mathbf{U}_{\mathbf{M}}) + \eta^2 \|\mathbf{M}\|_2^2 \leq 1 - \eta\lambda_*(\mathbf{U}_{\mathbf{M}}). \end{aligned}$$

Consequently, by Lemma 4.4 we have that \mathbf{Z}_N is a $(1 - \eta\lambda_*(\mathbf{U}_{\mathbf{M}}))^{N/2}$ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{I} . The conclusion follows by using $(1 - \eta\lambda_*(\mathbf{U}_{\mathbf{M}}))^{N/2} \leq \exp(-N\eta\lambda_*(\mathbf{U}_{\mathbf{M}})/2)$. \square

4.2 Construction of Square-Sparsification Chains

Here we define the square-sparsification chain we use in our algorithm and show how to compute such a chain efficiently. In other words, we show how to create the sequence of matrices that through careful application yield an almost linear time algorithm for solving an Eulerian Laplacian system.

Definition 4.6. [Square Sparsifier Chain] We call a sequence of matrices $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d \in \mathbb{R}^{n \times n}$ a *square-sparsifier chain of length d with parameter $0 < \alpha < \frac{1}{2}$ and error $\epsilon \leq 1/2$* (or a (d, ϵ, α) -chain for short) if under the definitions $\mathbf{L}_i = \mathbf{I} - \mathcal{A}_i$ and $\mathcal{A}_i^{(\alpha)} = \alpha \mathbf{I} + (1 - \alpha)\mathcal{A}_i$ for all i the following hold

1. $\|\mathcal{A}_i\|_2 \leq 1$ for all i ,
2. $\mathbf{I} - \mathcal{A}_i$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_{i-1}^{(\alpha)})^2$ for all $i \geq 1$,
3. $\ker(\mathbf{L}_i) = \ker(\mathbf{L}_i^\top) = \ker(\mathbf{L}_j) = \ker(\mathbf{L}_j^\top) = \ker(\mathbf{U}_{\mathbf{L}_i}) = \ker(\mathbf{U}_{\mathbf{L}_j})$ for all i, j .

Pseudocode for the construction of the square-sparsifier chain is given in Figure 4.2. In the remainder of this subsection we prove correctness of this construction. We first provide a helper lemma, Lemma 4.7 and then analyze the algorithm in Lemma 4.8. In Section 4.3 we prove additional properties regarding solver chains such as how the $\mathbf{U}_{\mathbf{L}_i}$ multiplicatively approximate each other and how the smallest eigenvalue at the end of the chain must eventually rise to at least a constant.

Lemma 4.7. *If for $\mathbf{A} \in \mathbb{R}_{\geq 0}^{n \times n}$ and $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1})$ the matrix $\mathcal{L} = \mathbf{D} - \mathbf{A}$ is an Eulerian Laplacian associated with a strongly connected graph then, $\|\mathbf{D}^{-1/2} \mathbf{A}^\top \mathbf{D}^{-1/2}\|_2 \leq 1$ and $\ker(\mathcal{L}) = \ker(\mathcal{L}^\top) = \ker(\mathbf{U}_{\mathcal{L}}) = \text{span}(\mathbf{D}^{1/2}\mathbf{1})$.*

Proof. Since \mathcal{L} is Eulerian $\mathbf{A}\mathbf{1} = \mathbf{A}^\top \mathbf{1} = \mathbf{D}\mathbf{1}$ and therefore we have $\|\mathbf{D}^{-1} \mathbf{A}^\top\|_\infty = \|\mathbf{A}^\top \mathbf{D}^{-1}\|_1 = 1$. Consequently, by Lemma B.4 we have that $\|\mathbf{D}^{-1/2} \mathbf{A}^\top \mathbf{D}^{-1/2}\|_2 \leq \sqrt{\|\mathbf{D}^{-1} \mathbf{A}^\top\|_\infty \cdot \|\mathbf{A}^\top \mathbf{D}^{-1}\|_1} \leq 1$. The characterization of the kernels follows from Lemma 2.2 and that $\mathcal{L}\mathbf{1} = \mathcal{L}^\top \mathbf{1} = \mathbf{U}_{\mathcal{L}}\mathbf{1} = 0$. \square

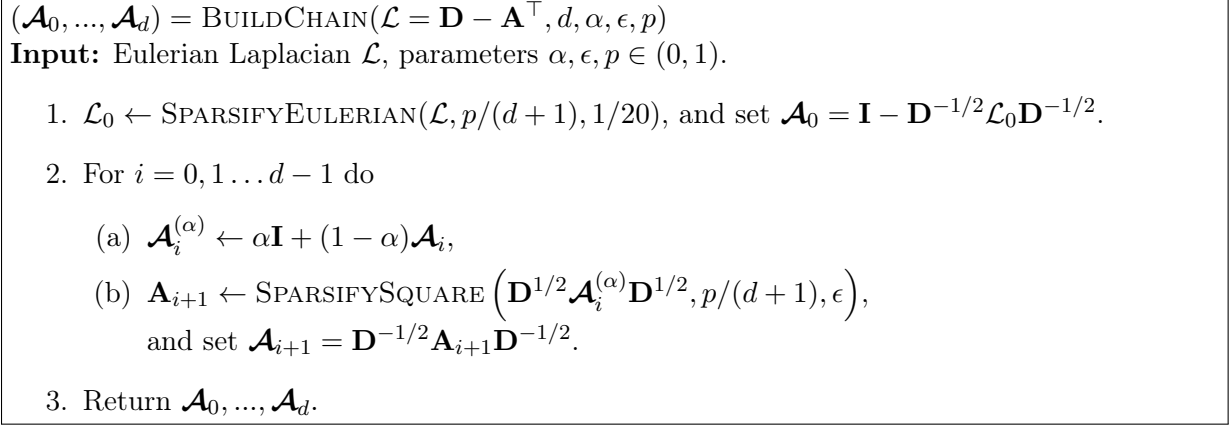


Figure 4.2: Algorithm for Constructing the Square-Sparsification Chain.

Lemma 4.8 (Chain Construction). *Let $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$ be an Eulerian Laplacian that is associated with a strongly connected graph, let $\alpha, \epsilon, p \in (0, 1)$, and let $d \geq 1$. Then in $\tilde{O}(\text{nnz}(\mathcal{L}) + n\epsilon^{-2}d)$ time the routine $\text{BUILDCHAIN}(\mathcal{L}, d, \alpha, \epsilon, p)$ produces $\mathcal{A}_0, \dots, \mathcal{A}_d \in \mathbb{R}^{n \times n}$ that with probability $1 - p$*

1. $\mathcal{A}_0, \dots, \mathcal{A}_d$ is a (d, α, ϵ) -chain,
2. $\text{nnz}(\mathcal{A}_i) = \tilde{O}(n\epsilon^{-2})$ for all i , and
3. $\mathbf{I} - \mathcal{A}_0$ is a $(1/20)$ -approximation of $\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}$.

Proof. By Theorem 3.16, the call to SPARSIFYEULERIAN computes in $\tilde{O}(m+n)$ time \mathcal{L}_0 that with probability $1 - p/(d+1)$ is a $(1/20)$ -sparsifier of \mathcal{L} with the same diagonal as \mathcal{L} . Consequently, for $\mathbf{A}_0 = \mathbf{D}^{1/2} \mathcal{L}_0 \mathbf{D}^{1/2}$ we have $\mathcal{L}_0 = \mathbf{D} - \mathbf{A}_0$. By Lemma 3.7 this implies that $\mathbf{I} - \mathcal{A}_0$ is a $(1/20)$ -approximation of $\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}$. Furthermore, Lemma 4.7 then implies that $\|\mathcal{A}_0\|_2 = \|\mathbf{D}^{-1/2} \mathbf{A}_0 \mathbf{D}^{-1/2}\|_2 \leq 1$, and $\ker(\mathbf{I} - \mathcal{A}_0) = \ker((\mathbf{I} - \mathcal{A}_0)^\top) = \ker(\mathbf{U}_{\mathbf{I} - \mathcal{A}_0}) = \text{span}(\mathbf{D}^{1/2} \mathbf{1})$. Thus, \mathcal{A}_0 has all the desired properties.

Now suppose the desired properties hold for $\mathcal{A}_0, \dots, \mathcal{A}_k$ with probability $1 - p(k+1)/(d+1)$, for some $k \in [0, d-1]$, and that for all $i \in [k]$ we have $\mathbf{A}_i \in \mathbb{R}_{\geq 0}^{n \times n}$ such that $\mathbf{D} - \mathbf{A}_i$ is an Eulerian Laplacian associated with a strongly connected graph. Under this assumption, clearly $\mathbf{D}^{-1/2} \mathcal{A}_k \mathbf{D}^{-1/2} = \alpha \mathbf{D} + (1 - \alpha) \mathbf{A}_k$ has both row and column sums equal to $\mathbf{D} \mathbf{1}$. By Theorem 3.19, the call to SPARSIFY SQUARE computes in $\tilde{O}(m + n\epsilon^{-2})$ time a matrix $\mathbf{A}_{k+1} \in \mathbb{R}_{\geq 0}^{n \times n}$ such that, with probability $1 - p(k+2)/(d+1)$, $\mathbf{D} - \mathbf{A}_{k+1}$ is an ϵ -sparsifier for $\mathbf{D} - \mathbf{A}_k^{(\alpha)} \mathbf{D}^{-1} \mathbf{A}_k^{(\alpha)}$, where $\mathbf{A}_k^{(\alpha)} = \alpha \mathbf{I} + (1 - \alpha) \mathbf{A}_k$. Again, using Lemma 3.7, we see that $\mathbf{I} - \mathcal{A}_{k+1}$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_k^{(\alpha)})^2$. Furthermore, since $\mathbf{D} - \mathbf{A}_k^{(\alpha)} \mathbf{D}^{-1} \mathbf{A}_k^{(\alpha)}$ contains $\mathbf{D} - \mathbf{A}_k$ as a subgraph, this Eulerian Laplacian is strongly connected and therefore so is $\mathbf{D} - \mathbf{A}_{k+1}$. Consequently, by Lemma 4.7 we have that $\|\mathcal{A}_{k+1}\|_2 = \|\mathbf{D}^{-1/2} \mathbf{A}_{k+1} \mathbf{D}^{-1/2}\|_2 \leq 1$, and $\ker(\mathbf{I} - \mathcal{A}_{k+1}) = \ker((\mathbf{I} - \mathcal{A}_{k+1})^\top) = \ker(\mathbf{U}_{\mathbf{I} - \mathcal{A}_{k+1}}) = \text{span}(\mathbf{D}^{1/2} \mathbf{1})$. Therefore, by induction all the desired properties hold. \square

4.3 Properties of the Square-Sparsification Chain

Here we prove several properties of the square-sparsification chain which we use in the analysis of our solver algorithm. The main result of this subsection is to prove the following:

Lemma 4.9. [Chain Properties] For length $d \geq 1$, parameter $\alpha = 1/4$, and error $\epsilon \in (0, 1/2)$, and (d, α, ϵ) -chain $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ the following properties hold:

1. $\kappa(\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}}) \leq 21$, for all $i \in [d]$, and
2. $\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}) \geq \min\{1/4, \lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_0}) \cdot ((1 - \epsilon)1.25)^d\}$.

The first property shows that the matrix $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$ changes only within a constant factor for a constant change in i , while the second implies that the smallest non-zero eigenvalue improves geometrically with squaring, up to some fixed value that depends on α .

The proof of Lemma 4.9 relies on several components. First, we use a lemma which shows how squaring changes the associated symmetric matrix (see Lemma B.7 in Appendix B). Combining with the sparsification guarantees from Lemma 3.6, we derive the first property. Second, we use a bound on how the smallest non-zero eigenvalue improves after squaring (see Lemma B.8 from Appendix B), in order to show that this is also the case with the matrices in our chain.

Proof of Lemma 4.9. By definition, $\mathbf{I} - \mathcal{A}_i$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_{i-1}^{(\alpha)})^2$, for all $i \geq 1$. Therefore by Lemma 3.6 we know that

$$(1 - \epsilon) \left(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_{i-1}^{(\alpha)})^2} \right) \preceq \mathbf{I} - \mathbf{U}_{\mathcal{A}_i} \preceq (1 + \epsilon) \left(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_{i-1}^{(\alpha)})^2} \right).$$

Applying Lemma B.7, we obtain

$$2\alpha \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}^{(\alpha)}} \right) \preceq \mathbf{I} - \mathbf{U}_{(\mathcal{A}_{i-1}^{(\alpha)})^2} \preceq (4 - 2\alpha) \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}^{(\alpha)}} \right).$$

Combining with the sparsification guarantee from above, we obtain:

$$(1 - \epsilon)2\alpha \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}^{(\alpha)}} \right) \preceq \mathbf{I} - \mathbf{U}_{\mathcal{A}_i} \preceq (1 + \epsilon)(4 - 2\alpha) \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}^{(\alpha)}} \right).$$

Finally, writing $\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}^{(\alpha)}} = \mathbf{I} - (\alpha\mathbf{I} + (1 - \alpha)\mathbf{U}_{\mathcal{A}_{i-1}}) = (1 - \alpha)\mathbf{U}_{\mathcal{A}_{i-1}}$, this gives:

$$(1 - \epsilon)2\alpha(1 - \alpha) \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}} \right) \preceq \mathbf{I} - \mathbf{U}_{\mathcal{A}_i} \preceq (1 + \epsilon)(4 - 2\alpha)(1 - \alpha) \left(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}} \right),$$

which shows that

$$\kappa(\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}}) \leq \frac{(1 + \epsilon)(4 - 2\alpha)}{(1 - \epsilon)2\alpha} \leq 21.$$

For the second part, we see that Lemma B.8 yields:

$$\lambda_*(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_{i-1}^{(\alpha)})^2}) \geq \min\{\alpha, (1 + \alpha)\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}})\}.$$

Combining with the first inequality from the sparsification guarantee, this gives

$$\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}) \geq (1 - \epsilon) \cdot \min\{\alpha, (1 + \alpha)\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i-1}})\}.$$

Applying this inequality d times yields the second part of the result. □

4.4 Pseudoinverse Properties

Here we show that ϵ -approximations in the square sparsifier chain imply useful properties for building approximate pseudoinverses. We provide key lemmas to show that an approximate pseudoinverse for $\mathbf{I} - \mathcal{A}_j$ can be transformed into one for $\mathbf{I} - \mathcal{A}_i$, where $i < j$. More precisely, this is based on Equation 2.3, and can be seen by substituting the condition from Definition 4.6 that $\mathbf{I} - \mathcal{A}_j$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_{j-1}^{(\alpha)})^2$ into the identity

$$(\mathbf{I} - \mathcal{A}_{j-1})^+ = (1 - \alpha) \left(\mathbf{I} - \mathcal{A}_{j-1}^{(\alpha)} \right)^+ = (1 - \alpha) \left(\mathbf{I} - (\mathcal{A}_{j-1}^{(\alpha)})^2 \right)^+ \left(\mathbf{I} + \mathcal{A}_{j-1}^{(\alpha)} \right).$$

This method of producing solvers accumulates error very quickly. However, as discussed in Section 4.1, we can reduce the error accumulation using preconditioned Richardson iterations. Consequently, the main goal of the remainder of this section is to formally bound this accumulated error so that we can ensure Richardson will quickly produce a high quality approximate pseudoinverse.

We prove this result in several steps. First we provide several properties regarding approximate pseudoinverses, showing that they are well behaved under composition, and that they exhibit desirable properties such as approximate triangle inequality, and being preserved under right multiplication. Then using these lemmas we prove the main result of this section, Lemma 4.13.

Given that ultimately we build our pseudoinverse recursively, in our presentation \mathbf{Z} will often take the role of a solver being used as a preconditioner. This is consistent with the way we use it, since all solvers we produce are linear operators.

The following lemma bounds the quality of a preconditioner obtained via composition.

Lemma 4.10 (Triangle Inequality of Approximate Pseudoinverses). *If matrix \mathbf{Z} is an ϵ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{U} , and $\widetilde{\mathbf{M}}^+$ is an ϵ' -approximate pseudoinverse of \mathbf{Z}^+ with respect to \mathbf{U} , and has the same left and right kernels as \mathbf{M} and \mathbf{Z} , then $\widetilde{\mathbf{M}}^+$ is an $(\epsilon + \epsilon' + \epsilon\epsilon')$ -approximate pseudoinverse of \mathbf{M} with respect to \mathbf{U} .*

Proof. Applying triangle inequality for the $\mathbf{U} \rightarrow \mathbf{U}$ norm, we see that, for all $x \in \mathbb{R}^n$,

$$\left\| \left(\mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}} \leq \left\| \left(\mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{Z}^+ \right) \vec{x} \right\|_{\mathbf{U}} + \left\| \left(\widetilde{\mathbf{M}}^+ \mathbf{Z}^+ - \widetilde{\mathbf{M}}^+ \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}}.$$

The first term is upper bounded by $\epsilon' \|\vec{x}\|_{\mathbf{U}}$ by the condition given in the statement. The second term can be rewritten as:

$$\left\| \left(\widetilde{\mathbf{M}}^+ \mathbf{Z}^+ - \widetilde{\mathbf{M}}^+ \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}} = \left\| \widetilde{\mathbf{M}}^+ \mathbf{Z}^+ \left(\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z} \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}} \leq \left\| \widetilde{\mathbf{M}}^+ \mathbf{Z}^+ \right\|_{\mathbf{U} \rightarrow \mathbf{U}} \cdot \left\| \left(\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z} \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}}.$$

Next, we bound the two components of the product. For the first one, using triangle inequality along with the condition given in the statement, we obtain

$$\left\| \widetilde{\mathbf{M}}^+ \mathbf{Z}^+ \right\|_{\mathbf{U} \rightarrow \mathbf{U}} \leq \left\| \mathbf{I}_{im(\mathbf{M})} \right\|_{\mathbf{U} \rightarrow \mathbf{U}} + \left\| \mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{Z}^+ \right\|_{\mathbf{U} \rightarrow \mathbf{U}} \leq 1 + \epsilon'.$$

The second term is by definition bounded by $\epsilon \|\vec{x}\|_{\mathbf{U}}$. Combining these bounds, we obtain

$$\left\| \left(\mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{M} \right) \vec{x} \right\|_{\mathbf{U}} \leq \epsilon' \|\vec{x}\|_{\mathbf{U}} + (1 + \epsilon') \epsilon \|\vec{x}\|_{\mathbf{U}} = (\epsilon + \epsilon' + \epsilon\epsilon') \|\vec{x}\|_{\mathbf{U}}.$$

□

The following lemma is used to show that ϵ -approximations obtained via the sparsification routines from Section 3 also yield matrices whose pseudoinverses are good preconditioners for the original.

Lemma 4.11. *Let \mathbf{M} be any matrix with $\mathbf{U}_{\mathbf{M}}$ positive semidefinite, such that $\ker(\mathbf{M}) = \ker(\mathbf{M}^\top) = \ker(\mathbf{U}_{\mathbf{M}})$. Suppose that matrix $\widetilde{\mathbf{M}}$ ϵ -approximates \mathbf{M} , for some $\epsilon \leq 1/2$. Then $\widetilde{\mathbf{M}}^+$ is an 2ϵ -approximate pseudoinverse for \mathbf{M} with respect to $\mathbf{U}_{\mathbf{M}}$. Furthermore, $\ker(\mathbf{U}_{\mathbf{M}}) = \ker(\mathbf{U}_{\widetilde{\mathbf{M}}})$.*

Proof. First, we prove that $\ker(\widetilde{\mathbf{M}}) = \ker(\widetilde{\mathbf{M}}^\top) = \ker(\mathbf{U}_{\widetilde{\mathbf{M}}})$ and that these kernels are the same as those of \mathbf{M} , \mathbf{M}^\top and $\mathbf{U}_{\mathbf{M}}$.

We already know that $\ker(\mathbf{M}) = \ker(\mathbf{M}^\top) = \ker(\mathbf{U}_{\mathbf{M}})$ and it is not hard to prove that $\ker(\widetilde{\mathbf{M}}), \ker(\widetilde{\mathbf{M}}^\top) \subseteq \ker(\mathbf{U}_{\widetilde{\mathbf{M}}})$. Thus, it suffices to prove that $\ker(\mathbf{U}_{\mathbf{M}}) \subseteq \ker(\widetilde{\mathbf{M}}), \ker(\widetilde{\mathbf{M}}^\top)$ and $\ker(\mathbf{U}_{\widetilde{\mathbf{M}}}) \subseteq \ker(\mathbf{U}_{\mathbf{M}})$.

To prove $\ker(\mathbf{U}_{\mathbf{M}}) \subseteq \ker(\widetilde{\mathbf{M}}), \ker(\widetilde{\mathbf{M}}^\top)$, consider any vector x in the kernel of \mathbf{M} . Then x is also in the kernel of $\mathbf{U}_{\mathbf{M}}$. However, by the definition of strong approximation, this means that x is in the kernel of $\mathbf{M} - \widetilde{\mathbf{M}}$ and $\mathbf{M}^\top - \widetilde{\mathbf{M}}^\top$. Since x is in the left and right kernels of \mathbf{M} , we have $0 = (\mathbf{M} - \widetilde{\mathbf{M}})x = \widetilde{\mathbf{M}}x$ and similarly obtain $0 = (\mathbf{M}^\top - \widetilde{\mathbf{M}}^\top)x = \widetilde{\mathbf{M}}x$. Thus, the left and right kernels of $\widetilde{\mathbf{M}}$ are supersets of $\ker(\mathbf{M})$.

For the reverse direction, note that Lemma 3.6 implies that $\mathbf{U}_{\mathbf{M}}$ and $\mathbf{U}_{\widetilde{\mathbf{M}}}$ approximate each other in the standard positive semidefinite sense for undirected matrices, and thus have the same kernel. Thus, the kernels of \mathbf{M} meet the requirements for being approximate pseudoinverses.

Now we can show the requisite inequality for $\widetilde{\mathbf{M}}^+$ being an approximate pseudoinverse of \mathbf{M} with respect to $\mathbf{U}_{\mathbf{M}}$. First we show that \mathbf{M}^+ is an ϵ -approximate pseudoinverse of $\widetilde{\mathbf{M}}$ with respect to $\mathbf{U}_{\mathbf{M}}$. We can apply a lemma upper bounding matrix norms whose proof can be found in the appendix (Lemma B.9) in order to obtain:

$$\left\| \mathbf{I}_{im(\mathbf{M})} - \mathbf{M}^+ \widetilde{\mathbf{M}} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \leq \left\| \mathbf{U}_{\mathbf{M}}^{+/2} \mathbf{M} \left(\mathbf{I}_{im(\mathbf{M})} - \mathbf{M}^+ \widetilde{\mathbf{M}} \right) \mathbf{U}_{\mathbf{M}}^{+/2} \right\|_2 = \left\| \mathbf{U}_{\mathbf{M}}^{+/2} \left(\mathbf{M} - \widetilde{\mathbf{M}} \right) \mathbf{U}_{\mathbf{M}}^{+/2} \right\|_2 \leq \epsilon.$$

Next we prove that this implies the desired conclusion by writing

$$\begin{aligned} \left\| \mathbf{I}_{im(\mathbf{M})} - \mathbf{M}^+ \widetilde{\mathbf{M}} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} &= \left\| (\widetilde{\mathbf{M}}^+ \mathbf{M} - \mathbf{I}_{im(\mathbf{M})}) \mathbf{M}^+ \widetilde{\mathbf{M}} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \\ &\geq \left\| \widetilde{\mathbf{M}}^+ \mathbf{M} - \mathbf{I}_{im(\mathbf{M})} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} - \left\| (\widetilde{\mathbf{M}}^+ \mathbf{M} - \mathbf{I}_{im(\mathbf{M})}) (\mathbf{M}^+ \widetilde{\mathbf{M}} - \mathbf{I}_{im(\mathbf{M})}) \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \\ &\geq \left\| \mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{M} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \left(1 - \left\| \mathbf{I}_{im(\mathbf{M})} - \mathbf{M}^+ \widetilde{\mathbf{M}} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \right). \end{aligned}$$

For the first inequality we used triangle inequality, for the second one we used the fact that the norm of a product is upper bounded by the product of norms, which follows immediately from applying the definition of our matrix norm. By rearranging terms, and using the fact that \mathbf{M}^+ is an ϵ -approximate pseudoinverse of $\widetilde{\mathbf{M}}$ with respect to $\mathbf{U}_{\mathbf{M}}$ we obtain:

$$\left\| \mathbf{I}_{im(\mathbf{M})} - \widetilde{\mathbf{M}}^+ \mathbf{M} \right\|_{\mathbf{U}_{\mathbf{M}} \rightarrow \mathbf{U}_{\mathbf{M}}} \leq \frac{\epsilon}{1 - \epsilon} \leq 2\epsilon.$$

□

Next, recall that the goal of the square-sparsifier chain is to allow an approximate inverse for $\mathbf{I} - \mathcal{A}_j$ to be used as preconditioner for $\mathbf{I} - \mathcal{A}_i$, with $i < j$. We show that our notion of approximate pseudoinverse is (approximately) preserved under right-multiplications, and when changing the reference matrix \mathbf{U} .

Lemma 4.12 (Composition of Approximate Pseudoinverses). *Let $\mathbf{Z}, \mathbf{M}, \mathbf{U} \in \mathbb{R}^{n \times n}$ be matrices such that \mathbf{U} is symmetric positive semidefinite, and $\ker(\mathbf{Z}) = \ker(\mathbf{Z}^\top) = \ker(\mathbf{M}) = \ker(\mathbf{M}^\top) \supseteq \ker(\mathbf{U})$. Then the following hold.*

1. (Preserved under right multiplication) *Let $\mathbf{C} \in \mathbb{R}^{n \times n}$ such that both \mathbf{C} and \mathbf{C}^\top are invariant on $\ker(\mathbf{M})$, in the sense that $x \in \ker(\mathbf{M})$ if and only if $\mathbf{C}x \in \ker(\mathbf{M})$, and similarly for \mathbf{C}^\top . Then \mathbf{Z} is an ϵ -approximate pseudoinverse for $\mathbf{C}\mathbf{M}$ with respect to \mathbf{U} if and only if $\mathbf{Z}\mathbf{C}$ is an ϵ -approximate pseudoinverse for \mathbf{M} with respect to \mathbf{U} .*
2. (Approximately preserved under norm change) *If \mathbf{Z} is an ϵ -approximate pseudoinverse for \mathbf{M} with respect to \mathbf{U} , then for any symmetric positive semidefinite matrix $\tilde{\mathbf{U}}$, such that $\ker(\tilde{\mathbf{U}}) = \ker(\mathbf{U})$, \mathbf{Z} is an $(\epsilon \cdot \sqrt{\kappa(\tilde{\mathbf{U}}, \mathbf{U})})$ -approximate pseudoinverse of \mathbf{M} with respect to $\tilde{\mathbf{U}}$.*

Proof. For preservation under right multiplication (claim 1), we immediately see that by associativity:

$$\|\mathbf{I}_{im(\mathbf{M})} - (\mathbf{Z}\mathbf{C})\mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} = \|\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}(\mathbf{C}\mathbf{M})\|_{\mathbf{U} \rightarrow \mathbf{U}}.$$

What we have left is to verify that kernel conditions are satisfied. The assumptions on \mathbf{C} , together with the fact that all the left and right kernels of \mathbf{Z} and \mathbf{M} coincide, ensure that $\ker(\mathbf{Z}\mathbf{C}) = \ker(\mathbf{M})$, $\ker(\mathbf{C}^\top \mathbf{Z}^\top) = \ker(\mathbf{Z}^\top)$, $\ker(\mathbf{C}\mathbf{M}) = \ker(\mathbf{M})$, $\ker(\mathbf{M}^\top \mathbf{C}^\top) = \ker(\mathbf{M}^\top)$. Therefore the matrices satisfy the kernel requirements for being approximate pseudoinverses.

For approximate preservation under change of norms (claim 2), the bound on $\kappa(\tilde{\mathbf{U}}, \mathbf{U})$ means that there exist α and β such that $\alpha\mathbf{U} \preceq \tilde{\mathbf{U}} \preceq \beta\mathbf{U}$ and $\beta/\alpha \leq \kappa(\tilde{\mathbf{U}}, \mathbf{U})$. Using this, we obtain:

$$\begin{aligned} \|\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}\mathbf{M}\|_{\tilde{\mathbf{U}} \rightarrow \tilde{\mathbf{U}}} &= \max_{\tilde{x}: \tilde{\mathbf{U}}\tilde{x} \neq 0} \frac{\|(\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}\mathbf{M})\tilde{x}\|_{\tilde{\mathbf{U}}}}{\|\tilde{x}\|_{\tilde{\mathbf{U}}}} \leq \max_{\tilde{x}: \mathbf{U}\tilde{x} \neq 0} \frac{\beta \|(\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}\mathbf{M})\tilde{x}\|_{\mathbf{U}}}{\alpha \|\tilde{x}\|_{\mathbf{U}}} \\ &\leq \sqrt{\kappa(\tilde{\mathbf{U}}, \mathbf{U})} \cdot \|\mathbf{I}_{im(\mathbf{M})} - \mathbf{Z}\mathbf{M}\|_{\mathbf{U} \rightarrow \mathbf{U}} \leq \epsilon \cdot \sqrt{\kappa(\tilde{\mathbf{U}}, \mathbf{U})}. \end{aligned}$$

□

The preservation under right-multiplications combined with Equation 2.3 suggests that if we have a linear operator \mathbf{Z} that is an approximate pseudoinverse for $\mathbf{I} - \mathcal{A}_j$, we can right-multiply it by $(1 - \alpha)(\mathbf{I} + \mathcal{A}_{j-1}^{(\alpha)})$ to form an operator that is an approximate pseudoinverse for $\mathbf{I} - \mathcal{A}_{j-1}$. This process can then be repeated down the chain, but will lead to an accumulation of error. The following lemma bounds the amount of error accumulated after repeating this process Δ times.

Lemma 4.13. *Let the sequence $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ be a (d, ϵ, α) -chain as specified in Definition 4.6, with $\epsilon \leq 1/2$ and $\alpha = 1/4$. Using the notation from Definition 4.6, consider the matrix*

$$\bar{\mathbf{Z}}_{i, i+\Delta} = (1 - \alpha)^\Delta (\mathbf{I} - \mathcal{A}_{i+\Delta})^+ \left(\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(\alpha)} \right) \cdots \left(\mathbf{I} + \mathcal{A}_i^{(\alpha)} \right),$$

for any $i, \Delta \geq 0$. Then $\bar{\mathbf{Z}}_{i, i+\Delta}$ is an $(\exp(5\Delta) \cdot \epsilon)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

Proof. We will prove this by induction on Δ . The base case $\Delta = 0$ follows immediately, since $\bar{\mathbf{Z}}_{i, i} = (\mathbf{I} - \mathcal{A}_i)^+$, which is a 0-approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

For the induction step, let us assume that the claim is true for $\Delta - 1$. Therefore, the matrix

$$\bar{\mathbf{Z}}_{i+1, i+\Delta} = (1 - \alpha)^{\Delta-1} (\mathbf{I} - \mathcal{A}_{i+\Delta})^+ \left(\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(\alpha)} \right) \cdots \left(\mathbf{I} + \mathcal{A}_{i+1}^{(\alpha)} \right)$$

is an $(\exp(5(\Delta - 1)) \cdot \epsilon)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_{i+1}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i+1}}$.

From Lemma 4.9 we see that for our choice of $\alpha = 1/4$, we have $\kappa(\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i+1}}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_i}) \leq 21$. Since the matrices $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$ and $\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i+1}}$ have the same kernel, we can use the bound on their relative condition number with Lemma 1, part 2 to obtain that $\bar{\mathbf{Z}}_{i+1, i+\Delta}$ is also a $(\sqrt{21} \exp(5(\Delta - 1)) \cdot \epsilon)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_{i+1}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

By definition, we have that $\mathbf{I} - \mathcal{A}_{i+1}$ is an ϵ -approximation of $\mathbf{I} - (\mathcal{A}_i^{(\alpha)})^2$. Therefore, by Lemma 4.11, we know that $(\mathbf{I} - \mathcal{A}_{i+1})^+$ is a 2ϵ -approximate pseudoinverse of $\mathbf{I} - (\mathcal{A}_i^{(\alpha)})^2$ with respect to $\mathbf{I} - \mathbf{U}_{(\mathcal{A}_i^{(\alpha)})^2}$. In order to change norms, we use Lemma B.7, which gives us that

$$\kappa \left(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_i^{(\alpha)})^2}, (1 - \alpha)(\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}) \right) = \kappa \left(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_i^{(\alpha)})^2}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_i} \right) \leq \frac{4 - 2\alpha}{2\alpha},$$

and therefore

$$\kappa \left(\mathbf{I} - \mathbf{U}_{(\mathcal{A}_i^{(\alpha)})^2}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_i} \right) \leq \frac{4 - 2\alpha}{2\alpha(1 - \alpha)} \leq \frac{28}{3}.$$

Therefore, using Lemma 2, and since $\sqrt{28/3} \cdot 2 \leq 7$, we obtain that $(\mathbf{I} - \mathcal{A}_{i+1})^+$ is a 7ϵ -approximate pseudoinverse of $\mathbf{I} - (\mathcal{A}_i^{(\alpha)})^2$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$. Combining these two results via the triangle inequality for approximate pseudoinverses (Lemma 4.10), we obtain that $\bar{\mathbf{Z}}_{i+1, i+\Delta}$ is an ϵ' -approximate pseudoinverse of $\mathbf{I} - (\mathcal{A}_i^{(\alpha)})^2$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$, where

$$\epsilon' = \sqrt{21} \exp(5(\Delta - 1))\epsilon + 7\epsilon + \sqrt{21} \exp(5(\Delta - 1))\epsilon \cdot 7\epsilon \leq 50 \exp(5(\Delta - 1))\epsilon.$$

Equivalently, by writing $\mathbf{I} - (\mathcal{A}_i^{(\alpha)})^2 = (\mathbf{I} + \mathcal{A}_i^{(\alpha)})(\mathbf{I} - \mathcal{A}_i^{(\alpha)}) = (1 - \alpha)(\mathbf{I} + \mathcal{A}_i^{(\alpha)}) \cdot (\mathbf{I} - \mathcal{A}_i)$, and applying the composition under multiplication property from Lemma 4.12 Part 1, we obtain that $\bar{\mathbf{Z}}_{i+1, i+\Delta} \cdot (1 - \alpha)(\mathbf{I} + \mathcal{A}_i^{(\alpha)})$ is an ϵ' -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$. Note that in order to correctly apply the lemma, we require the kernel requirement for $(1 - \alpha)(\mathbf{I} + \mathcal{A}_i^{(\alpha)})$ to be satisfied, but this follows easily since all left and right kernels of the other matrices involved are identical to $\ker(\mathbf{I} - \mathcal{A}_i^{(\alpha)})$.

Finally, since $\bar{\mathbf{Z}}_{i, i+\Delta} = \bar{\mathbf{Z}}_{i+1, i+\Delta} \cdot (1 - \alpha)(\mathbf{I} + \mathcal{A}_i^{(\alpha)})$, this is equivalent to saying that $\bar{\mathbf{Z}}_{i, i+\Delta}$ is an approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$, with error bounded by

$$50 \exp(5(\Delta - 1))\epsilon \leq \exp(5\Delta)\epsilon.$$

□

Note that the amount of error accumulated through this process is significantly greater than the sum of ϵ 's across the different levels of the chain. This is because we are measuring the quality of the approximate inverse with respect to a matrix that may change by a constant factor at each level, rather than with respect to a fixed one.

If we only invoke Lemma 4.13 for the first and last matrices of the chain ($i = 0, j = d$), it would give an error of $\exp(O(d))\epsilon = \text{poly}(\kappa(\mathcal{L}))\epsilon$, necessitating a sparsifier accuracy that is more or less keeping everything dense. Instead, in our algorithms we will only invoke the above result for $j \approx i + \sqrt{d}$. Between such steps, we will remove the accumulated error using the preconditioned Richardson iteration, which was described in Section 4.1.

4.5 The Recursive Solver

Here we combine the pseudoinverse properties of the solver chain proved in Lemma 4.13 with the preconditioned Richardson iteration from Lemma 4.4 to obtain an almost-linear time solver for Eulerian Laplacians. The resulting algorithm makes recursive calls on the square-sparsifier chain. These recursive calls can be viewed as phases. For some moderate value of Δ which we set to $\sqrt{\log \kappa}$, where κ is the condition number of $\mathbf{U}_{\mathbf{D}-1/2} \mathcal{L}_{\mathbf{D}-1/2}$, we utilize Lemma 4.13 to turn an approximate pseudoinverse of $\mathbf{L}_{i+\Delta}$ into an approximate pseudoinverse for \mathbf{L}_i with larger error. The error accumulated in this process is then removed via preconditioned Richardson iteration. This iteration leads to recursive calls to $\mathbf{L}_{i+\Delta}$.

The resulting algorithm is a linear operator: its output can be viewed as multiplying the input by a fixed matrix. To analyze it, it is helpful to define the notion of implicit matrices, which is a more succinct way of writing “linear operator”-style solver statements like those in [37] as well as subsequent works. An *implicit matrix* is a routine that applies a linear operator to a vector. Its *complexity* is defined as the time it takes to run it when given a vector. Note that if we have a matrix explicitly given, we can view it as an implicit matrix with complexity equal to one plus its number of nonzero entries.

If \mathbf{A} is an implicit matrix, then we will use the notation $\mathbf{A}\vec{x}$ to denote $\mathbf{A}(\vec{x})$. In particular, this notation choice means we can write $\mathbf{A}(\mathbf{B}(\cdot))$ as $\mathbf{A}\mathbf{B}$ and $\mathbf{A}(\cdot) + \mathbf{B}(\cdot)$ as $\mathbf{A} + \mathbf{B}$. If we form a new implicit matrix from two (or more) explicit matrices in either of these manners, the complexity of the new explicit matrix is equal to the sum of the complexities of the matrices it was formed from.

Because an implicit matrix implements a linear operator, we are—for the purposes of analysis—free to treat it as if it is an actual matrix and talk about things like its eigenvalues or whether it approximates something—provided we do so with the understanding that whenever we say such things, we are really talking about the linear operator that the implicit matrix implements. When possible, we will use \mathbf{Z} to denote implicit matrices that represent inverses of matrices, and \mathbf{M} to represent matrices related to linear systems that we are trying to solve.

In particular, preconditioned Richardson iteration from Lemma 4.4 can be viewed as an implicit matrix $\text{PRECONRICHARDSON}(\mathbf{I} - \mathcal{A}_i, \mathbf{M}, \frac{1}{2}, O(1/\Delta))$ built from $\mathbf{I} - \mathcal{A}_i$ and \mathbf{M} , which might themselves be implicit matrices. With this in mind we can state the function that does most of the work in our algorithm in Figure 4.3.

We now show that given a square-sparsifier chain and access to an implicit matrix that is an approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_d$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}$, we can efficiently compute an implicit matrix \mathbf{Z}_0 which is an approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_0$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_0}$.

This done by invoking the transformations of approximate pseudoinverses in Lemma 4.13, but also swapping out the exact $(\mathbf{I} - \mathcal{A}_i)^+$ with an operator which is an approximate pseudoinverse for it.

We first provide a helper lemma, which shows that error does not increase between recursive calls to the SOLVE routine.

Lemma 4.14. *Let $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ be a $(d, \hat{\epsilon}, 1/4)$ -chain. Let $0 \leq i < d$, and let $\Delta = \min\{\sqrt{d \log d}, d - i\}$. Suppose that $\hat{\epsilon} \leq \exp(-5\Delta)/30$, and that for any $\epsilon_\Delta \leq \exp(-5\Delta)/30$, calling the routine $\text{SOLVE}((\mathcal{A}_{i+\Delta}, \dots, \mathcal{A}_d), \hat{\lambda}, \epsilon_\Delta)$ returns an implicit matrix $\mathbf{Z}_{i+\Delta}$ which is an ϵ_Δ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_{i+\Delta}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_{i+\Delta}}$. Then, calling the routine $\text{SOLVE}((\mathcal{A}_i, \dots, \mathcal{A}_d), \hat{\lambda}, \epsilon)$ returns an implicit matrix \mathbf{Z}_i which is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.*

Proof. First we notice that by Lemma 4.9 part 1, $\kappa(\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}, \mathbf{I} - \mathbf{U}_{\mathcal{A}_{i+\Delta}}) \leq 21^\Delta$. Therefore, by the norm change property from Lemma 4.12 part 2, and our hypothesis, we obtain that $\mathbf{Z}_{i+\Delta}$ is an

SOLVE($(\mathcal{A}_i \dots \mathcal{A}_d), \widehat{\lambda}, \epsilon$)

Input: Matrices $\mathcal{A}_i \dots \mathcal{A}_d$ forming a subsequence of a $(d, \widehat{\epsilon}, 1/4)$ -chain corresponding to an Eulerian Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$, a lower bound $\widehat{\lambda}$ on $\lambda_*(\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2})$, accuracy ϵ .

Output: Implicit matrix that is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

1. If $i = d$,
 - (a) $\ell \leftarrow \min\{1/4, 1.125^d \cdot 0.9 \cdot \widehat{\lambda}\}$.
 - (b) Return PRECONRICHARDSON($\mathbf{I} - \mathcal{A}_d, \frac{\ell}{4} \mathbf{I}_{im(\mathbf{I} - \mathcal{A}_d)}, 1, \frac{8}{\ell^2} \log(1/\epsilon)$).
2. $\Delta \leftarrow \min\{\sqrt{d \log d}, d - i\}$.
3. $\widetilde{\mathbf{Z}}_i \leftarrow (1 - \alpha)^\Delta \cdot \text{SOLVE}(\mathcal{A}_{i+\Delta} \dots \mathcal{A}_d, \widehat{\lambda}, \exp(-5\Delta)/30) \cdot (\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(1/4)}) \dots (\mathbf{I} + \mathcal{A}_i^{(1/4)})$.
4. Return PRECONRICHARDSON($\mathbf{I} - \mathcal{A}_i, \widetilde{\mathbf{Z}}_i, 1, \log(1/\epsilon)$).

Figure 4.3: Algorithm that produces a matrix polynomial that produces an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$, using the global solver chain constructed via BUILDCHAIN.

$(\sqrt{21^\Delta} \cdot \epsilon_\Delta)$ - and therefore also a $(1/30)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_{i+\Delta}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

On the other hand, the error propagation down the chain, bounded in Lemma 4.13 gives that the matrix

$$\overline{\mathbf{Z}}_i = (1 - 1/4)^\Delta (\mathbf{I} - \mathcal{A}_{i+\Delta})^+ \left(\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(1/4)} \right) \dots \left(\mathbf{I} + \mathcal{A}_i^{(1/4)} \right)$$

is an $(\exp(5\Delta)\widehat{\epsilon})$ - and by the bound on $\widehat{\epsilon}$ also a $(1/30)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

Using these two facts, we can now show that the implicit matrix

$$\widetilde{\mathbf{Z}}_i = (1 - 1/4)^\Delta \mathbf{Z}_{i+\Delta} \left(\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(1/4)} \right) \dots \left(\mathbf{I} + \mathcal{A}_i^{(1/4)} \right)$$

is an $1/2$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

This can be easily seen by applying the properties of approximate pseudoinverses we proved in Section 4.4. Letting $\mathbf{M} = (1 - 1/4)^\Delta (\mathbf{I} + \mathcal{A}_{i+\Delta-1}^{(1/4)}) \dots (\mathbf{I} + \mathcal{A}_i^{(1/4)})$, and applying Lemma 1 part 2 we obtain that $(\mathbf{I} - \mathcal{A}_{i+\Delta})^+$ is a $(1/30)$ -approximate pseudoinverse of $\mathbf{M}(\mathbf{I} - \mathcal{A}_i)$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$. Applying the triangle inequality from Lemma 4.10 we then obtain that $\mathbf{Z}_{i+\Delta}$ is a $(1/10)$ -approximate pseudoinverse of $\mathbf{M}(\mathbf{I} - \mathcal{A}_i)$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$. Finally, applying Lemma 1 part 2 again, we obtain that $\widetilde{\mathbf{Z}}_i = \mathbf{Z}_{i+\Delta} \mathbf{M}$ is a $(1/10)$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$.

Finally, the guarantee on the output matrix $\mathbf{Z}_i = \text{PRECONRICHARDSON}(\mathbf{L}_i, \widetilde{\mathbf{Z}}_i, 1, \log(1/\epsilon))$ then follows from Lemma 4.4. \square

Given this, we can now analyze the quality of the implicit matrix produced by calling SOLVE on the entire square sparsification chain.

Lemma 4.15. *Given a $(d, \widehat{\epsilon}, 1/4)$ -square-sparsifier chain $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ constructed for a Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top$, with $\widehat{\epsilon} \leq \exp(-5\Delta)/30$, calling the routine SOLVE($(\mathcal{A}_0 \dots \mathcal{A}_d), \widehat{\lambda}, \epsilon$), where $\epsilon \leq$*

$\exp(-5\Delta)/30$, returns an implicit matrix \mathbf{Z} which is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_0$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_0}$.

Proof. The proof relies on Lemma 4.14, and follows from induction on the depth of the call to the SOLVE routine.

The base case is $i = d$, for which we prove that the operator

$$\mathbf{Z}_d = \text{PRECONRICHARDSON} \left(\mathbf{I} - \mathcal{A}_d, \frac{\ell}{4} \mathbf{I}_{\text{im}(\mathbf{I} - \mathcal{A}_d)}, 1, \frac{8}{\ell^2} \log(1/\epsilon) \right)$$

is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_d$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}$. In order to do so, we show that the input parameters fulfill the requirements for applying Lemma 4.5.

First we notice that by Lemma 4.9, we have $\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}) \geq \min\{1/4, \lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_0}) \cdot 1.125^d\}$. Also, from Lemma 4.8, we know that $\mathbf{I} - \mathcal{A}_0$ is a $(1/10)$ -approximation of $\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}$. Therefore by Lemma 3.6 we know that $\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_0}) \geq 9/10 \cdot \lambda_*(\mathbf{U}_{\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}}) \geq 9/10 \cdot \widehat{\lambda}$. Hence $\lambda_*(\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}) \geq \min\{1/4, 1.125^d \cdot 0.9 \cdot \widehat{\lambda}\} = \ell$.

By Lemma 4.7 we have $\|\mathbf{I} - \mathcal{A}_d\|_2 \leq 2$, and therefore, applying Lemma 4.5, we obtain that $\text{PRECONRICHARDSON}(\mathbf{I} - \mathcal{A}_d, \frac{\ell}{4} \cdot \mathbf{I}_{\text{im}(\mathbf{I} - \mathcal{A}_d)}, 1, \frac{8}{\ell^2} \log(1/\epsilon))$ returns an implicit matrix \mathbf{Z} that is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_d$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_d}$.

Now for the induction step, suppose that the induction hypothesis holds for $i + \Delta$. Then, by Lemma 4.14, the matrix produced when calling the chain starting at i , \mathbf{Z}_i , is an ϵ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_i$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_i}$. Therefore, this property also holds for the matrix at the bottom of the call stack, which is what we wanted to prove. \square

Having seen that the SOLVE routine controls the accumulation of error, as it produces an approximate pseudoinverse for the first matrix in the square sparsification chain, we can now use its output as a preconditioner for the Richardson iteration, which yields our final algorithm, described in Figure 4.4.

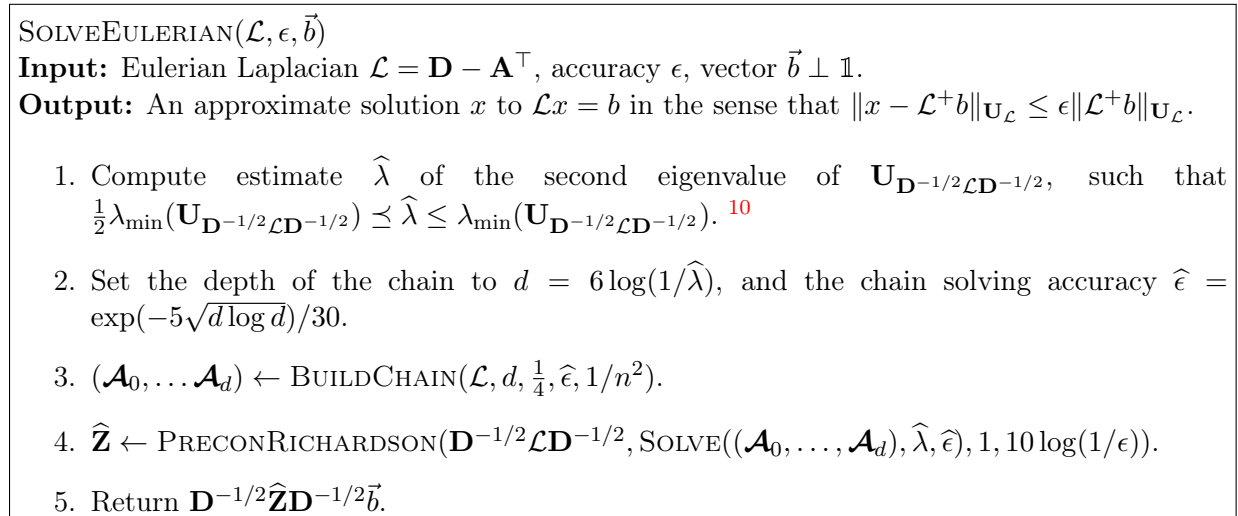


Figure 4.4: Full algorithm for solving Eulerian Laplacian systems.

¹⁰Since we know the nullspace of $\mathbf{U}_{\mathcal{L}}$, its minimum non-zero eigenvalue can be estimated in linear time to high accuracy with high probability via inverse powering. See e.g. Section 7 of [37] or Chapter 8 of [41].

What we have left is to bound the running time of our solver. We do so by analyzing the recursion tree, as well as the outermost call to the preconditioned Richardson iteration. The final bound is provided in the theorem below.

Theorem 4.16 (Eulerian Solver Guarantee). *Given an Eulerian Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{A}^\top \in \mathbb{R}^{n \times n}$ with m nonzero entries, and given an error parameter $0 < \epsilon \leq 1/2$, the algorithm $\text{SOLVEEULERIAN}(\mathcal{L}, \epsilon, \vec{b})$ returns, with probability at least $1 - 1/n$, an approximate solution \vec{x} to $\mathcal{L}\vec{x} = \vec{b}$ in the sense that*

$$\left\| \vec{x} - \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}} \leq \epsilon \left\| \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}}.$$

Furthermore, the total running time is

$$\tilde{O} \left(\left(m + ne^{O(\sqrt{\log \kappa \cdot \log \log \kappa})} \right) \log(1/\epsilon) \right),$$

where κ is the condition number of the normalized Laplacian $\mathbf{D}^{-1/2} \mathcal{L} \mathbf{D}^{-1/2}$.

Proof. By Lemma 4.8, it takes

$$\tilde{O}(m + n\hat{\epsilon}^{-2}d) = \tilde{O} \left(m + nd \cdot e^{O(\sqrt{d \log d})} \right)$$

time to build the square sparsification chain.

Next we analyze the cost of the recursive calls to SOLVE. As we can see in the description of SOLVE, when invoked on $\mathbf{I} - \mathcal{A}_d$, PRECONRICHARDSON requires $(8/\ell^2) \log(1/\hat{\epsilon})$ iterations, where $\ell = \min\{1/4, 1.125^d \cdot 0.9\hat{\lambda}\}$. Therefore, whenever $d = O(\log \hat{\lambda}^{-1})$, the base case of SOLVE requires

$$O \left(\frac{1}{\hat{\lambda} e^{O(d)}} \log(1/\hat{\epsilon}) \right) = O \left(\frac{\kappa \sqrt{d \log d}}{e^{\Omega(d)}} \right)$$

iterations.

For the cost of invoking $\text{SOLVE}(\mathbf{I} - \mathcal{A}_0, \hat{\lambda}, \hat{\epsilon})$, note that at each recursive call, the branching factor in the recursion is

$$O(\log(1/\hat{\epsilon})) = O \left(\sqrt{d \log d} \right),$$

due to the iterations of preconditioned Richardson, each of them invoking the solver for a matrix from further down the chain.

Furthermore, the depth of the call stack for SOLVE (or the number of layers of the recursion tree) is bounded by $d/\sqrt{d \log d} = \sqrt{d/\log d}$. Therefore the total number of recursive calls made before the final solve on $\mathbf{I} - \mathcal{A}_d$ is

$$O(\sqrt{d \log d})^{O(\sqrt{d/\log d})} = e^{O(\sqrt{d \log d})}.$$

Now, note that each of the recursive call requires one multiplication by each of the matrices in the chain. Therefore each such recursive call makes

$$\tilde{O}(nd\hat{\epsilon}^{-2}) = \tilde{O} \left(nd \cdot e^{O(\sqrt{d \log d})} \right)$$

work. Thus we see that the total amount of work it takes to construct and invoke the implicit matrix given by $\text{SOLVE}(\mathbf{I} - \mathcal{A}_0, \hat{\lambda}, \hat{\epsilon})$ is

$$\tilde{O} \left(nd \cdot e^{O(\sqrt{d \log d})} \right) \cdot e^{O(\sqrt{d \log d})} \cdot O \left(\frac{\kappa \sqrt{d \log d}}{e^{\Omega(d)}} \right) = \tilde{O} \left(n\kappa \cdot \frac{d}{e^{\Omega(\sqrt{d/\log d})}} \right).$$

Hence for our setting of $d = \Theta(\log \kappa)$, this quantity becomes

$$\tilde{O}\left(ne^{O(\sqrt{\log \kappa \log \log \kappa})}\right).$$

Finally, since by definition and Lemma 4.11 we know that $(\mathbf{I} - \mathcal{A}_0)^+$ is a $1/10$ -approximate pseudoinverse of $\mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{-1/2}$ with respect to $\mathbf{U}_{\mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{-1/2}}$, producing the implicit matrix $\widehat{\mathbf{Z}}$ requires $O(\log(1/\epsilon))$ iterations (by Lemma 4.4), each of them requiring one multiplication by $\mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{-1/2}$ and one call to the recursive SOLVE. Thus the total running time to construct and apply $\widehat{\mathbf{Z}}$ in SOLVEEULERIAN is

$$\tilde{O}\left(\left(m + ne^{O(\sqrt{\log \kappa \log \log \kappa})}\right) \log \frac{1}{\epsilon}\right).$$

To analyze the solution quality, let

$$\vec{y} \stackrel{\text{def}}{=} \widehat{\mathbf{Z}}\mathbf{D}^{-1/2}\vec{b},$$

and let $\vec{x} = \mathbf{D}^{-1/2}\vec{y}$ be the solution returned by the algorithm. Also, to simplify notation, let us define $\mathbf{L} = \mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{-1/2}$.

For any vector \vec{v} , let $\mathbf{I}_{\perp v}$ denote orthogonal projection orthogonal to \vec{v} . By Lemma 4.4, we have

$$\left\|\vec{y} - \mathbf{L}^+\mathbf{D}^{-1/2}\vec{b}\right\|_{\mathbf{U}_{\mathbf{L}}} \leq \epsilon \left\|\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b}\right\|_{\mathbf{U}_{\mathbf{L}}}.$$

Since $\mathbf{U}_{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{U}_{\mathcal{L}}\mathbf{D}^{-1/2}$, we have that for every vector \vec{v} , $\|\vec{v}\|_{\mathbf{U}_{\mathbf{L}}} = \|\mathbf{D}^{-1/2}\vec{v}\|_{\mathbf{U}_{\mathcal{L}}}$. Also, since $\ker(\mathbf{U}_{\mathcal{L}}) = \text{span}(\mathbb{1})$ we have that for every vector \vec{v} , $\|\vec{v}\|_{\mathbf{U}_{\mathcal{L}}} = \|\mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\vec{v}\|_{\mathbf{U}_{\mathcal{L}}}$. Using these, the above inequality is equivalent to

$$\epsilon \left\|\mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b}\right\|_{\mathbf{U}_{\mathcal{L}}} \geq \left\|\vec{x} - \mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b}\right\|_{\mathbf{U}_{\mathcal{L}}} = \|\vec{x} - \mathcal{L}^+\|_{\mathbf{U}_{\mathcal{L}}},$$

where we have used $\mathbf{I}_{\perp \mathbb{1}}\vec{x} = \vec{x}$.

To finish the proof, it suffices to show that $\mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b} = \mathcal{L}^+\vec{b}$. We first make the substitution $\mathbf{I}_{\perp \mathbb{1}} = \mathcal{L}^+\mathcal{L}$ and then use $\mathcal{L} = \mathbf{D}^{1/2}\mathbf{L}\mathbf{D}^{1/2}$

$$\mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b} = \mathcal{L}^+\mathcal{L}(\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2})\vec{b} = \mathcal{L}^+\mathbf{D}^{1/2}\mathbf{L}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b}.$$

Since $\mathbf{D}^{1/2}\mathbb{1}$ is the kernel of \mathbf{L} , we have $\mathbf{L}\mathbf{L}^+ = \mathbf{I}_{\perp \mathbf{D}^{1/2}\mathbb{1}}$. By the fact that $\vec{b} \perp \mathbb{1}$, we have $\mathbf{D}^{-1/2}\vec{b} \perp \mathbf{D}^{1/2}\mathbb{1}$, and therefore $\mathbf{I}_{\perp \mathbf{D}^{1/2}\mathbb{1}}\mathbf{D}^{-1/2}\vec{b} = \mathbf{D}^{-1/2}\vec{b}$. Making these substitutions, we obtain

$$\mathbf{I}_{\perp \mathbb{1}}\mathbf{D}^{-1/2}\mathbf{L}^+\mathbf{D}^{-1/2}\vec{b} = \mathcal{L}^+\vec{b}.$$

□

References

- [1] Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krininger, and Richard Peng. On fully dynamic graph sparsifiers. *CoRR*, abs/1604.02094, 2016. Available at: <http://arxiv.org/abs/1604.02094>.
- [2] Reid Andersen, Fan Chung, and Kevin Lang. Local partitioning for directed graphs using pagerank. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 166–178. Springer, 2007.

- [3] Gely P Basharin, Amy N Langville, and Valeriy A Naumov. The life and work of aa markov. *Linear Algebra and its Applications*, 386:3–26, 2004.
- [4] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
- [5] Joshua Batson, Daniel A. Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, August 2013.
- [6] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of computing*, STOC '96, pages 47–55, New York, NY, USA, 1996. ACM.
- [7] Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. Efficient sampling for Gaussian graphical models via spectral sparsification. *Proceedings of The 28th Conference on Learning Theory*, pages 364–390, 2015. Available at <http://jmlr.org/proceedings/papers/v40/Cheng15.pdf>.
- [8] Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.
- [9] Fan Chung and Olivia Simpson. Solving linear systems with boundary conditions using heat kernel pagerank. In *Algorithms and Models for the Web Graph - 10th International Workshop, WAW 2013, Cambridge, MA, USA, December 14-15, 2013, Proceedings*, pages 203–219, 2013.
- [10] Fan Chung and Olivia Simpson. Computing heat kernel pagerank and a local clustering algorithm. In *Combinatorial Algorithms - 25th International Workshop, IWOCA 2014, Duluth, MN, USA, October 15-17, 2014, Revised Selected Papers*, pages 110–121, 2014.
- [11] Fan Chung and Wenbo Zhao. A sharp pagerank algorithm with applications to edge ranking and graph sparsification. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 2–14. Springer, 2010.
- [12] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. *arXiv preprint arXiv:1608.03270*, 2016.
- [13] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *STOC*, pages 343–352, 2014.
- [14] Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{o}(m^{10/7} \log W)$ time. *CoRR*, abs/1605.01717, 2016.
- [15] Alina Ene, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Routing under balance. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 598–611, 2016. Available at: <https://arxiv.org/abs/1603.09009>.

- [16] Wai Shing Fung, Ramesh Hariharan, Nicholas J.A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 71–80, New York, NY, USA, 2011. ACM. Available at <http://arxiv.org/abs/1004.4080>.
- [17] Gorav Jindal and Pavel Kolev. An efficient parallel algorithm for spectral sparsification of laplacian and sddm matrix polynomials. *arXiv preprint arXiv:1507.07497*, 2015.
- [18] David R Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 648–657. ACM, 1994.
- [19] David R Karger. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, 47(1):46–76, 2000.
- [20] David R Karger and Matthew S Levine. Random sampling in residual graphs. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 63–66. ACM, 2002.
- [21] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron 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, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226, 2014.
- [22] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the 45th Annual Symposium on Theory of Computing*, STOC '13, pages 911–920, New York, NY, USA, 2013. ACM. Available at <http://arxiv.org/abs/1301.6628>.
- [23] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 235–244, Washington, DC, USA, 2010. IEEE Computer Society. Available at <http://arxiv.org/abs/1003.2958>.
- [24] Ioannis Koutis, Gary L. Miller, and Richard Peng. 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*, FOCS '11, pages 590–598, Washington, DC, USA, 2011. IEEE Computer Society. Available at <http://arxiv.org/abs/1102.4842>.
- [25] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, pages 842–850. ACM, 2016. Available at <http://arxiv.org/abs/1512.01892>.
- [26] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians: Fast, sparse, and simple. *CoRR*, abs/1605.02353, 2016.
- [27] Yin Tat Lee and Aaron 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*, pages 147–156. IEEE, 2013.
- [28] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}\sqrt{\text{rank}}$ iterations and faster algorithms for maximum flow. In *Foundations of*

- Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 424–433. IEEE, 2014. Available at <http://arxiv.org/abs/1312.6677> and <http://arxiv.org/abs/1312.6713>.
- [29] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 250–269, Oct 2015.
- [30] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 253–262, 2013.
- [31] Aleksander Madry. Computing maximum flow with augmenting electrical flows. *CoRR*, abs/1608.06016, 2016.
- [32] Lorenzo Orecchia and Nisheeth K. Vishnoi. Towards an SDP-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 532–545, 2011.
- [33] Richard Peng and Daniel A. Spielman. An efficient parallel solver for SDD linear systems. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC '14*, pages 333–342, New York, NY, USA, 2014. ACM. Available at <http://arxiv.org/abs/1311.3286>.
- [34] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. Available at: <http://www-users.cs.umn.edu/~saad/toc.pdf>.
- [35] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. Available at <http://arxiv.org/abs/0803.0929>.
- [36] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. Available at <http://arxiv.org/abs/0808.4134>.
- [37] Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. Available at <http://arxiv.org/abs/cs/0607105>.
- [38] Williams J Stewart. *Introduction to the numerical solutions of Markov chains*. Princeton Univ. Press, 1994.
- [39] Shang-Hua Teng. The Laplacian paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation, TAMC'10*, pages 2–14, Berlin, Heidelberg, 2010. Springer-Verlag.
- [40] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.
- [41] Nisheeth K. Vishnoi. $Lx = b$. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2013.
- [42] Zeyuan Allen Zhu, Zhenyu Liao, and Lorenzo Orecchia. Spectral sparsification and regret minimization beyond matrix multiplicative updates. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 237–245, 2015.

A Entrywise Sparsification

In this section we prove Theorem 3.9, our main result about entrywise sampling for sparsification. Our main technical tool for this is a rectangular matrix concentration of Tropp that we restate below:

Theorem A.1 (Matrix Bernstein (Theorem 1.6 of [40], restated)). *Let $\mathbf{Z}_1, \dots, \mathbf{Z}_k \in \mathbb{R}^{d_1 \times d_2}$ be independent random matrices such that $\mathbb{E}\mathbf{Z}_i = \mathbf{0}$ and $\|\mathbf{Z}_i\|_2 \leq R$ almost surely for all i . Then*

$$\Pr \left[\left\| \sum_{i \in [k]} \mathbf{Z}_i \right\|_2 \geq t \right] \leq (d_1 + d_2) \cdot \exp \left(\frac{-t^2/2}{\sigma^2 + Rt/3} \right)$$

where

$$\sigma^2 \stackrel{\text{def}}{=} \max \left\{ \left\| \sum_{i \in [k]} \mathbb{E}\mathbf{Z}_i \mathbf{Z}_i^\top \right\|_2, \left\| \sum_{i \in [k]} \mathbb{E}\mathbf{Z}_i^\top \mathbf{Z}_i \right\|_2 \right\}.$$

First we simplify this theorem, tailoring it to the case where we are sampling a sequence of matrices with the same expectation.

Theorem A.2. *Let \mathcal{D} be a distribution over $\mathbb{R}^{d_1 \times d_2}$. Let $\Sigma \stackrel{\text{def}}{=} \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A}$ and let $R_{\mathcal{D}}$ and $\sigma_{\mathcal{D}}^2$ satisfy*

$$\max \left\{ \|\mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A} \mathbf{A}^\top\|_2, \|\mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A}^\top \mathbf{A}\|_2 \right\} \leq \sigma_{\mathcal{D}}^2 \quad \text{and} \quad \max_{\mathbf{A} \in \text{supp}(\mathcal{D})} \|\mathbf{A}\|_2 \leq R_{\mathcal{D}}.$$

Then for $\mathbf{A}_1, \dots, \mathbf{A}_k$ sampled independently from \mathcal{D} we have

$$\Pr \left[\left\| \frac{1}{k} \sum_{i \in [k]} \mathbf{A}_i - \Sigma \right\|_2 \geq \epsilon \right] \leq (d_1 + d_2) \cdot \exp \left(\frac{-k\epsilon^2/2}{\sigma_{\mathcal{D}}^2 + R_{\mathcal{D}}\epsilon} \right).$$

and for $k \geq 64 \cdot \left(\frac{\sigma_{\mathcal{D}}^2}{\epsilon^2} + \frac{R_{\mathcal{D}}}{\epsilon} \right) \log \frac{d}{p}$ it is the case that $\Pr \left[\left\| \frac{1}{k} \sum_{i \in [k]} \mathbf{A}_i - \Sigma \right\|_2 \geq \epsilon \right] \leq p$.

Proof. Let $\mathbf{Z}_i \stackrel{\text{def}}{=} \mathbf{A}_i - \Sigma$. Clearly $\mathbb{E}\mathbf{Z}_i = \mathbf{0}$, and by Jensen's inequality,

$$\|\mathbf{Z}_i\|_2 \leq \|\mathbf{A}_i\|_2 + \|\Sigma\|_2 \leq \|\mathbf{A}_i\|_2 + \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \|\mathbf{A}\|_2 \leq 2 \cdot R_{\mathcal{D}}.$$

Furthermore,

$$\mathbf{0} \preceq \mathbb{E}\mathbf{Z}_i^\top \mathbf{Z}_i = \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} (\mathbf{A} - \Sigma)^\top (\mathbf{A} - \Sigma) = \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A}^\top \mathbf{A} - \Sigma^\top \Sigma \preceq \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A}^\top \mathbf{A}$$

and

$$\mathbf{0} \preceq \mathbb{E}\mathbf{Z}_i \mathbf{Z}_i^\top = \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} (\mathbf{A} - \Sigma) (\mathbf{A} - \Sigma)^\top = \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A} \mathbf{A}^\top - \Sigma \Sigma^\top \preceq \mathbb{E}_{\mathbf{A} \sim \mathcal{D}} \mathbf{A} \mathbf{A}^\top.$$

Consequently,

$$\max \left\{ \left\| \sum_{i \in [k]} \mathbb{E}\mathbf{Z}_i \mathbf{Z}_i^\top \right\|_2, \left\| \sum_{i \in [k]} \mathbb{E}\mathbf{Z}_i^\top \mathbf{Z}_i \right\|_2 \right\} \leq k \cdot \sigma_{\mathcal{D}}^2.$$

Therefore, by Theorem A.1 we have that for all t ,

$$\Pr \left[\left\| \sum_{i \in [k]} \mathbf{Z}_i \right\|_2 \geq t \right] \leq (d_1 + d_2) \cdot \exp \left(\frac{-t^2/2}{k \cdot \sigma_{\mathcal{D}}^2 + 2Rt/3} \right).$$

Since $\sum_{i \in [k]} \mathbf{Z}_i = k \cdot \left(\frac{1}{k} \sum_{i \in [k]} \mathbf{A}_i - \Sigma \right)$ picking $t = k \cdot \epsilon$ yields the result. \square

Using Theorem A.2 we can now prove Theorem 3.9, our main result of this section.

Proof of Theorem 3.9. First note that by the definition of s we have that

$$\sum_{i,j} p_{ij} = \frac{1}{s} \sum_{i,j} \left[\frac{\mathbf{A}_{ij}}{\vec{r}_i} + \frac{\mathbf{A}_{ij}}{\vec{c}_j} \right] = \frac{1}{s} [\# \text{ non-zero rows} + \# \text{ non-zero columns}] = 1$$

and therefore \mathcal{D} is a valid probability distribution. All that remains is to prove each of the claims of Theorem 3.9 by carefully applying Theorem A.2.

First apply Theorem A.2 for the distribution \mathcal{D} which assigns probability p_{ij} to matrix $\frac{\mathbf{A}_{ij} \vec{1}_i \vec{1}_j^\top}{\sqrt{\vec{r}_i \cdot \vec{c}_j}}$. For this application of Theorem A.2, using that $x \cdot y \leq \frac{1}{2}x^2 + \frac{1}{2}y^2$ we have

$$R_{\mathcal{D}} = \max_{i,j} \left\| \frac{\mathbf{A}_{ij} \vec{1}_i \vec{1}_j^\top}{\sqrt{\vec{r}_i \cdot \vec{c}_j}} \cdot \frac{1}{p_{ij}} \right\| = \max_{i,j} s \cdot \frac{1}{\sqrt{\vec{r}_i \cdot \vec{c}_j}} \left(\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right)^{-1} \leq \frac{s}{2}.$$

Furthermore we have

$$\left\| \mathbb{E}_{\mathbf{M} \sim \mathcal{D}} \mathbf{M} \mathbf{M}^\top \right\|_2 = s \left\| \sum_{i,j} \frac{1}{\vec{r}_i} \cdot \frac{1}{\vec{c}_j} \cdot \mathbf{A}_{ij} \cdot \vec{1}_i \vec{1}_i^\top \cdot \left(\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right)^{-1} \right\|_2 \leq s$$

and

$$\left\| \mathbb{E}_{\mathbf{M} \sim \mathcal{D}} \mathbf{M}^\top \mathbf{M} \right\|_2 = s \left\| \sum_{i,j} \frac{1}{\vec{r}_i} \cdot \frac{1}{\vec{c}_j} \cdot \mathbf{A}_{ij} \cdot \vec{1}_j \vec{1}_j^\top \cdot \left(\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right)^{-1} \right\|_2 \leq s.$$

Consequently, $\sigma_{\mathcal{D}} \leq s$ and since $\epsilon \in (0, 1)$ and k is chosen appropriately, the first inequality follows by Theorem A.2.

Next, we apply Theorem A.2 for the distribution \mathcal{D} which assigns probability p_{ij} to matrix $\vec{1}_i \frac{\mathbf{A}_{ij}}{\vec{r}_i}$. For this application of Theorem A.2 we have

$$R_{\mathcal{D}} = \max_{i,j} \left\| \frac{\mathbf{A}_{ij}}{\vec{r}_i} \cdot \frac{1}{p_{ij}} \right\| \leq s \cdot \frac{1}{\vec{r}_i} \cdot \left(\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right)^{-1} \leq s.$$

Furthermore we have

$$\sigma_{\mathcal{D}}^2 = \left\| \sum_j \frac{1}{\vec{r}_j^2} \cdot \mathbf{A}_{ij}^2 \cdot \left(\frac{1}{\vec{r}_i} + \frac{1}{\vec{c}_j} \right)^{-1} \cdot s \right\|_2 \leq s.$$

Since $\epsilon \in (0, 1)$, with k appropriately chosen, the second inequality follows by Theorem A.2. By symmetry the last inequality follows as well. Finally, by choosing p to be s times larger we can make all conditions hold simultaneously by union bound. \square

B Linear Algebra Facts

In this section we provide various general linear algebra facts we use throughout the paper.

Lemma B.1. *Suppose that $\|\mathbf{A} - \mathbf{B}\|_2 \leq \epsilon$ then for all $c > 0$ we have*

$$(1 - c)\mathbf{B}^\top \mathbf{B} - c^{-1}\epsilon^2 \mathbf{I} \preceq \mathbf{A}^\top \mathbf{A} \preceq (1 + c)\mathbf{B}^\top \mathbf{B} + (1 + c^{-1})\epsilon^2 \mathbf{I}$$

Proof. Using the trivial expansion of $\mathbf{A} = \mathbf{B} + (\mathbf{A} - \mathbf{B})$ we have that

$$x^\top \mathbf{A}^\top \mathbf{A} x - x^\top \mathbf{B}^\top \mathbf{B} x = 2x\mathbf{B}^\top (\mathbf{A} - \mathbf{B})x + x^\top (\mathbf{A} - \mathbf{B})^\top (\mathbf{A} - \mathbf{B})x$$

Now since $xy \leq \frac{c}{2}x^2 + \frac{1}{2c}y^2$ for all x, y and $c > 0$ we have

$$\left| 2x\mathbf{B}^\top (\mathbf{A} - \mathbf{B})x \right| \leq 2\|\mathbf{B}x\|_2 \|(\mathbf{A} - \mathbf{B})x\|_2 \leq c\|\mathbf{B}x\|_2^2 + c^{-1}\|(\mathbf{A} - \mathbf{B})x\|_2^2.$$

Combining this with the fact that

$$0 \leq x^\top (\mathbf{A} - \mathbf{B})^\top (\mathbf{A} - \mathbf{B})x = \|(\mathbf{A} - \mathbf{B})x\|_2^2 \leq \epsilon^2 \|x\|_2^2 = \epsilon^2 x^\top \mathbf{I} x$$

we obtain the result. \square

Lemma B.2. For all $\mathbf{A} \in \mathbb{R}^{n \times n}$ and symmetric PSD $\mathbf{M}, \mathbf{N} \in \mathbb{R}^{n \times n}$ such that $\ker(\mathbf{M}) \subseteq \ker(\mathbf{A}^\top)$ and $\ker(\mathbf{N}) \subseteq \ker(\mathbf{A})$ we have

$$\|\mathbf{M}^{-1/2} \mathbf{A} \mathbf{N}^{-1/2}\|_2 = \max_{x, y \neq 0} \frac{x^\top \mathbf{A} y}{\sqrt{(x^\top \mathbf{M} x)(y^\top \mathbf{N} y)}} = 2 \cdot \max_{x, y \neq 0} \frac{x^\top \mathbf{A} y}{x^\top \mathbf{M} x + y^\top \mathbf{N} y}$$

where in each of the maximization problems we define $0/0$ to be 0 .

Proof. Let $L \stackrel{\text{def}}{=} \|\mathbf{M}^{-1/2} \mathbf{A} \mathbf{N}^{-1/2}\|_2$. Since $\|x\|_2 = \max_{\|y\|_2=1} y^\top x$ we have that

$$L = \max_{\|x\|_2 = \|y\|_2 = 1} x^\top \mathbf{M}^{-1/2} \mathbf{A} \mathbf{N}^{-1/2} y.$$

Now, performing the change of basis $x := \mathbf{M}^{-1/2} x$ and $y := \mathbf{N}^{-1/2} y$ we have

$$L = \max_{\|x\|_{\mathbf{M}} = \|y\|_{\mathbf{N}} = 1} x^\top \mathbf{A} y = \max_{x, y \neq 0} \frac{x^\top \mathbf{A} y}{\|x\|_{\mathbf{M}} \|y\|_{\mathbf{M}}}$$

where in each of these maximization problems we restrict that $x \in \text{im}(\mathbf{M})$ and $y \in \text{im}(\mathbf{N})$. However, for all $x \perp \text{im}(\mathbf{M})$ or $y \perp \text{im}(\mathbf{N})$, i.e. $x \in \ker(\mathbf{M})$ or $y \in \ker(\mathbf{N})$, we have that either $\|x\|_{\mathbf{M}} = 0$ or $\|y\|_{\mathbf{N}} = 0$ and $x^\top \mathbf{A} y = 0$. Consequently, the above equalities hold without the $x \in \text{im}(\mathbf{M})$ and $y \in \text{im}(\mathbf{N})$ restriction by our definition of $0/0 = 0$. The final equality we wish to prove follows from the fact that $\|x\|_{\mathbf{M}} \|y\|_{\mathbf{N}} \leq \frac{1}{2}(\|x\|_{\mathbf{M}}^2 + \|y\|_{\mathbf{N}}^2)$ and that this inequality is tight when $\|x\|_{\mathbf{M}} = \|y\|_{\mathbf{N}} = 1$. \square

Lemma B.3. For any $\mathbf{M} \in \mathbb{R}^{n \times n}$ and symmetric positive semidefinite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ such that $\mathbf{A} \preceq \mathbf{B}$ we have that $\|\mathbf{A}^{1/2} \mathbf{M}\|_2 \leq \|\mathbf{B}^{1/2} \mathbf{M}\|_2$ and $\|\mathbf{M} \mathbf{A}^{1/2}\|_2 \leq \|\mathbf{M} \mathbf{B}^{1/2}\|_2$.

Proof. The first claim follows from the fact that adopting the convention $0/0 = 0$

$$\|\mathbf{A}^{1/2} \mathbf{M}\|_2 = \max_{x \in \mathbb{R}^n} \frac{\|\mathbf{A}^{1/2} \mathbf{M} x\|_2}{\|x\|_2} = \max_{x \in \mathbb{R}^n} \frac{\sqrt{x^\top \mathbf{M}^\top \mathbf{A} \mathbf{M} x}}{\|x\|_2} \leq \max_{x \in \mathbb{R}^n} \frac{\sqrt{x^\top \mathbf{M}^\top \mathbf{B} \mathbf{M} x}}{\|x\|_2} = \|\mathbf{B}^{1/2} \mathbf{M}\|_2.$$

The second follows from this and the fact that $\|\mathbf{A}^{1/2} \mathbf{M}\|_2 = \|\mathbf{M}^\top \mathbf{A}^{1/2}\|_2$. \square

Lemma B.4. Let $\mathbf{M} \in \mathbb{R}^{n \times m}$, $a \in \mathbb{R}_{\geq 0}^n$ and $b \in \mathbb{R}_{\geq 0}^m$ be arbitrary. Let $\mathbf{A} = \text{diag}(a)$ and $\mathbf{B} = \text{diag}(b)$. We have that for all $\alpha, \beta \in [0, 1]$

$$\|\mathbf{A} \mathbf{M} \mathbf{B}\|_2 \leq \sqrt{\|\mathbf{A}^{2\alpha} \mathbf{M} \mathbf{B}^{2\beta}\|_\infty \cdot \|\mathbf{A}^{2(1-\alpha)} \mathbf{M} \mathbf{B}^{2(1-\beta)}\|_1}.$$

Consequently, for PSD diagonal matrices $\mathbf{D}_1 \in \mathbb{R}^{n \times n}$ and $\mathbf{D}_2 \in \mathbb{R}^{m \times m}$ we have

$$\|\mathbf{D}_1^{-1/2} \mathbf{M} \mathbf{D}_2^{-1/2}\|_2 \leq \max \left\{ \|\mathbf{D}_1^{-1} \mathbf{M}\|_\infty, \|\mathbf{D}_2^{-1} \mathbf{M}^\top\|_\infty \right\}.$$

Proof. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be arbitrary with $\|x\|_2 = \|y\|_2 = 1$. We have

$$x^\top \mathbf{A} \mathbf{M} \mathbf{B} y = \sum_{i,j} \mathbf{M}_{i,j} a_i b_j x_i y_j \leq \sum_{i,j} |\mathbf{M}_{i,j}| \cdot a_i \cdot b_j \cdot |x_i| \cdot |y_j|$$

Consequently, by Cauchy Schwarz we have that

$$\begin{aligned} (x^\top \mathbf{A} \mathbf{M} \mathbf{B} y)^2 &\leq \left(\sum_{i,j} |\mathbf{M}_{i,j}| \cdot a_i^{2\alpha} \cdot b_j^{2\beta} \cdot x_i^2 \right) \cdot \left(\sum_{i,j} |\mathbf{M}_{i,j}| \cdot a_i^{2(1-\alpha)} \cdot b_j^{2(1-\beta)} \cdot y_j^2 \right) \\ &\leq \|\mathbf{A}^{2\alpha} \mathbf{M} \mathbf{B}^{2\beta}\|_\infty \cdot \|\mathbf{A}^{2(1-\alpha)} \mathbf{M} \mathbf{B}^{2(1-\beta)}\|_1. \end{aligned}$$

The final conclusion follows from the fact that for any matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$

$$\|\mathbf{C}\|_1 = \|\mathbf{C}^\top\|_\infty.$$

□

Lemma B.5. *If $\mathbf{M} \in \mathbb{R}^{n \times n}$ is a symmetric matrix with $\|\mathbf{M}\|_2 \leq 1$, then $\mathbf{0} \preceq \mathbf{I} - \mathbf{M}^2 \preceq 2 \cdot (\mathbf{I} - \mathbf{M})$.*

Proof. Since \mathbf{M} is symmetric we have that \mathbf{M}^2 and \mathbf{M} are mutually diagonalizable and the above inequality reduces to showing that $0 \leq 1 - x^2 \leq 2 \cdot (1 - x)$ for $x \in \mathbb{R}$ with $|x| \leq 1$. The left hand side of the inequality is true because $x^2 \leq 1$ and the right hand side follows by noticing that it is equivalent to $0 \leq x^2 - 2x + 1 = (x - 1)^2$ after rearranging terms. □

The following gives a similar statement involving the symmetrization of an arbitrary matrix. While it is based on the above proof, it loses a factor of 2 over the previous lemma.

Lemma B.6. *If $\mathbf{M} \in \mathbb{R}^{n \times n}$ is a possibly asymmetric matrix satisfying $\|\mathbf{M}\|_2 \leq 1$ then*

$$\mathbf{0} \preceq \mathbf{I} - \mathbf{U}_{\mathbf{M}^2} \preceq 2(\mathbf{I} - \mathbf{U}_{\mathbf{M}}^2) \preceq 4(\mathbf{I} - \mathbf{U}_{\mathbf{M}}).$$

Proof. Since the norm of \mathbf{M} is at most 1, we immediately obtain $\|(\mathbf{M}^2)^\top\|_2 = \|\mathbf{M}^2\|_2 \leq 1$, $\|\mathbf{M}^\top \mathbf{M}\|_2 \leq 1$, $\|\mathbf{M} \mathbf{M}^\top\|_2 \leq 1$. Then, by triangle inequality,

$$\|\mathbf{U}_{\mathbf{M}^2}\|_2 = \left\| \frac{1}{2}(\mathbf{M}^2 + (\mathbf{M}^2)^\top) \right\|_2 \leq \frac{1}{2}(\|\mathbf{M}^2\|_2 + \|(\mathbf{M}^2)^\top\|_2) \leq 1.$$

Therefore $\mathbf{U}_{\mathbf{M}^2} \preceq \mathbf{I}$, yielding the left hand side of the desired inequality.

Next, we note that these inequalities imply $\mathbf{M}^\top \mathbf{M} \preceq \mathbf{I}$ and $\mathbf{M} \mathbf{M}^\top \preceq \mathbf{I}$, yielding

$$(\mathbf{M} + \mathbf{M}^\top)^2 = \mathbf{M}^2 + \mathbf{M}^\top \mathbf{M} + \mathbf{M} \mathbf{M}^\top + (\mathbf{M}^\top)^2 \preceq \mathbf{M}^2 + (\mathbf{M}^\top)^2 + 2\mathbf{I}.$$

Consequently,

$$\mathbf{I} - \mathbf{U}_{\mathbf{M}^2} = \mathbf{I} - \frac{1}{2}(\mathbf{M}^2 + (\mathbf{M}^2)^\top) \preceq 2\mathbf{I} - \frac{1}{2}(\mathbf{M} + \mathbf{M}^\top)^2 = 2(\mathbf{I} - \mathbf{U}_{\mathbf{M}}^2).$$

Finally, since $\mathbf{U}_{\mathbf{M}}$ is symmetric with $\|\mathbf{U}_{\mathbf{M}}\|_2 \leq 1$, by Lemma B.5 we have $\mathbf{I} - \mathbf{U}_{\mathbf{M}}^2 \preceq 2 \cdot (\mathbf{I} - \mathbf{U}_{\mathbf{M}})$. □

Lemma B.7. *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a matrix such that $\|\mathbf{M}\|_2 \leq 1$. Furthermore, for $\alpha \in [0, 1)$ let $\mathbf{N} = \alpha \mathbf{I} + (1 - \alpha) \mathbf{M}$ and let $\mathbf{L}_i = \mathbf{I} - \mathbf{U}_{\mathbf{N}^i}$. Then,*

$$2\alpha \mathbf{L}_1 \preceq \mathbf{L}_2 \preceq (4 - 2\alpha) \cdot \mathbf{L}_1.$$

Proof. Note that $\mathbf{I} - \mathbf{N} = (1 - \alpha)(\mathbf{I} - \mathbf{M})$, and therefore $\mathbf{L}_1 = (1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M})$. The first identity gives us that

$$\mathbf{N}^2 = \alpha^2 \mathbf{I} + 2\alpha(1 - \alpha)\mathbf{M} + (1 - \alpha)^2 \mathbf{M}^2.$$

Consequently,

$$\begin{aligned} \mathbf{L}_2 &= \mathbf{I} - \alpha^2 \mathbf{I} - 2\alpha(1 - \alpha)\mathbf{U}_\mathbf{M} - (1 - \alpha)^2 \mathbf{U}_{\mathbf{M}^2} \\ &= (1 - \alpha^2 - (1 - \alpha)^2) \mathbf{I} - 2\alpha(1 - \alpha)\mathbf{U}_\mathbf{M} + (1 - \alpha^2) (\mathbf{I} - \mathbf{U}_{\mathbf{M}^2}) \\ &= 2\alpha(1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M}) + (1 - \alpha)^2 (\mathbf{I} - \mathbf{U}_{\mathbf{M}^2}) \\ &= 2\alpha \mathbf{L}_1 + (1 - \alpha)^2 (\mathbf{I} - \mathbf{U}_{\mathbf{M}^2}). \end{aligned}$$

This yields the first part of the inequality, since the second term in the last line is positive semidefinite. Now by Lemma B.6 we know that

$$\mathbf{0} \preceq \mathbf{I} - \mathbf{U}_{\mathbf{M}^2} \preceq 4(\mathbf{I} - \mathbf{U}_\mathbf{M}) = \frac{4}{1 - \alpha} \mathbf{L}_1,$$

Plugging this into our previous identity we obtain

$$\mathbf{L}_2 \preceq 2\alpha \mathbf{L}_1 + (1 - \alpha)^2 \cdot \frac{4}{1 - \alpha} \mathbf{L}_1 = (4 - 2\alpha) \mathbf{L}_1,$$

thus yielding the result. \square

Lemma B.8 (Condition Number Improvement). *Let a nonzero matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ be such that $\ker(\mathbf{M}) = \ker(\mathbf{M}^\top)$, and $\|\mathbf{M}\|_2 \leq 1$. For $\alpha \in (0, 1/4]$ let $\mathbf{N} \stackrel{\text{def}}{=} \alpha \mathbf{I} + (1 - \alpha)\mathbf{M}$. Then, for $\lambda_* \stackrel{\text{def}}{=} \lambda_*(\mathbf{I} - \mathbf{U}_\mathbf{M})$ we have*

$$\lambda_*(\mathbf{I} - \mathbf{U}_{\mathbf{N}^2}) \geq \min\{\alpha, (1 + \alpha)\lambda_*\}. \quad (\text{B.1})$$

Proof. Note that we can write

$$\mathbf{U}_{\mathbf{N}^2} = \frac{1}{2} (\mathbf{N}^2 + (\mathbf{N}^\top)^2) = \left(\frac{1}{2} (\mathbf{N} + \mathbf{N}^\top) \right)^2 - \left(\frac{1}{2} (\mathbf{N} - \mathbf{N}^\top) \cdot \frac{1}{2} (\mathbf{N} - \mathbf{N}^\top)^\top \right) \preceq \mathbf{U}_\mathbf{N}^2.$$

Therefore $\mathbf{I} - \mathbf{U}_{\mathbf{N}^2} \succeq \mathbf{I} - \mathbf{U}_\mathbf{N}^2$, so it is sufficient to lower bound the smallest nonzero eigenvalue of the latter. By expanding, we obtain:

$$\mathbf{I} - \mathbf{U}_\mathbf{N}^2 = \mathbf{I} - (\alpha \mathbf{I} + (1 - \alpha)\mathbf{U}_\mathbf{M})^2 = \mathbf{I} - (\mathbf{I} - (1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M}))^2.$$

Since $\|\mathbf{M}\|_2 \leq 1$, we also have $\|\mathbf{U}_\mathbf{M}\|_2 \leq 1$ by triangle inequality, and thus $\lambda_* \mathbf{I}_{im(\mathbf{M})} \preceq \mathbf{I} - \mathbf{U}_\mathbf{M} \preceq 2\mathbf{I}_{im(\mathbf{M})}$. Therefore,

$$\mathbf{I} - (1 - \alpha)2\mathbf{I}_{im(\mathbf{M})} \preceq \mathbf{I} - (1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M}) \preceq \mathbf{I} - (1 - \alpha)\lambda_* \mathbf{I}_{im(\mathbf{M})},$$

and equivalently

$$\mathbf{I}_{\perp im(\mathbf{M})} + (2\alpha - 1)\mathbf{I}_{im(\mathbf{M})} \preceq \mathbf{I} - (1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M}) \preceq \mathbf{I}_{\perp im(\mathbf{M})} + (1 - (1 - \alpha)\lambda_*)\mathbf{I}_{im(\mathbf{M})}.$$

Hence, after squaring, each eigenvalue of the middle term will become upper bounded by the maximum of the squares of those in the lower and the upper bound. This can be seen as a matrix version of the inequality $b^2 \leq \max\{a^2, c^2\}$, if $a \leq b \leq c$. Hence,

$$(\mathbf{I} - (1 - \alpha)(\mathbf{I} - \mathbf{U}_\mathbf{M}))^2 \preceq \mathbf{I}_{\perp im(\mathbf{M})} + \max\{(2\alpha - 1)^2, (1 - (1 - \alpha)\lambda_*)^2\} \mathbf{I}_{im(\mathbf{M})},$$

so after subtracting both sides from \mathbf{I} we obtain:

$$\mathbf{I} - \mathbf{U}_{\mathbf{N}}^2 \succeq 1 - \max\{(2\alpha - 1)^2, (1 - (1 - \alpha)\lambda_*)^2\} \mathbf{I}_{im(\mathbf{M})}.$$

Therefore,

$$\begin{aligned} \lambda_*(\mathbf{I} - \mathbf{U}_{\mathbf{N}^2}) &\geq \min\{1 - (2\alpha - 1)^2, 1 - (1 - (1 - \alpha)\lambda_*)^2\} \\ &= \min\{1 - (2\alpha - 1)^2, 2(1 - \alpha)\lambda_* - (1 - \alpha)^2\lambda_*^2\}. \end{aligned}$$

Observe that if $\lambda_* \leq (1 - 3\alpha)(1 - \alpha)^{-2}$, then the second part of the lower bound is at least

$$2(1 - \alpha)\lambda_* - (1 - \alpha)^2(1 - 3\alpha)(1 - \alpha)^{-2}\lambda_* = (1 + \alpha)\lambda_*.$$

Otherwise, it can be lower bounded simply by

$$2(1 - \alpha)\lambda_* - (1 - \alpha)^2\lambda_* = (1 - \alpha^2)\lambda_* \geq 1 - 3\alpha.$$

Finally, since for $\alpha \leq 1/4$, both $1 - 3\alpha \geq \alpha$ and $1 - (1 - 2\alpha)^2 \geq \alpha$ are true, the result follows. \square

Lemma B.9. *If \mathbf{L} is a matrix with $\ker(\mathbf{L}) = \ker(\mathbf{L}^\top) = \ker(\mathbf{U}_{\mathbf{L}})$, and $\mathbf{U}_{\mathbf{L}}$ is positive semidefinite, then*

$$\mathbf{U}_{\mathbf{L}} \preceq \mathbf{L}^\top \mathbf{U}_{\mathbf{L}}^+ \mathbf{L}.$$

Furthermore, for any matrix \mathbf{A} with the same left and right kernels as \mathbf{L} , one has that

$$\|\mathbf{A}\|_{\mathbf{U}_{\mathbf{L}} \rightarrow \mathbf{U}_{\mathbf{L}}} \leq \left\| \mathbf{U}_{\mathbf{L}}^{+/2} \mathbf{L} \mathbf{A} \mathbf{U}_{\mathbf{L}}^{+/2} \right\|_2.$$

Proof. We decompose \mathbf{L} and \mathbf{L}^\top as the sum/difference of a symmetric matrix \mathbf{U} and a skew symmetric matrix \mathbf{V} . Specifically, write $\mathbf{L} = \mathbf{U} + \mathbf{V}$ for

$$\mathbf{U} := \mathbf{U}_{\mathbf{L}} = (\mathbf{L} + \mathbf{L}^\top)/2 \text{ and } \mathbf{V} := (\mathbf{L} - \mathbf{L}^\top)/2.$$

This gives

$$\mathbf{L}^\top \mathbf{U} + \mathbf{L} = \mathbf{U}^\top \mathbf{U} + \mathbf{U} + \mathbf{U}^\top \mathbf{V} + \mathbf{V}^\top \mathbf{U} + \mathbf{U} + \mathbf{V}^\top \mathbf{V} + \mathbf{V}.$$

As $\mathbf{U}\mathbf{U}^\top \mathbf{V} = \mathbf{V}$ and $\mathbf{V}^\top \mathbf{U} + \mathbf{U} = \mathbf{V}^\top$ by our kernel assumptions, and $\mathbf{V} = -\mathbf{V}^\top$, this simplifies to

$$\mathbf{L}^\top \mathbf{U} + \mathbf{L} = \mathbf{U} + \mathbf{V}^\top \mathbf{U} + \mathbf{V} \succeq \mathbf{U},$$

where we used the assumption that $\mathbf{U} \succeq \mathbf{0}$ to guarantee that $\mathbf{V}^\top \mathbf{U} + \mathbf{V} \succeq \mathbf{0}$ for the final inequality.

The second part of the lemma follows by writing

$$\|\mathbf{A}\|_{\mathbf{U}_{\mathbf{L}} \rightarrow \mathbf{U}_{\mathbf{L}}} = \left\| \mathbf{U}_{\mathbf{L}}^{1/2} \mathbf{A} \mathbf{U}_{\mathbf{L}}^{+/2} \right\|_2 = \left\| \mathbf{U}_{\mathbf{L}}^{1/2} \mathbf{L}^+ \left(\mathbf{L} \mathbf{A} \mathbf{U}_{\mathbf{L}}^{+/2} \right) \right\|_2,$$

then applying the equivalent form of the previous inequality $\mathbf{L}^\top \mathbf{U}_{\mathbf{L}} \mathbf{L}^+ \preceq \mathbf{U}_{\mathbf{L}}^+$ in order to obtain

$$\left\| \mathbf{U}_{\mathbf{L}}^{1/2} \mathbf{L}^+ \left(\mathbf{L} \mathbf{A} \mathbf{U}_{\mathbf{L}}^{+/2} \right) \right\|_2 \leq \left\| \mathbf{U}_{\mathbf{L}}^{+/2} \mathbf{L} \mathbf{A} \mathbf{U}_{\mathbf{L}}^{+/2} \right\|_2.$$

\square

C Decomposition

Here we discuss the proof of Theorem 3.15, our main result on decomposing directed graphs. The theorem follows directly from standard results involving decomposing undirected graphs into expanders [37, 36, 21].

Before stating the decomposition result, we first define the *conductance* of a graph:

Definition C.1. Given an undirected graph $G(V, E, w)$, we define the conductance of a $S \subseteq V$ by

$$\Phi(S) \stackrel{\text{def}}{=} \frac{\sum_{(u,v) \in E: u \in S, v \notin S} w_{uv}}{\min\{\text{vol}(S), \text{vol}(V \setminus S)\}},$$

where $\text{vol}(S) \stackrel{\text{def}}{=} \sum_{u \in S} \sum_{v: (u,v) \in E} w_{uv}$.

The conductance of G is then defined as

$$\Phi(G) = \min_{S \subseteq V, S \neq \emptyset} \Phi(S)$$

Relating the conductance of an undirected graph to its smallest nontrivial eigenvalue is done via Cheeger's inequality:

Theorem C.2 (Cheeger's inequality, rephrased). *Given an undirected graph $G(V, E, w)$, with a symmetric Laplacian $\mathbf{U} = \mathbf{D} - \mathbf{A}$, one has that \mathbf{U} 's spectral gap satisfies:*

$$\lambda_2(\mathbf{D}^{-1/2} \mathbf{U} \mathbf{D}^{-1/2}) \geq \frac{\Phi(G)^2}{4}.$$

We refer to the following lemma, which is implicit in [37, 36].

Lemma C.3 (Lemma 31 from [21]). *For an unweighted graph $G = (V, E)$, in $\tilde{O}(m)$ time, we can produce a partition V_1, \dots, V_k of V , and a collection of sets $S_1, \dots, S_k \subseteq V$ with the following properties:*

1. For all i , $S_i \subseteq V_i$.
2. For all i , there exists a set T_i with $S_i \subseteq T_i \subseteq V_i$, such that $\Phi(G(T_i)) \geq \Omega(1/\log^2 n)$.
3. At least half of the edges are found within the sets $\{S_i\}$, i.e.

$$\sum_{i=1}^k |E(S_i)| = \sum_{i=1}^k |\{(a, b) \in E : a, b \in S_i\}| \geq \frac{1}{2} |E|.$$

We use this decomposition lemma in order to first prove the result for unweighted graphs. The more general weighted version will then follow from a bucketing argument.

The decomposition can be produced by iteratively applying Lemma C.3 on the symmetrized input graph \mathbf{U} , choosing $\mathcal{L}^{(i)}$ to be the directed Laplacian induced on vertices in S_i , and $\mathbf{U}^{(i)}$ be the undirected Laplacian induced on vertices in T_i . Since $E[S_i] \subseteq E[T_i]$, clearly $\mathbf{diag}(\mathbf{S}_{\mathcal{L}^{(i)}}) \preceq \mathbf{diag}(\mathbf{U}^{(i)})$.¹¹ The lower bound on the spectral gap for each \mathbf{U}_i is given by the second property of Lemma C.3, combined with Cheeger's inequality (Theorem C.2): the spectral gap of $\mathbf{U}^{(i)}$ is at

¹¹Note that the decomposition lemma gives us a much stronger property than what we are using, since our sparsification routine only requires degree dominance.

least $1/\tilde{O}(1)$. Also, note that the sum of supports of these $\mathbf{U}^{(i)}$ is $O(n)$, since the graphs T_i are vertex-disjoint subgraphs of G , we have $\sum_{i=1}^k \mathbf{U}^{(i)} \preceq \mathbf{U}$.

Removing the graphs induced by S_i (or, equivalently, $\mathcal{L}^{(i)}$), for all i , reduces the number of edges in G by half. Therefore, after $\lceil \log n \rceil$ iterations of applying Lemma C.3, the edges on G will have been exhausted, and we are done. Since each iteration produces undirected Laplacians whose sum is bounded by \mathbf{U} , the sum of all the undirected Laplacians produced during the $\lceil \log n \rceil$ iterations is at most $\lceil \log n \rceil \cdot \mathbf{U}$. Also, the sum of support sizes $O(n \log n)$. Hence we have a $(\tilde{O}(n), 1/\tilde{O}(1), \tilde{O}(1))$ -decomposition of G .

In order to obtain a weighted version of the theorem, we initially decompose the graph G into $b = \lceil \log(w_{\max}/w_{\min}) \rceil$ graphs G_1, \dots, G_b , where w_{\max} and w_{\min} represent the maximum, and minimum arc weight in G , respectively, and $G_j = (V, E_j)$ with $E_j = \{e \in E : w_{\min} \cdot 2^{t-1} \leq w_e < w_{\min} \cdot 2^t\}$. For each of these graphs, the cover corresponding to the unweighted G_j , scaled by $w_{\min} \cdot 2^t$, becomes a $(\tilde{O}(n), 1/\tilde{O}(1), \tilde{O}(1))$ -decomposition of the weighted G_j . Therefore, taking the union of all the decompositions from the b graphs, we obtain a $(\tilde{O}(bn), 1/\tilde{O}(1), \tilde{O}(b))$ -decomposition of G . Since all weights are polynomially bounded, $b = \tilde{O}(1)$, and we obtain the desired result.

In addition, it can be shown that the result from Lemma C.3 can be made parallel using [32]. Indeed, using their SDP-based balanced partitioning routine, one can in $\tilde{O}(1)$ parallel time and $\tilde{O}(m)$ work find a balanced cut with with polylogarithmic (i.e. $1/\tilde{O}(1)$) conductance, or certify that none exists. Such a partitioning routine is then called recursively on the pieces of the input graph that are not yet certified to be expanders, yielding a nearly-linear work, polylogarithmic time algorithm which produces the partition from Lemma C.3. We also refer the reader to Section 6 of [33], for a discussion concerning the parallelization of the decomposition routine.

D The Complete Solver

In this section we provide some details of the full algorithm for computing stationary distributions and solving directed Laplacians. Various applications of these routines are presented in [12] and briefly enumerated in Section 1.2. We provide details on these two routines of computing stationary and solving linear system because they are most important for completing the picture.

We start by stating the stationary computation algorithm given as Algorithm 1 in Section 3.2 of [12]. One difference in presentation is that the routine as shown in [12] relies on solving matrices whose diagonal entries are strictly bigger than the total magnitude of off-diagonal entries in the corresponding row/column. On the other hand, we have only presented a solver for Eulerian Laplacians. The diagonal entries of these matrices are equal to the total magnitude of the off-diagonal entries. As a result, we also need to incorporate the reduction from such matrices to Eulerian Laplacians from Section 5 of [12]. Pseudocode of this routine is in Figure D.1.

This can then be turned into a solver for a strongly connected Laplacian by rescaling it by the stationary, and solve the (approximately) Eulerian Laplacian that result from this. The pseudocode in Figure D.2 is based on Sections 7.1. and 7.3. of [12].

E Approximating the Harmonic Symmetrization

To solve an Eulerian Laplacian system $\mathcal{L}\vec{x} = \vec{b}$, [12] instead solved the system $\mathcal{L}^\top \mathbf{U}_{\mathcal{L}}^+ \mathcal{L} = \mathcal{L}^\top \mathbf{U}_{\mathcal{L}} \vec{b}$. Since $\mathbf{U}_{\mathcal{L}}$ is a symmetric Laplacian, its pseudoinverse can be applied in nearly linear time via a variety of methods [37, 23, 24, 22, 27, 33, 25], and therefore given a linear system solver for $\mathcal{L}^\top \mathbf{U}_{\mathcal{L}}^+ \mathcal{L}$ one is achieved for \mathcal{L} with only a polylogarithmic running time overhead at worst.

$\vec{s} = \text{COMPUTESTATIONARY}(\mathcal{L}, \alpha)$
Input: $n \times n$ directed Laplacian \mathcal{L} , with diagonal \mathbf{D} .
Restart parameter $\alpha \in [0, \frac{1}{2}]$.
Output: approximate stationary distribution \vec{s} .

1. Set $\vec{x}^{(0)} \leftarrow \mathbf{D}^{-1}\mathbf{1}$, $\epsilon \leftarrow \text{poly}(\frac{\alpha}{n})$.
2. For $t = 0, \dots, k = 3 \ln(\alpha^{-1})$
 - (a) Set $\vec{e}^{(t)} \leftarrow \max\{0, \mathcal{L}\vec{x}^{(t)}, \text{diag}(\vec{x}^{(t)})\mathcal{L}\mathbf{1}\}$, and let $\mathbf{E}^{(t)} = \text{diag}(\vec{e}^{(t)})$, $\mathbf{X}^{(t)} = \text{diag}(\vec{x}^{(t)})$ be the corresponding diagonal matrices.
 - (b) Create $\mathcal{L}^{(t+1)} \in \mathbb{R}^{(n+1) \times (n+1)}$ from $(\mathbf{E}^{(t)} + \mathcal{L})\mathbf{X}^{(t)}$ by adding a row/column to make all row/column sums 0.
 - (c) Create a length $n + 1$ vector $\vec{b}^{(t)}$ with sum 0 whose first n entries are given by the vector $\frac{1}{\|\mathbf{D}^{-1}\vec{e}^{(t)}\|_1} \mathbf{D}^{-1}\vec{e}^{(t)}$.
 - (d) Let $\vec{z}^{(t+1)}$ be the first n entries of the vector returned by $\text{SOLVEEULERIAN}(\mathcal{L}^{(t)}, \vec{b}^{(t)}, \epsilon)$, and $\vec{x}^{(t+1)} \leftarrow \mathbf{X}^{(t)}\vec{z}^{(t+1)}$.
3. Return $\vec{s} = \frac{\mathbf{D}\vec{x}^{(k+1)}}{\|\mathbf{D}\vec{x}^{(k+1)}\|_1}$.

Figure D.1: Stationary Computational Algorithm. This routine combines the reduction from solving strictly row/column dominant matrices to Eulerian Laplacians from Section 5 of [12] with the stationary finding algorithm from Section 3.

The matrix $\mathcal{L}^\top \mathbf{U}_\mathcal{L}^+ \mathcal{L}$ was referred to in [12] as the *harmonic symmetrization* of \mathcal{L} and it was shown that linear systems in this harmonic symmetrization can be solved in $O((nm^{3/4} + n^{2/3}m) \log^{O(1)}(n\kappa))$ time. Here we show that if $\tilde{\mathbf{A}}$ is a strong approximation for \mathbf{A} , then $\tilde{\mathbf{A}}^\top \mathbf{U}_\mathbf{A}^+ \tilde{\mathbf{A}}$ is a spectral approximation for $\mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A}$. Consequently, by simply producing a sparsifier for \mathcal{L} in nearly linear time using the results of Section 3, solving the harmonic symmetrization using [12], and then using this solver as a preconditioner in the standard symmetric sense [34], yields an $O(m + n^{7/3} \log^{O(1)}(n\kappa))$ time algorithm for solving directed Laplacian systems.

Lemma E.1. *If $\tilde{\mathbf{A}}$ is an ϵ -strong-approximation of \mathbf{A} for $\epsilon \in (0, 1/2)$ and $\ker(\mathbf{U}_\mathbf{A}) \subseteq \ker(\mathbf{A})$, then*

$$(1 - 2\epsilon)^2 \mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A} \preceq \tilde{\mathbf{A}}^\top \mathbf{U}_\mathbf{A}^+ \tilde{\mathbf{A}} \preceq (1 + \epsilon)^3 \mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A}.$$

Proof. Let $\mathbf{M} = \mathbf{U}_\mathbf{A}^{+/2} \tilde{\mathbf{A}} \mathbf{U}_\mathbf{A}^{+/2}$ and $\mathbf{N} = \mathbf{U}_\mathbf{A}^{+/2} \mathbf{A} \mathbf{U}_\mathbf{A}^{+/2}$. The definition of strong approximation implies that $\|\mathbf{M} - \mathbf{N}\|_2 \leq \epsilon$. Applying Lemma B.1, a general lemma regarding the difference of matrices, yields that

$$(1 - \epsilon)\mathbf{N}^\top \mathbf{N} - \epsilon^{-1}\epsilon^2 \mathbf{I} \preceq \mathbf{M}^\top \mathbf{M} \preceq (1 + \epsilon)\mathbf{N}^\top \mathbf{N} + (1 + \epsilon^{-1})\epsilon^2 \mathbf{I}.$$

Now let $\vec{x} \in \mathbb{R}^n$ be an arbitrary vector perpendicular to the kernel of $\mathbf{U}_\mathbf{A}$. For such a vector \vec{x} , we have

$$\vec{x}^\top \mathbf{N}^\top \mathbf{N} \vec{x} = \vec{x}^\top \mathbf{U}_\mathbf{A}^{+/2} \mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A} \mathbf{U}_\mathbf{A}^{+/2} \vec{x} \geq \vec{x}^\top \mathbf{U}_\mathbf{A}^{+/2} \mathbf{U}_\mathbf{A} \mathbf{U}_\mathbf{A}^{+/2} \vec{x} \geq \vec{x}^\top \mathbf{I} \vec{x},$$

$\vec{s} = \text{SOLVEFULL}(\mathcal{L}, \alpha)$ Input: $n \times n$ directed Laplacian \mathcal{L} , with diagonal \mathbf{D} , desired error ϵ . vector \vec{b} Output: approximate solution to $\mathcal{L}\vec{x} = \vec{b}$.
<ol style="list-style-type: none"> 1. Estimate the mixing time of \mathcal{L}, T_{mix} by binary searching on the mixing times of $\mathcal{L} + \alpha\mathbf{I}$. 2. Add $\frac{\epsilon}{T_{\min\text{poly}}(n)}$ to \mathcal{L} to get a strictly diagonally dominant matrix $\widehat{\mathcal{L}}$. 3. Compute approximate stationary distribution of $\widehat{\mathcal{L}}$, \vec{s}. 4. Let $\vec{x} \leftarrow \text{SOLVEEULERIAN}(\widehat{\mathcal{L}}\mathbf{D}^{-1}\mathbf{S}, \vec{b}, \epsilon/\text{poly}(n))$. 5. Return $\mathbf{D}^{-1}\mathbf{S}\vec{x}$.

Figure D.2: Full solver algorithm for strongly connected Laplacians. It first perturbs \mathcal{L} to form a matrix where a good stationary distribution is easy to compute, and uses the normalization given by it to reduce the problem solving on an Eulerian Laplacian.

where we used that by Lemma B.9 the harmonic symmetrization spectrally dominates the symmetric Laplacian, i.e., $\mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A} \succeq \mathbf{U}_\mathbf{A}$. Consequently, for $\mathbf{C} = \mathbf{A}^\top \mathbf{U}_\mathbf{A}^+ \mathbf{A}$ and $\widetilde{\mathbf{C}} = \widetilde{\mathbf{A}}^\top \mathbf{U}_\mathbf{A}^+ \widetilde{\mathbf{A}}$ we have

$$(1 - 2\epsilon)\vec{x}^\top \mathbf{U}_\mathbf{A}^{+/2} \mathbf{C} \mathbf{U}_\mathbf{A}^{+/2} \vec{x} \preceq \vec{x}^\top \mathbf{U}_\mathbf{A}^{+/2} \widetilde{\mathbf{C}} \mathbf{U}_\mathbf{A}^{+/2} \vec{x} \preceq (1 + 2\epsilon + \epsilon^2)\vec{x}^\top \mathbf{U}_\mathbf{A}^{+/2} \mathbf{C} \mathbf{U}_\mathbf{A}^{+/2} \vec{x}.$$

One can easily see that when $x \in \ker(\mathbf{U}_\mathbf{A})$ all of the terms are 0. Hence, we have that the above holds for all x . Since $\ker(\mathbf{U}_\mathbf{A}) \subseteq \ker(\mathbf{A})$ this in turn implies $(1 - 2\epsilon)\mathbf{C} \preceq \widetilde{\mathbf{C}} \preceq (1 + \epsilon)^2\mathbf{C}$. Furthermore, since $(1 - \epsilon)\mathbf{U}_\mathbf{A} \preceq \mathbf{U}_\mathbf{A}^- \preceq (1 + \epsilon)\mathbf{U}_\mathbf{A}$ by Lemma 3.6, we also have $(1 - \epsilon)\mathbf{U}_\mathbf{A}^+ \preceq \mathbf{U}_\mathbf{A}^+ \preceq (1 + \epsilon)\mathbf{U}_\mathbf{A}^+$. The result follows from combining and using $(1 - 2\epsilon) \leq (1 - \epsilon)$. \square

F Reducing the Condition Number

In this section, we present a reduction from the problem of (approximately) solving Eulerian Laplacians to solving Eulerian Laplacians that are at most polynomially ill-conditioned, with a logarithmic dependence on condition number. This reduces the overall dependency on κ to logarithmic instead of sub-polynomial. The main result of this section is:

Theorem F.1. *There exists a procedure CRUDE SOLVE ILL CONDITIONED which, when given an $n \times n$ Eulerian Laplacian \mathcal{L} with m non-zeros such that the condition number of $\mathcal{L} + \mathcal{L}^\top$ is bounded by κ , returns a crude approximate solution x to $\mathcal{L}x = b$ in the sense that*

$$\|\vec{x} - \mathcal{L}^+ \vec{b}\|_{\mathbf{U}_\mathcal{L}} \leq \frac{1}{2} \|\mathcal{L}^+ \vec{b}\|_{\mathbf{U}_\mathcal{L}}.$$

This procedure performs only $O(\log(n\kappa))$ calls to an approximate Eulerian Laplacian solver, each on $O(n)$ vertices with $O(m)$ nonzero entries, with error parameter $\frac{1}{n^{O(1)}}$ and each with condition number $n^{O(1)}$, plus $O(m \log(n\kappa))$ additional work.

Furthermore, if the Eulerian solver is an implicit polynomial, the overall procedure is an implicit polynomial as well.

Combining this routine with iterative refinement / preconditioned Richardson iteration as stated in Lemma 4.2 implies that one can obtain an ϵ -approximate solution with $O(\log(n\kappa) \log(1/\epsilon))$ such solver calls and $O(m \log(n\kappa) \log(1/\epsilon))$ additional work. We note that this construction can also be used to reduce the $\log \kappa$ dependencies in [33] to $\log n$.

We include this construction here because it removes the $\exp(\sqrt{\log \kappa})$ dependencies in our algorithms and replaces them with similar terms in n . Since problems on directed graphs can be highly ill-conditioned even when the edge weights are all small integers, this can potentially be a significant improvement.

Remark. We believe that developing combinatorial techniques to mitigate the effects of ill-conditioning is an important endeavor in both theory and practice, and that the scheme we include reflects only partial progress. While the technique we describe is sufficient to improve our running time bounds, we conjecture that far better reduction schemes are possible. Some ways in which an ideal such result could improve on the one presented here include:

1. The overhead in work and depth could be as low as $O(1)$: the different scales would only have minimal overlap, and processing could occur in parallel.
2. The scheme could only need to manipulate numbers with double or quadruple the precision of n/ϵ , instead of involving words whose sizes are $O(1)$ bigger. Treating these increases with similar emphasis as constants in approximation algorithms may be a more realistic model of the costs of floating point arithmetic.
3. As highly ill conditioned systems often arise under floating point arithmetic (due to the exponent) such reductions should ideally be robust to floating instead of fixed point arithmetic.

Systematically developing strong versions of such “condition number reducing reductions” is potentially a challenging but important direction for future work.

The main idea of our reduction is to collar the Laplacian to a fixed “scale” of edge weights, contracting edges above the scale while adding a smaller multiple of a clique to bound the lower eigenvalue. The algorithm then uses the Laplacian at a given scale to route demand between vertices connected at about that scale. Our algorithm is defined around the following contraction and projection operators:

Definition F.2. Given a partition of $[n]$ into k sets S_1, S_2, \dots, S_k ,

1. the contraction operator \mathbf{C} is the linear map $\mathbb{R}^n \rightarrow \mathbb{R}^k$ mapping $\vec{1}_i$ to $\vec{1}_j$ if $i \in S_j$.
2. let Proj be the orthogonal projection onto the kernel of \mathbf{C} .

Pseudocode based on these operators is in Figure F.1.

Our proofs crucially rely on the following fact that relates the ordinary (arithmetic) and harmonic symmetrizations of Eulerian Laplacians. It is immediate from Lemma 13 of [12], and one side of it is also present as Lemma B.9.

Lemma F.3. *Let \mathcal{L} be an Eulerian Laplacian and $\mathbf{U}_{\mathcal{L}}$ its symmetrization $\frac{\mathcal{L} + \mathcal{L}^{\top}}{2}$. Then*

$$\mathbf{U}_{\mathcal{L}} \preceq \mathcal{L}^{\top} \mathbf{U}_{\mathcal{L}} \mathcal{L} \preceq 2(n-1)^2 \mathbf{U}_{\mathcal{L}}.$$

We also need a simple technical lemma about the extreme values in solutions to Eulerian Laplacian systems in terms of the support of the demand vector.

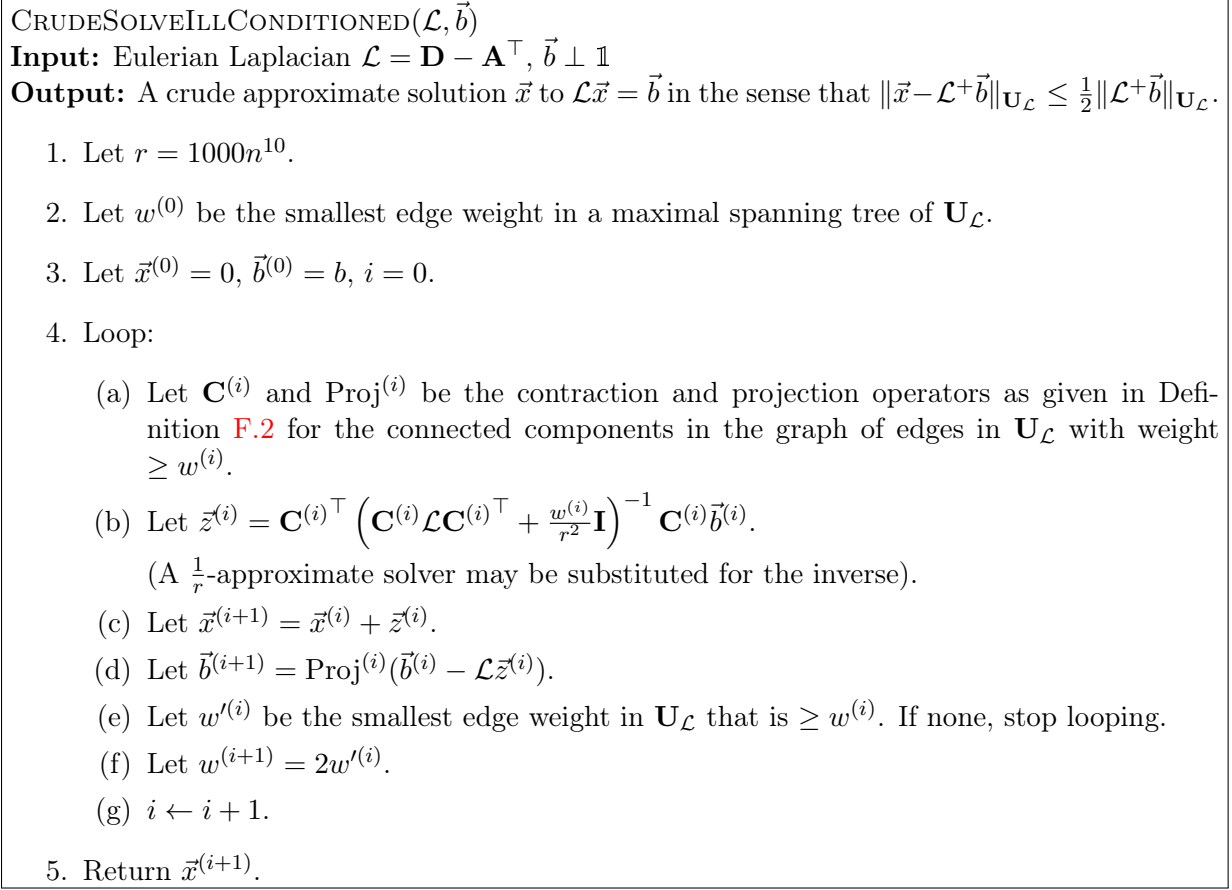


Figure F.1: Reduction to solving well-conditioned Eulerian systems

Lemma F.4. *Let \mathcal{L} be a connected Eulerian Laplacian and \vec{x} and \vec{b} be nonzero vectors such that $\mathcal{L}\vec{x} = \vec{b}$. Then the maximum value of the entries of \vec{x} , as well as the minimum value, must be attained on $\text{sup } \vec{b}$.*

Proof. Consider the set S of all entries of \vec{b} attaining the maximum value v . If this set contains every vertex, it automatically overlaps $\text{sup } \vec{b}$. Otherwise, we have:

$$\sum_{i \in S} \vec{b}_i = \left(\sum_{i \in S, j \notin S} w_{ji} \vec{x}_j \right) - \left(\sum_{i \in S, j \notin S} w_{ij} \vec{x}_i \right) < v \left(\sum_{i \in S, j \notin S} w_{ji} \right) - v \left(\sum_{i \in S, j \notin S} w_{ij} \right) = 0.$$

Here the last equality is due to \mathcal{L} being Eulerian: the total weight entering and leaving S is equal. The sum of the entries from S in \vec{b} is strictly negative and thus nonzero, so S must overlap with the support of \vec{b} .

The case of the minimum value is analogous. □

Next, we show that if the demands for an Eulerian Laplacian system are supported on a well-connected subset of the graph, perturbing the system by a small multiple of the identity matrix cannot induce too much error.

Lemma F.5. *Let \mathcal{L} be a connected Eulerian Laplacian with corresponding undirected Laplacian $\mathbf{U}_{\mathcal{L}}$, and let S be a subset of the vertices such that any two vertices in S are connected by a path in $\mathbf{U}_{\mathcal{L}}$ containing only edges of weight at least β . Let \vec{b} be a vector in the image of \mathcal{L} supported on S , and let $\alpha > 0$. Then*

$$\left\| (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{b} - \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}} \leq n \sqrt{\frac{\alpha}{\beta}} \left\| \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}}.$$

Proof. We define $\vec{x} = \mathcal{L}^+ \vec{b}$ and $\vec{y} = (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{b}$. Writing \vec{y} as $\vec{x} + (\mathcal{L} + \alpha \mathbf{I})^{-1} (\vec{b} - (\mathcal{L} + \alpha \mathbf{I}) \vec{x})$ gives:

$$\vec{y} - \vec{x} = (\mathcal{L} + \alpha \mathbf{I})^{-1} (\mathcal{L} \vec{x} - (\mathcal{L} + \alpha \mathbf{I}) \vec{x}) = -\alpha (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{x}.$$

which when substituted into the $\mu_{\mathcal{L}}$ norm gives:

$$\|\vec{y} - \vec{x}\|_{\mathbf{U}_{\mathcal{L}}}^2 = \alpha^2 \vec{x}^\top \left(\mathcal{L}^\top + \alpha \mathbf{I} \right)^{-1} \mathbf{U}_{\mathcal{L}} (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{x}.$$

Since $\mathbf{U}_{\mathcal{L}} \preceq \mathbf{U}_{\mathcal{L}} + \alpha \mathbf{I}$, we can invoke Lemma F.3 with the matrix $(\mathcal{L} + \alpha \mathbf{I})$ to get:

$$\|\vec{y} - \vec{x}\|_{\mathbf{U}_{\mathcal{L}}}^2 \leq \frac{\alpha^2}{\alpha} \vec{x}^\top \vec{x} = n \alpha \|\vec{x}\|_{\infty}^2.$$

Now, by Lemma F.4, the full range of the entries of \vec{x} occurs within S . The existence of a path with at most n vertices connecting the minimum and maximum of these entries implies that

$$\|\vec{x}\|_{\mathbf{U}_{\mathcal{L}}}^2 \geq \frac{\beta}{n} \|\vec{x}\|_{\infty}^2.$$

Putting this together we get

$$\|\vec{y} - \vec{x}\|_{\mathbf{U}_{\mathcal{L}}}^2 \leq n^2 \frac{\alpha}{\beta} \|\vec{x}\|_{\mathbf{U}_{\mathcal{L}}}^2.$$

□

Next, we handle the case of simultaneous demands within multiple well-connected components. We also switch the error bound to be in $\|\vec{b}\|_{\mathbf{U}_{\mathcal{L}^+}}$ to facilitate our later steps.

Lemma F.6. *Let \mathcal{L} be a connected Eulerian Laplacian with corresponding undirected Laplacian $\mathbf{U}_{\mathcal{L}}$, and let S_1, S_2, \dots, S_k be the connected components of the graph consisting of those edges in $\mathbf{U}_{\mathcal{L}}$ with edges of weight at least β . Let \vec{b} be a vector such that $\sum_{i \in S_j} \vec{b}_i = 0$ for all j , and let $\alpha > 0$. Then*

$$\left\| (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{b} - \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}} \leq 2n^{7/2} \sqrt{\frac{\alpha}{\beta}} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Proof. We decompose \vec{b} as

$$\vec{b} = \sum_j \vec{b}_j$$

where \vec{b}_j is supported on S_j , and 0 everywhere else.

Now we aim to bound $\|\vec{b}_j\|_{\mathbf{U}_{\mathcal{L}^+}}$. Note that there is an electrical flow \vec{y} on $\mathbf{U}_{\mathcal{L}}$ with energy $\|\vec{b}\|_{\mathbf{U}_{\mathcal{L}^+}}^2$ routing overall demands \vec{b} . We define \vec{y}_j as the restriction of the flow to the internal edges of S_j , and let its residuals be \vec{b}'_j . Since this flow is on a subset of the edges, its total energy is almost most $\|\vec{b}\|_{\mathbf{U}_{\mathcal{L}^+}}^2$ so this certifies that $\|\vec{b}'_j\|_{\mathbf{U}_{\mathcal{L}^+}} \leq \|\vec{b}\|_{\mathbf{U}_{\mathcal{L}^+}}$. Then

$$\left\| \vec{b}'_j - \vec{b}_j \right\|_1 \leq \frac{n}{\sqrt{\beta}} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}$$

since it is at most the ℓ_1 norm of the flows on the edges incident to but not contained in S_j in y , and each of these $\leq n^2$ edges has weight at most β by the assumption of S_j being the connected components on edges with weights $\geq \beta$. $\vec{b}'_j - \vec{b}_j$ is also supported on S_j ; since S_j is connected by edges of weight $\geq \beta$, we have

$$\left\| \vec{b}'_j - \vec{b}_j \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq \sqrt{n\beta} \left\| \vec{b}'_j - \vec{b}_j \right\|_1 \leq n^{3/2} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Then by the triangle inequality

$$\left\| \vec{b}_j \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq \left\| \vec{b}'_j \right\|_{\mathbf{U}_{\mathcal{L}^+}} + \left\| \vec{b}'_j - \vec{b}_j \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq 2n^{3/2} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Lemma B.9 then implies that $\left\| \mathcal{L}^+ \vec{b}_j \right\|_{\mathbf{U}} \leq \left\| \vec{b}_j \right\|_{\mathbf{U}_{\mathcal{L}^+}}$. Finally, we apply Lemma F.5 to each \vec{b}_j , yielding that

$$\left\| (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{b}_j - \mathcal{L}^+ \vec{b}_j \right\|_{\mathbf{U}_{\mathcal{L}}} \leq 2n^{5/2} \sqrt{\frac{\alpha}{\beta}} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Summing over the up to n different \vec{b}_j and applying the triangle inequality over this sum gives

$$\left\| (\mathcal{L} + \alpha \mathbf{I})^{-1} \vec{b} - \mathcal{L}^+ \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}}} \leq 2n^{7/2} \sqrt{\frac{\alpha}{\beta}} \left\| \vec{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}}$$

as desired. \square

We now have the tools to analyze CRUDESOLVEILLCONDITIONED. Our analyses rely on the following key intermediate variables:

1.

$$\vec{q}^{(i)} \stackrel{\text{def}}{=} \mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)} \mathcal{L} \mathbf{C}^{(i)\top} \right)^+ \mathbf{C}^{(i)} \vec{b}^{(i)}.$$

2.

$$\vec{e}^{(i)} \stackrel{\text{def}}{=} \vec{z}^{(i)} - \vec{q}^{(i)},$$

where $\vec{z}^{(i)}$ is the ‘shifted’ solution obtained on Step 4b.

3.

$$\vec{f}^{(i)} \stackrel{\text{def}}{=} \text{Proj}^{(i)} \left(\mathcal{L} \vec{e}^{(i)} \right).$$

4.

$$\vec{b}^{*(i)} \stackrel{\text{def}}{=} \vec{b} - \sum_{j < i} \vec{f}^{(j)}.$$

We first show that the right hand side in the iterations can be expressed as a close form involving the errors.

Lemma F.7. *For all i ,*

$$\vec{b}^{(i)} = \left(\mathbf{I} - \mathcal{L} \mathbf{C}^{(i-1)\top} \left(\mathbf{C}^{(i-1)} \mathcal{L} \mathbf{C}^{(i-1)\top} \right)^+ \mathbf{C}^{(i-1)} \right) \vec{b}^{*(i)}.$$

Proof. First we note two equations about the projection operator $\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i-1)\top} \left(\mathbf{C}^{(i-1)}\mathcal{L}\mathbf{C}^{(i-1)\top}\right)^+ \mathbf{C}^{(i-1)}\right)$.

Now we prove this by induction. The base case of $i = 0$ follows from the two sides being identical. For the inductive case, substituting in the construction of $\vec{b}^{(i+1)}$ on Step 4d gives:

$$\begin{aligned} \vec{b}^{(i+1)} &= \text{Proj}^{(i)}(\vec{b}^{(i)} - \mathcal{L}\vec{z}^{(i)}) \\ &= \text{Proj}^{(i)}\left(\vec{b}^{(i)} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\vec{b}^{(i)} - \mathcal{L}\vec{e}^{(i)}\right) \\ &= \text{Proj}^{(i)}\left(\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{b}^{(i)} - \text{Proj}^{(i)}(\mathcal{L}\vec{e}^{(i)})\right) \\ &= \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{b}^{(i)} - \vec{f}^{(i)}. \end{aligned}$$

Here, the last line follows from the fact that $\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{b}^{(i)}$ is already in the kernel of $\mathbf{C}^{(i)}$, as

$$\begin{aligned} \mathbf{C}^{(i)}\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{b}^{(i)} &= \mathbf{C}^{(i)}\vec{b}^{(i)} - \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)\left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\vec{b}^{(i)} \\ &= \mathbf{C}^{(i)}\vec{b}^{(i)} - \mathbf{C}^{(i)}\vec{b}^{(i)} \\ &= 0 \end{aligned}$$

We similarly have

$$\begin{aligned} \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\mathcal{L}\mathbf{C}^{(i)\top} &= \mathcal{L}\mathbf{C}^{(i)\top} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right) \\ &= \mathcal{L}\mathbf{C}^{(i)\top} - \mathcal{L}\mathbf{C}^{(i)\top} \\ &= 0 \end{aligned}$$

Since the image of $\mathbf{C}^{(i-1)\top}$ is contained in the image of $\mathbf{C}^{(i)\top}$, this also implies that

$$\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\mathcal{L}\mathbf{C}^{(i-1)\top} = 0$$

and hence that

$$\begin{aligned} \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i-1)\top} \left(\mathbf{C}^{(i-1)}\mathcal{L}\mathbf{C}^{(i-1)\top}\right)^+ \mathbf{C}^{(i-1)}\right) \\ = \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right). \end{aligned}$$

We also have $\left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{f}^{(i)} = \vec{f}^{(i)}$ as $\vec{f}^{(i)}$, output by $\text{Proj}^{(i)}$, is in the kernel of $\mathbf{C}^{(i)}$. Putting these together and substituting in the induction hypothesis on i gives

$$\begin{aligned} \vec{b}^{(i+1)} &= \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\left(\vec{b}^{*(i)} - \vec{f}^{(i)}\right) \\ &= \left(\mathbf{I} - \mathcal{L}\mathbf{C}^{(i)\top} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top}\right)^+ \mathbf{C}^{(i)}\right)\vec{b}^{*(i+1)} \end{aligned}$$

which shows that the identity holds for $i + 1$ as well. \square

Lemma F.8. For all i ,

$$\vec{x}^{(i)} + \mathcal{L}^+ \vec{b}^{(i)} = \sum_{j < i} \vec{e}^{(j)} + \mathcal{L}^+ \vec{b}^{*(i)}.$$

Proof. Again, we proceed by induction.

$$\begin{aligned} \vec{x}^{(i+1)} + \mathcal{L}^+ \vec{b}^{(i+1)} &= \vec{x}^{(i)} + \vec{z}^{(i)} + \mathcal{L}^+ \text{Proj}^{(i)} \left(\vec{b}^{(i)} - \mathcal{L} \vec{z}^{(i)} \right) \\ &= \vec{x}^{(i)} + \vec{q}^{(i)} + \vec{e}^{(i)} + \mathcal{L}^+ \text{Proj}^{(i)} \left(\vec{b}^{(i)} - \mathcal{L} \vec{q}^{(i)} - \mathcal{L} \vec{e}^{(i)} \right) \\ &= \vec{x}^{(i)} + \vec{q}^{(i)} + \vec{e}^{(i)} + \mathcal{L}^+ \text{Proj}^{(i)} \left(\vec{b}^{(i)} - \mathcal{L} \vec{q}^{(i)} \right) - \mathcal{L}^+ \text{Proj}^{(i)} \left(\mathcal{L} \vec{e}^{(i)} \right) \\ &= \vec{x}^{(i)} + \vec{q}^{(i)} + \vec{e}^{(i)} + \mathcal{L}^+ \vec{b}^{(i)} - \vec{q}^{(i)} - \mathcal{L}^+ \vec{f}^{(i)}. \end{aligned}$$

The last line here follows from the fact that $\vec{b}^{(i)} - \mathcal{L} \vec{q}^{(i)}$ is already inside the kernel of $\mathbf{C}^{(i)}$. Cancelling the $\vec{q}^{(i)}$ terms and applying the induction hypothesis then gives

$$\begin{aligned} \vec{x}^{(i+1)} + \mathcal{L}^+ \vec{b}^{(i+1)} &= \vec{x}^{(i)} + \mathcal{L}^+ \vec{b}^{(i)} + \vec{e}^{(i)} - \mathcal{L}^+ \vec{f}^{(i)} \\ &= \mathcal{L}^+ \vec{b}^{*(i)} + \sum_{j < i} \vec{e}^{(j)} + \vec{e}^{(i)} - \mathcal{L}^+ \vec{f}^{(i)} \\ &= \mathcal{L}^+ \vec{b}^{*(i+1)} + \sum_{j < i+1} \vec{e}^{(j)}. \end{aligned}$$

□

These relations then allows us to bound the global error via the guarantees of the separate approximate solves. As these solves produce solutions with relative error, we first need to bound the norm of $\vec{b}^{(i)}$:

Lemma F.9. For all i ,

$$\left\| \vec{b}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq 3n \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Proof. First, note that

$$\left\| \mathbf{C}^{(i-1)} \vec{b}^{*(i)} \right\|_{(\mathbf{C}^{(i-1)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i-1)\top})^+} \leq \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Lemma F.3 then gives

$$\left\| \left(\mathbf{C}^{(i-1)} \mathcal{L} \mathbf{C}^{(i-1)\top} \right)^+ \mathbf{C}^{(i-1)} \vec{b}^{*(i)} \right\|_{(\mathbf{C}^{(i-1)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i-1)\top})} \leq \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}$$

or equivalently

$$\left\| \mathbf{C}^{(i-1)\top} \left(\mathbf{C}^{(i-1)} \mathcal{L} \mathbf{C}^{(i-1)\top} \right)^+ \mathbf{C}^{(i-1)} \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}}} \leq \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Now applying Lemma F.3, we get

$$\left\| \mathcal{L} \mathbf{C}^{(i-1)\top} \left(\mathbf{C}^{(i-1)} \mathcal{L} \mathbf{C}^{(i-1)\top} \right)^+ \mathbf{C}^{(i-1)} \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq 2n \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

The result then follows from the triangle inequality with $\vec{b}^{*(i)}$.

□

For the next step we will use the following fact about matrices:

Fact F.10. For any symmetric positive semidefinite matrix \mathbf{M} and arbitrary matrix \mathbf{C} , and for any vector \vec{v} in the image of \mathbf{M} ,

$$\|\mathbf{C}\vec{v}\|_{(\mathbf{C}\mathbf{M}\mathbf{C}^\top)^+} \leq \|\vec{v}\|_{\mathbf{M}^+}.$$

Notably, this fact is equivalent to the standard result that the Schur complement of a positive semidefinite matrix is spectrally dominated by the matrix.

Proof. We can prove this using duality of norms:

$$\begin{aligned} \|\mathbf{C}\vec{v}\|_{(\mathbf{C}\mathbf{M}\mathbf{C}^\top)^+} &= \max_{\|\vec{u}\|_{(\mathbf{C}\mathbf{M}\mathbf{C}^\top)^+} \leq 1} \langle \vec{u}, \mathbf{C}\vec{v} \rangle \\ &= \max_{\|\mathbf{C}^\top \vec{u}\|_{\mathbf{M}^+} \leq 1} \langle \mathbf{C}^\top \vec{u}, \vec{v} \rangle \\ &\leq \max_{\|\vec{u}'\|_{\mathbf{M}^+} \leq 1} \langle \vec{u}', \vec{v} \rangle \\ &= \|\vec{v}\|_{\mathbf{M}^+}. \end{aligned}$$

□

We begin to bound the error terms:

Lemma F.11.

$$\|\vec{e}^{(i)}\|_{\mathbf{U}_\mathcal{L}} \leq \frac{12n^{9/2}}{r} \|\vec{b}^{*(i)}\|_{\mathbf{U}_\mathcal{L}^+}.$$

Proof. First, we note that by Fact F.10

$$\|\mathbf{C}^{(i)}\vec{b}^{(i)}\|_{(\mathbf{C}^{(i)}\mathbf{U}_\mathcal{L}\mathbf{C}^{(i)\top})^+} \leq \|\vec{b}^{(i)}\|_{\mathbf{U}_\mathcal{L}^+} \leq 3n \|\vec{b}^{*(i)}\|_{\mathbf{U}_\mathcal{L}^+},$$

where the last inequality is by Lemma F.9. Now, define intermediate variables:

1.

$$\vec{q}^{(i)} \stackrel{\text{def}}{=} (\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top})^+ \mathbf{C}^{(i)}\vec{b}^{(i)},$$

2.

$$\vec{z}^{h(i)} \stackrel{\text{def}}{=} \left(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2}\mathbf{I} \right)^{-1} \mathbf{C}^{(i)}\vec{b}^{(i)},$$

and $\vec{z}^{j(i)}$ analogously, but as the output of a $\frac{1}{r}$ -approximate solver for the system $(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2}\mathbf{I})\vec{x} = \mathbf{C}^{(i)}\vec{b}^{(i)}$.

By rearranging and applying the triangle inequality, we have

$$\|\vec{e}^{(i)}\|_{\mathbf{U}_\mathcal{L}} \leq \|\vec{z}^{j(i)} - \vec{z}^{h(i)}\|_{(\mathbf{C}^{(i)}\mathbf{U}_\mathcal{L}\mathbf{C}^{(i)\top})^+} + \|\vec{z}^{h(i)} - \vec{q}^{(i)}\|_{(\mathbf{C}^{(i)}\mathbf{U}_\mathcal{L}\mathbf{C}^{(i)\top})}$$

The first term captures the error induced by using the approximate rather than exact solver, while the second captures the error induced from adding the multiple of the identity (the more serious issue).

For the first term, we have

$$\begin{aligned}
\left\| \vec{z}^{\prime(i)} - \vec{z}^{\prime\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} \right)} &\leq \left\| \vec{z}^{\prime(i)} - \vec{z}^{\prime\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2} \mathbf{I} \right)} \\
&\leq \frac{1}{r} \left\| \vec{z}^{\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2} \mathbf{I} \right)} \quad (\text{by definition of approximate solver}) \\
&\leq \frac{1}{r} \left\| \mathbf{C}^{(i)} \vec{b}^{\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2} \mathbf{I} \right)^{-1}} \quad (\text{by Lemma F.3}) \\
&\leq \frac{1}{r} \left\| \mathbf{C}^{(i)} \vec{b}^{\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} \right)^+} \\
&\leq \frac{3n}{r} \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}}^+}.
\end{aligned}$$

For the second term, we will apply Lemma F.6. We will use the fact that $\vec{b}^{\prime(i)}$ is in the image of $\mathbf{C}^{(i-1)}$ —or equivalently, that its entries on any connected component of the edges in $\mathbf{U}_{\mathcal{L}}$ with weight $\geq w^{(i-1)}$ sum to 0. By the definition of $w^{(i)}$, these are the same as the edges with weight $\geq \frac{w^{(i)}}{2}$. Furthermore, contracting can only increase the connectivity of a component, so $\mathbf{C}^{(i)} \vec{b}^{\prime(i)}$ satisfies the same property relative to $\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} \right)$. Then we can apply Lemma F.6 with $\alpha = \frac{w^{(i)}}{r^2}$ and $\beta = \frac{w^{(i)}}{2}$:

$$\left\| \vec{z}^{\prime(i)} - \vec{q}^{\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} \right)} \leq \frac{3n^{7/2}}{r} \left\| \mathbf{C}^{(i)} \vec{b}^{\prime(i)} \right\|_{\left(\mathbf{C}^{(i)} \mathbf{U}_{\mathcal{L}} \mathbf{C}^{(i)\top} \right)^+} \leq \frac{9n^{9/2}}{r} \left\| \vec{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}}^+}.$$

Summing these two bounds gives the desired result. \square

It remains to bound the norms of the other error vectors $\vec{f}^{\prime(i)}$.

Lemma F.12. *For all i ,*

$$\left\| \vec{f}^{\prime(i)} \right\|_{\mathbf{U}_{\mathcal{L}}^+} \leq 6n^{5/2} \left\| \vec{e}^{\prime(i)} \right\|_{\mathbf{U}_{\mathcal{L}}}.$$

Proof. First, we apply Lemma F.3, showing that $\|\mathcal{L}\vec{e}^{\prime(i)}\|_{\mathbf{U}_{\mathcal{L}}^+} \leq 2n\|\vec{e}^{\prime(i)}\|_{\mathbf{U}_{\mathcal{L}}}$.

Now we will show that the $\text{Proj}^{(i)}$ operator cannot increase the $\mathbf{U}_{\mathcal{L}}^+$ norm by more than a factor of $2n^{3/2}$.

The proof is similar to that of Lemma F.6: we consider the electrical flow $\mathbf{U}_{\mathcal{L}}$ that meets the demands $\mathcal{L}\vec{e}^{\prime(i)}$. We denote this flow with $\vec{y}^{\prime(i)}$, and define $\vec{y}^{\prime(i) \prime}$ as the restriction of $\vec{y}^{\prime(i)}$ to the edges of weight $\geq w^{(i)}$. We then let the residue of this flow be $\vec{b}^{\prime(i) \prime}$, and write:

$$\text{Proj}^{(i)} \left(\mathcal{L}\vec{e}^{\prime(i)} \right) = \text{Proj}^{(i)} \left(\vec{b}^{\prime(i) \prime} \right) + \text{Proj}^{(i)} \left(\mathcal{L}\vec{e}^{\prime(i)} - \vec{b}^{\prime(i) \prime} \right).$$

Since $\vec{b}^{\prime(i) \prime}$ is induced by a flow $\vec{y}^{\prime(i) \prime}$ wholly within the components with weights $\geq w^{(i)}$,

$$\text{Proj}^{(i)} \left(\vec{b}^{\prime(i) \prime} \right) = \vec{b}^{\prime(i) \prime}.$$

Furthermore, since $\vec{y}^{\prime(i) \prime}$ rounds the demand $\vec{b}^{\prime(i) \prime}$, and is a restriction of \vec{y} , we have

$$\left\| \vec{b}^{\prime(i) \prime} \right\|_{\mathbf{U}_{\mathcal{L}}^+}^2 \leq \mathcal{E}_{\mathbf{U}_{\mathcal{L}}} \left(\vec{y}^{\prime(i) \prime} \right) \leq \mathcal{E}_{\mathbf{U}_{\mathcal{L}}} \left(\vec{y}^{\prime(i)} \right) \leq \left\| \mathcal{L}\vec{e}^{\prime(i)} \right\|_{\mathbf{U}_{\mathcal{L}}^+}^2,$$

where $\mathcal{E}_{\mathbf{U}_{\mathcal{L}}}(\vec{y})$ denotes the electrical energy of the flow \vec{y} on $\mu_{\mathcal{L}}$.

On the other hand, $\mathcal{L}\bar{e}^{(i)} - \bar{b}^{(i)'}$ is the residual of the flow $\bar{y}^{(i)} - \bar{y}^{(i)'}$, which is supported on edges with weight $< w^{(i)}$ and also has energy at most $\|\mathcal{L}\bar{e}^{(i)}\|_{\mathbf{U}_{\mathcal{L}^+}}^2$. Since each edge can contribute to the residual of at most two vertices, we have

$$\left\| \mathcal{L}\bar{e}^{(i)} - \bar{b}^{(i)'} \right\|_1 \leq 2 \left\| \bar{y}^{(i)} - \bar{y}^{(i)'} \right\|_1 \leq 2n \left\| \bar{y}^{(i)} - \bar{y}^{(i)'} \right\|_2 \leq \frac{2n}{\sqrt{w^{(i)}}} \left\| \mathcal{L}\bar{e}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Finally, using the fact that $\text{Proj}^{(i)}$ can at most double the ℓ_1 norm of its input and that the $\mathbf{U}_{\mathcal{L}^+}$ norm of demands connected by edges of weight at least $w^{(i)}$ is at most $\sqrt{nw^{(i)}}$ times the ℓ_1 norm of those demands, we have

$$\left\| \text{Proj}^{(i)} \left(\mathcal{L}\bar{e}^{(i)} - \bar{b}^{(i)'} \right) \right\|_{\mathbf{U}_{\mathcal{L}^+}}^+ \leq 4n^{3/2} \left\| \mathcal{L}\bar{e}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Applying the triangle inequality to $\bar{b}^{(i)'}$ and $\mathcal{L}\bar{e}^{(i)} - \bar{b}^{(i)'}$ then gives the desired result. \square

Note that combining the previous two lemmas shows that

$$\left\| \bar{f}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}} \leq \frac{72n^7}{r} \left\| \bar{b}^{*(i)} \right\|_{\mathbf{U}_{\mathcal{L}^+}}.$$

Putting these together with the breakdown of errors then gives the overall guarantees.

Proof of Theorem F.1. First, note that the number of rounds is bounded by $\min\{n^2, O(\log(n\kappa))\}$. The former is from the number of edges, while the latter follows from the fact that the largest and smallest eigenvalues of $\mathbf{U}_{\mathcal{L}}$ are within $\text{poly}(n)$ factors of the smallest and largest weighted vertex degrees.

This then implies by induction that $\|\bar{b}^{*(i)}\|_{\mathbf{U}_{\mathcal{L}^+}} \leq 2\|\bar{b}\|_{\mathbf{U}_{\mathcal{L}^+}}$ for all i – since, assuming it held for all previous i , each of the at most n^2 error terms $\bar{f}^{(j)}$ had norm at most $\frac{1}{5n^3}\|\bar{b}^{*(i)}\|_{\mathbf{U}_{\mathcal{L}^+}}$ (by Lemmas F.11 and F.12). By Lemma F.11 each $\bar{e}^{(i)}$ had norm at most $\frac{1}{40n^{11/2}}$.

Then applying Lemma F.8 on the final configuration (where $\bar{b}^{(i+1)} = 0$) with these bounds (and again using the fact that there are most n^2 iterations) implies that

$$\begin{aligned} \left\| \bar{x} - \mathcal{L}^+\bar{b} \right\|_{\mathbf{U}_{\mathcal{L}}} &\leq \sum_i \left(\left\| \bar{e}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}}} + \left\| \mathcal{L}^+\bar{f}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}}} \right) \\ &\leq \sum_i \left(\left\| \bar{e}^{(i)} \right\|_{\mathbf{U}_{\mathcal{L}}} + \left\| \bar{f}^{(i)} \right\|_{\mathbf{U}^+} \right) \\ &\leq \frac{1}{4n} \left\| \bar{b} \right\|_{\mathbf{U}_{\mathcal{L}^+}} \\ &\leq \frac{1}{2} \left\| \mathcal{L}^+\bar{b} \right\|_{\mathbf{U}_{\mathcal{L}}}. \end{aligned}$$

Here the second and last inequalities follow from Lemma F.3. This is the desired bound on the final error of the solver.

Finally, we need to show that the procedure can be implemented in the desired runtime. We note that the contracted matrices $(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top})$ are still Eulerian Laplacians, and the contractions cannot increase the number of vertices or edges. The actual systems solved are in $(\mathbf{C}^{(i)}\mathcal{L}\mathbf{C}^{(i)\top} + \frac{w^{(i)}}{r^2}\mathbf{I})$, or an Eulerian Laplacian plus positive diagonal. This matrix can be reduced, with the reduction in Section 5 of [12], to solving an Eulerian Laplacian with asymptotically the same sparsity and condition number. The symmetrized matrix for each system has min eigenvalue at least $\frac{w^{(i)}}{r^2}$ and max eigenvalue at most $O(nw^{(i)})$, so all condition numbers of their symmetrizations are polynomially bounded, as desired. \square