

Derandomizing Directed Random Walks in Almost-Linear Time

Rasmus Kyng*,
ETH Zurich
kyng@inf.ethz.ch

Simon Meierhans*,
ETH Zurich
mesimon@inf.ethz.ch

Maximilian Probst Gutenberg*,
ETH Zurich
maximilian.probst@inf.ethz.ch

August 24, 2022

Abstract

In this article, we present the first *deterministic* directed Laplacian \mathbf{L} systems solver that runs in time almost-linear in the number of non-zero entries of \mathbf{L} . Previous reductions imply the first deterministic almost-linear time algorithms for computing various fundamental quantities on directed graphs including stationary distributions, personalized PageRank, hitting times and escape probabilities.

We obtain these results by introducing *partial symmetrization*, a new technique that makes the Laplacian of an Eulerian directed graph “less directed” in a useful sense, which may be of independent interest. The usefulness of this technique comes from two key observations: Firstly, the partially symmetrized Laplacian preconditions the original Eulerian Laplacian well in Richardson iteration, enabling us to construct a solver for the original matrix from a solver for the partially symmetrized one. Secondly, the undirected structure in the partially symmetrized Laplacian makes it possible to sparsify the matrix *very crudely*, i.e. with large spectral error, and still show that Richardson iterations converge when using the sparsified matrix as a preconditioner. This allows us to develop deterministic sparsification tools for the partially symmetrized Laplacian.

Together with previous reductions from directed Laplacians to Eulerian Laplacians, our technique results in the first deterministic almost-linear time algorithm for solving linear equations in directed Laplacians. To emphasize the generality of our new technique, we show that two prominent existing (randomized) frameworks for solving linear equations in Eulerian Laplacians can be derandomized in this way: the squaring-based framework of [CKP⁺17] and the sparsified Cholesky-based framework of [PS22].

*The research leading to these results has received funding from grant no. 200021 204787 of the Swiss National Science Foundation.

1 Introduction

The development of spectral graph sparsification and nearly-linear time solvers for Laplacian linear equations initiated by the seminal article of Spielman and Teng [ST04], that has since been split into three parts [ST13, ST11, ST14], is foundational for algorithmic spectral graph theory and one of the success stories in the design of graph algorithms. Until recently, spectral techniques were mainly used for analysing undirected graphs. While many of the techniques developed in algorithmic spectral graph theory heavily use that Laplacians of undirected graphs are symmetric, positive semi-definite matrices, Cohen-Kelner-Peebles-Peng-Sidford-Vladu [CKP⁺16a] first demonstrated that the structure of directed Laplacians can be used to accelerate linear equation solvers. Together with Rao they later introduced the first notion of spectral approximation for directed Laplacians [CKP⁺17], which enabled them to develop sparsification tools in the directed setting. These tools were used to build the first almost-linear time directed Laplacian solver, operating in the framework of an undirected Laplacian solver by Peng and Spielman [PS14]. The runtime was improved to nearly-linear time by Cohen-Kelner-Kyng-Peebles-Peng-Rao-Sidford [CKK⁺18] and further improved by Peng and Song [PS22] very recently. Both operate within sparsified-Cholesky frameworks for solving undirected Laplacian linear equations: [CKK⁺18] uses the framework of Kyng and Sachdeva [KS16] and [PS22] uses the framework of Kyng-Lee-Peng-Sachdeva-Spielman [KLP⁺16].

All previous fast algorithms for solving directed Laplacian linear equations rely on sampling for globally sparsifying directed graphs while retaining a spectral $(1 \pm \epsilon)$ -approximation guarantee for some $\epsilon < 1$, according to the notion of approximation by [CKP⁺17]¹. This suggests two main approaches to derandomizing these solvers: (1) developing a fast deterministic $(1 \pm \epsilon)$ -approximate spectral sparsification routine or (2) introducing cruder forms of approximation and adapting the algorithms to cope with weaker guarantees. However, even for undirected graphs, no deterministic almost-linear time algorithms that achieve $(1 \pm \epsilon)$ -approximate spectral sparsification are known, and this is a major obstacle to approach (1). We circumvent this issue by instead taking the route (2): we develop a new way of measuring approximation via the suitability as a preconditioner in Richardson, which allows cruder guarantees. We achieve this by introducing a “robustification” step before sparsification, which we call β -partial symmetrization. Partial symmetrization counteracts the fragility of directed approximations and is at the core of our new crude deterministic sparsification procedure for Eulerian Laplacians. It lets us derandomize the frameworks of [CKP⁺17] and [PS22] with an almost-linear runtime.

1.1 Prior Work

Undirected Laplacian Solvers. The first nearly-linear time Laplacian solver [ST13, ST11, ST14] sparked the development of a field producing a whole host of distinct algorithms for solving Laplacian linear equations [KMP14, KMP11, KOSZ13, CKP⁺14, PS14, KLP⁺16, KS16, JS21]. Of these, our algorithm is most similar to [PS14] and its directed counterpart [CKP⁺17], which works by repeatedly squaring the adjacency matrix. In Appendix C we consider a Cholesky-factorisation based framework, which is motivated by [KLP⁺16] in the undirected setting and was recently translated to the directed setting by [PS22].

¹When we refer to $(1 \pm \epsilon)$ -approximations for Eulerian Laplacians in the introduction and overview we mean ϵ -approximations as in Definition 3.1. We use this naming for convenience when comparing to undirected approximations.

Spectral Sparsification. A crucial building block for spectral graph algorithms, including linear equation solvers, is the ability to sparsify undirected graphs while preserving their spectral properties. This is a stronger notion of sparsification than cut-sparsifiers, which only preserve the approximate size of cuts. Such spectral sparsifiers were first introduced in [ST11], and later strengthened and simplified by Spielman and Srivastava [SS11]. It is known that for every n -vertex graph there exists a spectral sparsifier with $O(n)$ -edges, and such sparsifiers can be constructed deterministically, as shown by Batson, Spielman, and Srivastava [BSS12]. However, no deterministic almost-linear time algorithms are known for $(1 \pm \epsilon)$ -spectral sparsification. Recently, Chuzhoy-Gao-Li-Nanongkai-Peng-Saranurak [CGL⁺20a] presented an almost-linear time algorithm achieving $n^{o(1)}$ -spectral sparsifiers for undirected graphs via deterministic expander decompositions. Our algorithm relies on both their sparsification and their expander decomposition results.

Directed Laplacian Solvers. Before [CKP⁺16a, CKP⁺17] it was unclear that directed Laplacians, which are a natural generalization of undirected Laplacians to directed graphs, also exhibit properties that allow for useful notions of sparsification and/or accelerated solving of linear equations. The reduction to strongly connected Eulerian Laplacians recovered some of the spectral properties of undirected Laplacians, and allowed for the development of a useful notion of sparsification. Current fast algorithms for solving Eulerian Laplacian linear equations either follow a squaring [CKP⁺17] or a sparsified-Cholesky approach [CKK⁺18, PS22]. Both rely on spectral sparsification techniques developed in [CKP⁺17].

Low Space Algorithms. Recently, the first deterministic $\tilde{O}(\log N)$ -space solver for Eulerian Laplacians was introduced by Ahmadinejad-Kelner-Murtagh-Peebles-Sidford-Vadhan [AKM⁺20]. They show that the Rozeman and Vadhan [RV05] deterministic squaring conserves approximation under squaring for a new, stronger measure of approximation. The directed to Eulerian reduction remains a major obstacle to solving directed Laplacian linear equations in small space.

Applications of Directed Laplacian Solvers. There are numerous applications of directed Laplacian solvers given in Section 7 of [CKP⁺16c], most of which are deterministic reductions to directed Laplacian system solving. The deterministic ones include

- solving large classes of linear systems,
- computing personalised PageRank vectors,
- estimating the stationary distribution,
- and simulating random walks.

Since the reductions are deterministic, we obtain deterministic almost-linear time algorithms for all these problems².

1.2 Our Contributions

We introduce the first notion of crude approximation for Eulerian Laplacians \mathbf{L} . It is defined via the suitability as a preconditioner in the Richardson iteration and sparse approximations are

²Note that we require an upper bound κ on the condition number, and hence mixing time.

constructed using *partial symmetrization* to increase robustness. This technique allows us to trade off additional Richardson iterations for a behaviour more akin to the undirected setting. The obtained *deterministic* crude global sparsification routine ultimately allows us to derandomize two prominent frameworks for solving directed Laplacian linear equations. We summarize our main result assuming polynomially bounded edge weights and condition number.

Theorem 1.1 (Informal Version of Theorem 7.1). *Given an Eulerian Laplacian $\mathbf{L}_{\vec{G}}$ associated with a strongly connected Eulerian Graph \vec{G} with n vertices and m edges, a vector $\mathbf{b} \in \text{im}(\mathbf{L}_{\vec{G}})$, and a parameter $\epsilon \in (0, 1)$ the algorithm $\text{SOLVEEULERIAN}(\mathbf{L}_{\vec{G}}, \epsilon)$ in time $m^{1+o(1)} \log \epsilon^{-1}$ computes a vector \mathbf{x} satisfying*

$$\|\mathbf{x} - \mathbf{L}_{\vec{G}}^+ \mathbf{b}\|_{U_{\mathbf{L}_{\vec{G}}}} \leq \epsilon \|\mathbf{L}_{\vec{G}}^+ \mathbf{b}\|_{U_{\mathbf{L}_{\vec{G}}}}$$

where $U_{\mathbf{A}} = (\mathbf{A} + \mathbf{A}^T)/2$ for any square matrix \mathbf{A} .

Previous reductions allow us to reduce general directed Laplacian solvers to $\log^{O(1)}(n\kappa^{-1}\epsilon^{-1})$ Eulerian solves with polynomially bounded condition number and edge weights. We state the main theorem as a corollary.

Theorem 1.2. *Given a directed Laplacian $\mathbf{L}_{\vec{G}} = \mathbf{D}_{\vec{G}} - \mathbf{A}_{\vec{G}}^T$ associated with a directed Graph \vec{G} with n vertices and m edges, a vector $\mathbf{b} \in \text{im}(\mathbf{L}_{\vec{G}})$, a parameter $\epsilon \in (0, 1)$ and an upper bound $\kappa \geq \max(\kappa(\mathbf{D}_{\vec{G}}), \kappa(\mathbf{L}_{\vec{G}}))$ there is an algorithm $\text{SOLVEFULL}(\mathbf{L}_{\vec{G}}, \epsilon)$ that in time $m^{1+o(1)} \log^{O(1)}(\kappa\epsilon^{-1})$ computes a vector \mathbf{x} satisfying*

$$\|\mathbf{x} - \mathbf{L}_{\vec{G}}^+ \mathbf{b}\|_2 \leq \epsilon \|\mathbf{L}_{\vec{G}}^+ \mathbf{b}\|_2$$

where $\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \|\mathbf{A}^+\|_2$ denotes the condition number of a given matrix \mathbf{A} .

A key application of these results is the deterministic simulation of random walks in almost-linear time as presented in Sections 7.4 and 7.5 of [CKP⁺16c].

Crude Global Sparsification. Our global sparsification routine is based on first increasing the robustness via β -partial symmetrization. Then we bucket by edge weight and layer partitions of undirected and unweighted graphs into expanders. First, the directed part in such an expander can be sparsified in a crude way that conserves the degrees. Then, the undirected part can be replaced with a sparse expander. Since the undirected part is scaled with a factor β by β -partial symmetrization it dominates the directed part which allows us to bound the error of the first step.

Open Questions. Our techniques, in particular β -partial symmetrization, might be of interest for deterministically simulating random walks in small space. Further, the development of a composable notion of crude approximation for Eulerian Laplacians is an interesting task. However, there are Eulerian Laplacians that do not precondition each other regardless of stepsize, which is a major obstruction (See Appendix D).

2 Overview

Existing randomized algorithms for solving linear equations in directed Eulerian Laplacians can be classified as belonging to one of two general frameworks: squaring-solvers and sparsified-cholesky-solvers. Both heavily rely on $(1 \pm \epsilon)$ -approximate graph sparsification techniques for two separate tasks: 1) *globally sparsifying* a directed graph \underline{G} with m -edges in time close to m and 2) *sparsifying the square graph* \underline{G}^2 in time close to m . The square graph \underline{G}^2 is the graph with adjacency matrix $\mathbf{A}_{\underline{G}} \mathbf{D}_{\underline{G}}^{-1} \mathbf{A}_{\underline{G}}$ where $\mathbf{A}_{\underline{G}}$ denotes the adjacency matrix of \underline{G} and $\mathbf{D}_{\underline{G}}$ denotes the diagonal matrix containing the degrees of the Eulerian graph \underline{G} . For some graphs the running time of 2) is sublinear in the number of edges of the sparsified graph \underline{G}^2 since squaring can drastically increase the density. We provide derandomized algorithms for both tasks, allowing us to derandomize two prominent frameworks. We focus on the squaring-solver [CKP⁺17], and sketch the sparsified-Cholesky solver [PS22] in Appendix C³.

While we match the $(1 \pm \epsilon)$ -approximation and roughly the runtime of randomized routines for 2), our algorithm for globally sparsifying directed graphs only achieves a crude $n^{o(1)}$ -approximation guarantee. This is not surprising, since deterministic $(1 \pm \epsilon)$ -approximate sparsification in almost linear time is an open problem even for undirected graphs. However, the current notions of spectral approximation for directed graphs due to [CKP⁺17] break down for approximation factor ϵ larger than 1. To address this problem, we directly define approximation via the suitability as a preconditioner in the Richardson iteration. In particular, suppose we want to solve a linear equation in \mathbf{M} , by preconditioning with a matrix \mathbf{N} . Roughly speaking, if, in an appropriate norm $\|\cdot\|$, we have $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{N}^+ \mathbf{M}\| \leq 1 - \alpha$, then we can solve the linear equation in \mathbf{M} to high accuracy using $\tilde{O}(1/\alpha)$ preconditioned Richardson iterations where we apply \mathbf{M} and \mathbf{N}^+ once per iteration. A priori, it may be surprising that our approach relies on distinguishing the case $1 - \|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{N}^+ \mathbf{M}\| \geq n^{o(-1)}$ from the case $1 - \|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{N}^+ \mathbf{M}\| \approx 0$, however, our partial symmetrization technique and a careful choice of norm makes this possible.

Our notion of high-error approximation is not directly composable, but it does algorithmically compose: If \mathbf{A} is preconditioned by \mathbf{B} and \mathbf{B} is preconditioned by \mathbf{C} , then a solver for \mathbf{A} can be constructed with access to \mathbf{B} and \mathbf{C}^+ using two layers of preconditioned Richardson. The development of a directly composable notion of spectral approximation in directed graphs for approximation factors larger than 1 remains an interesting open problem.

2.1 Global Sparsification

Given an Eulerian graph $\underline{G} = \underline{G}_0$ our sparsification routine does not directly produce a sparse graph $\tilde{\underline{G}} = \underline{G}_3$, but does so via a detour involving two other graphs \underline{G}_1 and \underline{G}_2 :

- \underline{G}_0 is the initial graph \underline{G} .
- \underline{G}_1 is obtained by β -partially-symmetrizing \underline{G}_0 . It has a directed and a undirected part.
- \underline{G}_2 is obtained from \underline{G}_1 by sparsifying its directed part without touching the undirected part.
- \underline{G}_3 is obtained by also sparsifying the undirected part.

³[PS22] also relies on randomness for finding a linear sized almost independent subset. We provide a simple procedure to derandomize this step in Appendix C.3.

We call such a bundle of four directed graphs a quadruple. Our construction ensures that $\mathbf{L}_{\underline{G}_{i+1}}^+$ is a suitable preconditioner for $\mathbf{L}_{\underline{G}_i}$ with respect to the norm $\|\cdot\|_{U_{\underline{G}_{i+1}} \rightarrow U_{\underline{G}_{i+1}}}$ for $i = 0, 1, 2$, meaning that $\mathbf{L}_{\underline{G}_i}^+$ can be (approximately) applied by applying $\mathbf{L}_{\underline{G}_{i+1}}^+$ for $N = \text{Exp}(O((\log n)^{1/10}))$ times via the preconditioned Richardson iteration⁴. Given an oracle for applying $\mathbf{L}_{\underline{G}_3}^+$ we can use it N times to apply $\mathbf{L}_{\underline{G}_2}^+$, and then in turn apply $\mathbf{L}_{\underline{G}_1}^+$ via N applications of $\mathbf{L}_{\underline{G}_2}^+$, until we can finally apply $\mathbf{L}_{\underline{G}_0}^+$. This yields a procedure for applying $\mathbf{L}_{\underline{G}_0}^+$ that relies on $N^3 = \text{Exp}(O((\log n)^{1/10}))$ applications of $\mathbf{L}_{\underline{G}_3}^+$. Next we describe the way the graphs \underline{G}_1 , \underline{G}_2 and \underline{G}_3 are constructed in more detail. Our full global sparsification results are presented in Section 4.

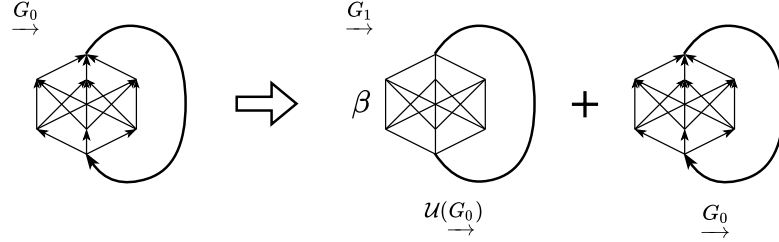


Figure 1: Obtaining \underline{G}_1 from \underline{G}_0 via β -partial symmetrization. Given the graph \underline{G}_0 as depicted on the left hand side, we add β times its undirectification. This could double the amount of edges, but the graph \underline{G}_1 has much less directed structure, which we will use to sparsify it in the next steps.

From \underline{G}_0 to \underline{G}_1 : Robustness through Partial Symmetrization. Let $\mathcal{U}(\underline{G}_0)$ denote the undirectification of the graph \underline{G}_0 , i.e. the graph obtained by replacing each directed edge with an undirected edge of half the weight. Then we simply obtain $\underline{G}_1 = \beta \cdot \mathcal{U}(\underline{G}_0) + \underline{G}_0$ where $\beta = n^{o(1)}$ is a sub-polynomial factor. Surprisingly, even though \underline{G}_1 removes a lot of directed structure, $\mathbf{L}_{\underline{G}_0}^+$ can be applied via $O(\beta)$ applications of $\mathbf{L}_{\underline{G}_1}^+$. Although \underline{G}_1 might seem similar to $\beta \cdot \mathcal{U}(\underline{G}_0)$, in fact, they behave very differently. The key technical observation is that

$$\left\| \mathbf{L}_{\mathcal{U}(\underline{G}_1)}^{+/2} (\mathbf{L}_{\underline{G}_0} - \mathbf{L}_{\underline{G}_1}) \mathbf{L}_{\mathcal{U}(\underline{G}_1)}^{+/2} \right\|_2 = \left\| \mathbf{L}_{(1+\beta)\mathcal{U}(\underline{G}_0)}^{+/2} \mathbf{L}_{\beta\mathcal{U}(\underline{G}_0)} \mathbf{L}_{(1+\beta)\mathcal{U}(\underline{G}_0)}^{+/2} \right\|_2 = \frac{1}{\beta + 1}$$

which relies on the fact that the directed graphs *cancel out* additively, only leaving us with symmetric Laplacians. See Figure 1 for an illustration.

From \underline{G}_1 to \underline{G}_2 : Patching Expander Parts. The graph $\underline{G}_1 = \beta \cdot \mathcal{U}(\underline{G}_0) + \underline{G}_0$ is made up of two parts: an undirected graph $\beta \cdot \mathcal{U}(\underline{G}_0)$ and a directed graph \underline{G}_0 . In this step we aim to sparsify the directed part by constructing a sparse graph \underline{R} with the same in- and out-degrees as \underline{G}_0 . Then $\underline{G}_2 = \beta \cdot \mathcal{U}(\underline{G}_0) + \underline{R}$. Our strategy for obtaining \underline{R} relies on the structure of the undirected graph $\mathcal{U}(\underline{G}_0)$ and deterministic expander decompositions as presented in [CGL⁺20a].

⁴Because not all approximations refer to the same norm, this does unfortunately not ensure that $\mathbf{L}_{\underline{G}_3}^+$ is a suitable preconditioner for $\mathbf{L}_{\underline{G}_0}$.

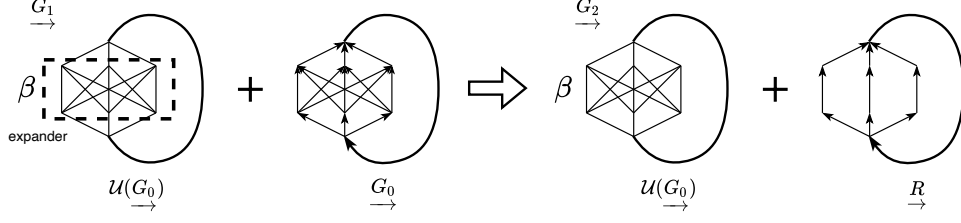


Figure 2: Obtaining \underline{G}_2 from \underline{G}_1 via patching. The dashed box contains the complete bipartite graph, which is a good expander. We use this information to drastically sparsify the directed part on the same vertex set V' , only ensuring that the degrees match up by increasing the weight of the edges. Since the induced subgraph on V' remains a good expander when summing up, this does not alter the spectral structure of the graph by much.

Our main technical observation here is that given a vertex set V' so that $\mathcal{U}(\underline{G}_0)[V']$ is a good enough expander, we can replace the directed graph $\underline{G}_0[V']$ in \underline{G}_1 with *any other* directed graph \underline{R}' , as long as $\underline{G}_0[V']$ and \underline{R}' have exactly the same in- and out-degrees, retaining a close approximation between $\underline{G}_1 - \underline{G}_0[V'] + \underline{R}'$ and \underline{G}_1 . Our algorithm for constructing \underline{R} first removes the weighted structure from $\mathcal{U}(\underline{G}_0)$ by bucketing by edge weight, and then layers deterministic undirected and unweighted expander decompositions. These are partitions of undirected and unweighted graphs into expanding parts and a remainder, and the layering is achieved by recursing on the remainder. A single expanding part can be greedily sparsified using any sparse greedily constructed graph \underline{R}' as introduced above. Summing up all of these yields \underline{R} . Crucially, while the error sums up over layers and buckets, the disjointness of the expander parts ensures this is not the case within a single decomposition. See Figure 2 for an illustration.

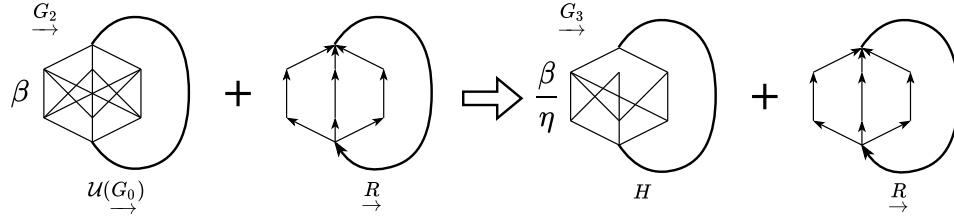


Figure 3: Obtaining \underline{G}_3 from \underline{G}_2 via undirected sparsification. The undirected part $\beta \cdot \mathcal{U}(\underline{G}_0)$ of \underline{G}_2 is sparsified using previously known spectral sparsification routines for undirected graphs. Since both remaining parts are sparse, their sum is a sparse graph.

From \underline{G}_2 to \underline{G}_3 : Scaling an Undirected Sparsifier. Finally we sparsify the undirected part $\beta \cdot \mathcal{U}(\underline{G}_0)$ obtaining a sparse undirected graph H and let $\underline{G}_3 = H + \underline{R}$. This can be achieved via known theorems for deterministic low-accuracy undirected graph sparsification provided by [CGL⁺20b]. To obtain a good preconditioner we have to scale H , but not \underline{R} , with the inverse of an appropriate rate η . This relies crucially on our observation that when the approximation error is only on the undirected part, learning rates can be leveraged more effectively than for general Eulerian approximation. We are left with a sum of two sparse graphs H and \underline{R} , which is a sparse directed graph. See Figure 3 for an illustration.

2.2 Sparsified Squaring

Given a directed graph \underline{G} with Laplacian $\mathbf{L}_{\underline{G}} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T \in \mathbb{R}^{n \times n}$, the Laplacian of its square \underline{G}^2 is given by

$$\mathbf{L}_{\underline{G}^2} = \mathbf{D}_{\underline{G}} - \underbrace{\mathbf{A}_{\underline{G}}^T \mathbf{D}_{\underline{G}}^{-1} \mathbf{A}_{\underline{G}}}_{\text{adjacency matrix}} = \mathbf{D}_{\underline{G}} - \sum_{i=1}^n \underbrace{\frac{1}{\mathbf{D}_{\underline{G}}(i, i)} (\mathbf{A}(i, :))^T \cdot (\mathbf{A}(:, i))^T}_{\mathbf{A}_i^T}.$$

We consider the directed product graphs $\mathbf{L}_i = \mathbf{D}_i - \mathbf{A}_i^T$ with adjacency matrix \mathbf{A}_i . Consider the matrix

$$\mathbf{L} = \begin{pmatrix} \text{diag}(\mathbf{A}_i \mathbf{1}) & \mathbf{0} \\ \mathbf{0} & \text{diag}(\mathbf{A}_i^T \mathbf{1}) \end{pmatrix} - \begin{pmatrix} \mathbf{0} & \mathbf{A}_i \\ \mathbf{A}_i^T & \mathbf{0} \end{pmatrix}$$

which is the Laplacian of a bipartite product graph G . Such graphs are constant expanders, which allows for a simple trick. First, we sparsify the undirected bipartite product graph \mathbf{L} to high accuracy ϵ , by adapting a procedure from [KLP⁺15] to be degree preserving. We call the sparse bipartite graph we obtain $\tilde{\mathbf{L}}$. Then we obtain a sparsified version $\tilde{\mathbf{A}}_i^T$ of \mathbf{A}_i^T by simply taking the bottom left block of $\tilde{\mathbf{L}}$. We show that $\tilde{\mathbf{L}}_i = \mathbf{D}_i - \tilde{\mathbf{A}}_i^T$ is a ϵ/Φ^2 -approximation of \mathbf{L}_i , where Φ is the expansion of G . Using the fact that bipartite product graphs are constant expanders lets us directly translate the approximation guarantee, up to a constant overhead in runtime. We obtain an $(1 \pm \epsilon)$ -approximation $\mathbf{L}_{\tilde{\underline{G}}^2} = \sum_{i=1}^n \tilde{\mathbf{L}}_i$ of $\mathbf{L}_{\underline{G}^2}$ with $\text{nnz}(\mathbf{L}_{\tilde{\underline{G}}^2}) \leq O(\text{nnz}(\mathbf{L}_{\underline{G}})\epsilon^{-4})$ in almost linear time. Similar deterministic squaring techniques were developed for small space algorithms [RV05, AKM⁺20], albeit the slightly different guarantees. We believe its likely that their approach to be adapted to our setting, but for convenience we adapt a more direct approach.

2.3 The Squaring Framework

Consider an Eulerian Laplacian $\mathbf{L}_{\underline{G}} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T$ where $\mathbf{A}_{\underline{G}}$ is the adjacency matrix of an Eulerian graph \underline{G} and $\mathbf{D}_{\underline{G}}$ is the diagonal matrix containing the degrees. Then the normalised Laplacian is given by $\mathbf{N} = \mathbf{I} - \mathbf{A}^T$ where $\mathbf{A} = \mathbf{D}_{\underline{G}}^{+1/2} \mathbf{A}_{\underline{G}}^T \mathbf{D}_{\underline{G}}^{+1/2}$, having the property that $\|\mathbf{A}\|_2 \leq 1$. The Neumann series expansion yields

$$(\mathbf{I} - \mathbf{A})^+ \mathbf{b} = \sum_{i=0}^{\infty} \mathbf{A}^i \mathbf{b} = \prod_{i=0}^{\infty} (\mathbf{I} + \mathbf{A}^{2^k}) \mathbf{b}$$

for \mathbf{b} orthogonal to the kernel of $(\mathbf{I} - \mathbf{A})^+$. Given a $1/\text{poly}(n)$ lower bound on the smallest eigenvalue λ_* of $\mathbf{I} - \mathbf{A}$, truncating the product expansion after $\Theta(\log n)$ factors (and hence squarings) yields a constant relative error. A very convenient equality in the same spirit is given by

$$(\mathbf{I} - \mathbf{A})^+ \mathbf{b} = (\mathbf{I} - \mathbf{A}^2)^+ (\mathbf{I} + \mathbf{A}) \mathbf{b} \tag{1}$$

for \mathbf{b} orthogonal to the kernel of $(\mathbf{I} - \mathbf{A})^+$ which is at the center of the squaring mechanism of [CKP⁺16b], the algorithm our squaring solver resembles most. Their squaring scheme is in turn inspired by the squaring solver for symmetric Laplacians presented in [PS14].

This leaves us with the task of solving linear equations in $\mathbf{I} - \mathcal{A}^2$, which is another normalised Laplacian. However, this is the normalised Laplacian of the square graph \underline{G}^2 , and it can be shown that squaring drastically improves the condition number of the problem, such that after $k = \Theta(\log n)$ squaring steps linear equations can be solved to high accuracy quickly via a simple iterative scheme. To avoid periodic behaviours we consider the normalised adjacency matrix $\mathcal{A}_l^{(\alpha)} := \alpha \mathbf{I} + (1 - \alpha)\mathcal{A}_l$, which can be interpreted as adding self loops proportional to the out-degrees of \underline{G} .

Sparsified Squaring. Since squaring not only improves the condition number, but may also quickly increase the density of the graph, we let $\mathcal{A}_0 = \mathcal{A}$ and iteratively obtain \mathcal{A}_{j+1} by implicitly sparsifying $(\mathcal{A}_j^{(\alpha)})^2$ using our sparsified squaring technique with accuracy parameter ϵ . We can conclude from (1) that

$$(\mathbf{I} - \mathcal{A})^+ \mathbf{b} \approx \underbrace{(1 - \alpha)^{d-1} (\mathbf{I} - \mathcal{A}_d)^+ \left(\mathbf{I} + (\mathcal{A}_{d-1}^{(\alpha)})^2 \right) \cdots \left(\mathbf{I} - (\mathcal{A}_0^{(\alpha)})^2 \right)}_{:= \mathbf{Z}} \mathbf{b}$$

as we can ensure that \mathbf{b} is orthogonal to the known kernel of $(\mathbf{I} - \mathcal{A})^+$. However, the repeated sparsification accumulates an error proportional to ϵe^d , and it is imperative that it stays below 1 such that \mathbf{Z} is an approximate pseudoinverse of $\mathbf{I} - \mathcal{A}$. Therefore, we have to choose ϵ proportional to e^{-d} . We conclude from the previous subsection that $\text{nnz}(\mathcal{A}_d) = O(\text{nnz}(\mathcal{A})e^{4d^2})$. Unfortunately, we cannot set $d = \Theta(\log n)$ without ending up with potentially dense matrices. Therefore, we set $d = \Theta((\log n)^{1/3})$ and have $e^{4d^2} = n^{o(1)}$.

Global Sparsification and Chains of Sparse Matrices. Since $d = \Theta((\log n)^{1/3})$ squarings do not sufficiently decrease the condition number, we globally sparsify after d sparsified squarings and repeat. Given $\mathcal{A}_0^{(0)} = \mathbf{D}_{\underline{G}}^{+/2} \mathcal{A}_{\underline{G}} \mathbf{D}_{\underline{G}}^{+/2}$ for $i = 0, \dots, \Theta((\log n)^{2/3})$ we iteratively construct:

- Given $\mathcal{A}_0^{(i)}$, construct $\mathcal{A}_0^{(i)}, \dots, \mathcal{A}_d^{(i)}$ by sparsified squaring as described in the previous paragraph.
- Let \underline{H} be the graph with adjacency matrix $\mathbf{D}_{\underline{G}}^{1/2} \mathcal{A}_d^{(i)} \mathbf{D}_{\underline{G}}^{1/2}$. Globally sparsify \underline{H} obtaining $\tilde{\underline{H}}$. Then let $\mathcal{A}_0^{(i+1)} = \mathbf{D}_{\underline{G}}^{+/2} \mathcal{A}_{\tilde{\underline{H}}} \mathbf{D}_{\underline{G}}^{+/2}$.

We discuss these collections of squaring chains linked by global sparsification in Section 6. For our algorithm to run in almost-linear time, it is imperative that these chains are constructed once, and then our algorithm operates recursively on them.

The Recursive Algorithm. Our global sparsification routine allows us to solve linear equations in $\mathbf{I} - \mathcal{A}_0^{(i)}$ by solving $\text{Exp}(O((\log n)^{1/10}))$ linear equations in $\mathbf{I} - \mathcal{A}_0^{(i+1)}$. Since linear equations in $\mathbf{I} - \mathcal{A}_0^{(\Theta(\log n)^{2/3})}$ are easy to solve using standard iterative procedures, this is the depth of our recursion. Therefore, the total amount of branches is $\text{Exp}(O((\log n)^{2/3+1/10})) = n^{o(1)}$. Since all involved matrices contain an almost linear amount of entries, this gives an almost linear time deterministic algorithm for solving linear equations involving Eulerian Laplacians. That is, because preconditioned Richardson only does matrix vector multiplications with the matrix and the preconditioner.

2.4 The Sparsified-Cholesky Framework

Very recently [PS22] showed that the framework of [KLP⁺16] directly works for Eulerian Laplacians by developing new tools for analysing the accumulation of error. Our sparsification tools can also be used to derandomize this algorithm. Unlike the squaring framework, which makes progress by improving the condition number, sparsified-Cholesky frameworks operate by eliminating rows and columns like Gaussian elimination. Such elimination steps can be directly interpreted as deleting a vertex from the graph and adding a weighted clique.

Some algorithms eliminate one vertex at a time [KS16, CKK⁺18], but [KLP⁺16] and [PS22] eliminate a large set of $\Omega(n)$ vertices together. To do so, a linear sized ρ -row-column-diagonally-dominant (ρ -RCDD) subset V' of the vertices is chosen for some constant ρ . A set of vertices is ρ -RCDD, if for each vertex a ρ -fraction of the weighted in-edges come from $V \setminus V'$ and a ρ -fraction of the weighted out-edges go to $V \setminus V'$ ⁵. Then the vertices belonging to this set can be eliminated using $O(\log \log n)$ sparsified squaring operations. While randomized algorithms using this paradigm can afford to globally sparsify after each squaring step, we have to allow for some build up of the edge count. Namely, we wait for a subpolynomial number of elimination rounds, and then globally sparsify and recurse. As previously, global sparsifications correspond to branching points when applying the inverse. We give a more detailed description in Appendix C.

2.5 Reduction to the Eulerian Setting with bounded Condition Number

Previous work [CKP⁺16c, CKP⁺16b] reduced solving linear equations in directed Laplacians $\mathbf{L} = \mathbf{D} - \mathbf{A}^T$ to solving $\log^{O(1)}(n\kappa^{-1}\epsilon^{-1})$ systems involving Eulerian Laplacians with polynomially bounded condition number and edge weights to constant accuracy, where κ is an upper bound on the maximum of $\kappa(\mathbf{D})$ and $\kappa(\mathbf{L})$ (See Appendix D and F of [CKP⁺16b] and Sections 5, 7.1 and 7.3 of [CKP⁺16c]). They use that edge weights are polynomially bounded in the proof of Lemma C.3 in Appendix C of [CKP⁺16b]. Different reductions to the Eulerian case were presented by Ahmadinejad-Jambulapati-Saberi-Sidford [AJSS19] and in the thesis of Peebles [Pee19].

3 Preliminaries

3.1 Linear Algebra

Matrices. We denote matrices as bold upper case letters \mathbf{A} . For a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, we let $\text{nnz}(\mathbf{A})$ denote its number of non-zero entries and for $X \subseteq [n]$, $Y \subseteq [n]$ we let $\mathbf{A}(X, Y) = \mathbf{A}_{XY}$ denote the $|X| \times |Y|$ submatrix containing the entries with index in $X \times Y$. If $X = Y$, we write $\mathbf{A}[X]$ as a shorthand for $\mathbf{A}(X, X)$. When selecting submatrices, we let $l : u$ denote the set $\{l, l+1, \dots, u\}$ for $u \geq l$ and $:$ the set of all columns/rows, e.g. $\mathbf{A}(1 : 3, :)$ denotes the submatrix of \mathbf{A} consisting of the first 3 rows of \mathbf{A} . Further, we let \mathbf{A}^+ denote the Moore-Penrose-Pseudoinverse of matrix \mathbf{A} . Finally, \mathbf{I} denotes the identity matrix. Sometimes we denote its dimension with \mathbf{I}_n .

Vectors. We denote vectors as bold lower case letters \mathbf{v} . Further, we let $\mathbf{1}$ denote the all ones vector, $\mathbf{0}$ the zero vector and \mathbf{e}_i denotes the i -th vector of the standard basis. Sometimes we indicate the dimension with $\mathbf{1}_n$ and $\mathbf{0}_n$.

⁵Notice that [KLP⁺15] is concerned with the undirected case.

The Loewner-order. For symmetric matrices \mathbf{A} and \mathbf{B} , we let $\mathbf{A} \preceq \mathbf{B}$ iff for all vectors \mathbf{x} : $\mathbf{x}^T \mathbf{A} \mathbf{x} \leq \mathbf{x}^T \mathbf{B} \mathbf{x}$. We define \prec, \succeq and \succ analogously. If a symmetric matrix \mathbf{A} satisfies $\mathbf{0} \preceq \mathbf{A}$ we call it *PSD*. For a *PSD* matrix \mathbf{A} , we let $\mathbf{A}^{1/2}$ denote the unique matrix so that $\mathbf{A}^{1/2} \mathbf{A}^{1/2} = \mathbf{A}$.

Norms. For every vector \mathbf{x} , we let $\|\mathbf{x}\|_{\mathbf{H}} := \sqrt{\mathbf{x}^T \mathbf{H} \mathbf{x}}$ for a *PSD* matrix \mathbf{H} . We let $\|\mathbf{M}\|_{\mathbf{H} \rightarrow \mathbf{H}} := \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{M} \mathbf{x}\|_{\mathbf{H}}}{\|\mathbf{x}\|_{\mathbf{H}}}$ for a *PSD* matrix \mathbf{H} . Notice that $\|\mathbf{M}\|_{\mathbf{H} \rightarrow \mathbf{H}} = \left\| \mathbf{H}^{1/2} \mathbf{M} \mathbf{H}^{1/2} \right\|_2$. Further, we let $\|\mathbf{M}\|_1$ and $\|\mathbf{M}\|_{\infty}$ denote the maximum ℓ_1 norm of a column and row of \mathbf{M} respectively.

Condition Number. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ we let $\kappa(\mathbf{A}) := \|\mathbf{A}\|_2 \|\mathbf{A}^+\|_2$ denote its condition number. Further, for *PSD* matrices \mathbf{A} and \mathbf{B} with the same kernel we let $\kappa(\mathbf{A}, \mathbf{B}) := \kappa(\mathbf{A}^{+1/2} \mathbf{B} \mathbf{A}^{+1/2})$.

Misc. We let $\text{Exp}(x) := e^x$. In this paper $\tilde{O}(\cdot)$ suppresses poly-logarithmic factors in n . For a *PSD* matrix \mathbf{A} , we let $\lambda_*(\mathbf{A})$ denote its smallest nonzero eigenvalue.

3.2 Graphs

General Notation. We let $G = (V, E, \omega)$ denote an undirected graph where $\omega(e) = \omega(u, v)$ denotes the weight of edge $e = (u, v)$. Further, we let $\underline{G} = (V, E, \omega)$, where the edge weight $\omega(e) = \omega(u, v)$ of edge e now depends on its direction. Sometimes we omit ω for unit weight (aka unweighted) graphs. When sometimes also write $V(\underline{G}), E(\underline{G})$ and $\omega_{\underline{G}}$ to avoid ambiguity. We let ω^{\max} and ω^{\min} denote the maximum and minimum edge weight respectively.

Undirectification. For a directed graph $\underline{G} = (V, E, \omega)$, we let $\mathcal{U}(\underline{G}) = (V, E', \omega)$, with $\{u, v\} \in E'$ iff $(u, v) \in E$ or $(v, u) \in E$ and $\omega\{u, v\} = \frac{1}{2}(\omega(u, v) + \omega(v, u))$, denote its undirectification (where we use the convention $\omega(u, v) = 0$ for $(u, v) \notin E$).

Induced Subgraphs. For $G = (V, E, \omega)$ and $X \subseteq V$ we let $G[X]$ denote the induced subgraph on X . For a directed graph \underline{G} we define $\underline{G}[X]$ analogously.

3.3 Graph Laplacians

General Notation. For an undirected graph $G = (V, E, \omega)$ we denote its (graph) Laplacian as $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$ where \mathbf{D}_G is the diagonal matrices containing the degrees and $\mathbf{A}_G(i, j) = \mathbf{A}_G(j, i) = \omega((i, j))$. Generalizing this notation to directed graphs $\underline{G} = (V, E, \omega)$, we let $\mathbf{L}_{\underline{G}} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T$ where the adjacency matrix is given by $\mathbf{A}_{\underline{G}}(i, j) = \omega((i, j))$ and the diagonal matrix $\mathbf{D}_{\underline{G}}(i, i) = \sum_j \mathbf{A}_{\underline{G}}(i, j)$ contains the out degrees. Naturally $\mathbf{1}^T \mathbf{L}_{\underline{G}} = \mathbf{0}$. For an undirected graph G , we let $\deg_G(v)$ be the (weighted) degree of vertex v . For directed graphs we let $\deg_G^-(v)$ and $\deg_G^+(v)$ denote in- and out-degree respectively.

Eulerian Laplacians. If $\mathbf{L}_{\underline{G}} \mathbf{1} = \mathbf{0}$ we call a Laplacian Eulerian. This correspond to the underlying directed graph \underline{G} being Eulerian, i.e. having equal in- and out-degree for each vertex.

Symmetrization. For any matrix \mathbf{A} we denote $\mathbf{U}_{\mathbf{A}} = \mathbf{U}(\mathbf{A}) := \frac{1}{2}(\mathbf{A} + \mathbf{A}^T)$. Notice that for an Eulerian Laplacian $\mathbf{L}_{\vec{G}}$ we have $\mathbf{U}_{\mathbf{L}_{\vec{G}}} = \mathbf{L}_{\mathcal{U}(\vec{G})}$. This is a crucial fact exploited by all algorithms for directed Laplacians including ours. Symmetric Laplacians are *PSD*.

Induced subgraphs and submatrices. The reader should note that for a Laplacian $\mathbf{L}_{\vec{G}}$ the matrices $\mathbf{L}_{\vec{G}[X]}$ and $\mathbf{L}_{\vec{G}}[X]$ are not equivalent unless there is no edge from X to $V \setminus X$ or vice versa. Specifically, while the off-diagonal entries are equal, we have $\mathbf{D}_{\vec{G}[X]}(i, i) \leq \mathbf{D}_{\vec{G}}[X](i, i)$.

3.4 Directed Graph Approximation

We will use the notions of approximation for directed graphs introduced in [CKP⁺16b].

Definition 3.1 (Asymmetric Matrix Approximation, Definition 3.1 in [CKP⁺16b]). *A (possibly asymmetric) matrix $\tilde{\mathbf{A}}$ is said to be an ϵ -matrix-approximation of \mathbf{A} if*

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

Remark 3.2. Notice that Definition 3.1 is not symmetric.

3.5 Expanders

Expander graphs, or expanders for short, play a central role in both the workings and analysis of our algorithms. Notice that while our algorithm operates on directed graphs, we only use expanders in the context of undirected graphs. Whenever we use the term expander, we mean expanders with respect to *conductance*. We follow the notational conventions of [CGL⁺20b].

Definition 3.3 (Conductance). *For a weighted but undirected graph $G = (V, E, \omega)$, given a set $\emptyset \subset S \subset V$ we let $\delta_G(S) := \sum_{(u,v) \in E: u \in S, v \notin S} \omega(u, v)$ and $\text{vol}_G(S) := \sum_{v \in S} \sum_{u \in V} \omega(u, v)$. Then we define the conductance*

$$\Phi_G(S) = \frac{\delta_G(S)}{\min\{\text{vol}_G(S), \text{vol}_G(V \setminus S)\}}.$$

Definition 3.4 (Expander). *We call a graph $G = (V, E, \omega)$ a Φ -expander (with respect to conductance) if $\min_{S: \emptyset \subset S \subset V} \Phi_G(S) \geq \Phi$. We further let $\Phi_G = \min_{S: \emptyset \subset S \subset V} \Phi_G(S)$.*

Given a vector $\mathbf{d} \in \mathbb{R}_{>0}^n$ we let $G(\mathbf{d})$ denote the weighted and undirected graph on n vertices with $\omega(i, j) = \frac{\mathbf{d}(i) \cdot \mathbf{d}(j)}{\|\mathbf{d}\|_1}$. Note that $\mathbf{L}_{G(\mathbf{d})} = \text{diag}(\mathbf{d}) - \frac{\mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|_1}$.

Fact 3.5 (Observation 6.6 in [CGL⁺20b]). *$G(\mathbf{d})$ is a $1/2$ -expander.*

We will frequently use that Laplacians $\mathbf{L}_H = \mathbf{D}_H - \mathbf{A}_H^T$ of good expanders H are well approximated by $\mathbf{L}_{G(\text{diag}(\mathbf{D}_H))}$. To establish this, we need the following lemma.

Lemma 3.6 (Lemma 6.7 in [CGL⁺20b]). *Let G and H be two undirected weighted n -vertex graphs on the same vertex set, such that $\mathbf{D}_G = \mathbf{D}_H$. Assume further that $\Phi(G), \Phi(H) \geq \Phi$ for some threshold Φ . Then for any real vector $\mathbf{x} \in \mathbb{R}^n$: $\frac{\Phi^2}{4} \mathbf{x}^T \mathbf{L}_G \mathbf{x} \leq \mathbf{x}^T \mathbf{L}_H \mathbf{x} \leq \frac{4}{\Phi^2} \mathbf{x}^T \mathbf{L}_G \mathbf{x}$.*

We state a direct corollary in a form that turns out to be convenient for our arguments.

Corollary 3.7 (Non Uniform Degree Bound). *Given a Φ -expander G with Laplacian $\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G^T$ we have*

$$\frac{\Phi^2}{4}(\mathbf{D}_G - \frac{\mathbf{d}_G \mathbf{d}_G^T}{\|\mathbf{d}_G\|_1}) \preceq \mathbf{L}_G \preceq \frac{4}{\Phi^2}(\mathbf{D}_G - \frac{\mathbf{d}_G \mathbf{d}_G^T}{\|\mathbf{d}_G\|_1})$$

for $\mathbf{d}_G = \text{diag}(\mathbf{D}_G)$ and $\Phi \leq 1/2$.

Proof. Directly follows from Fact 3.5 and Lemma 3.6. \square

Next we state the definition of the expander decomposition for unweighted and undirected graphs in the notation of [CGL⁺20b].

Definition 3.8 (See Section 6 of [CGL⁺20b], Proposed by [KVV04, GR99]). *A (ϵ, Φ) -expander decomposition of a undirected and unweighted graph $G = (V, E)$ is a partition $\mathcal{P} = \{V_1, \dots, V_k\}$ of the vertex set V such that for all $i \in [k]$ the conductance of $G[V_i]$ is at least Φ and $\sum_{i=1}^k \delta_G(V_i) \leq \epsilon \text{vol}_G(V)$.*

The next theorem shows that expander decompositions can be computed in almost linear time.

Theorem 3.9 (Corollary 7.7 in [CGL⁺20b]). *There is a deterministic algorithm that, given an undirected and unweighted graph $G = (V, E)$ with m edges and parameters $\epsilon \in (0, 1]$ and $1 \leq r \leq O(\log m)$, computes a (ϵ, Φ) -expander decomposition of G with $\Phi \geq \Omega(\epsilon/(\log m)^{O(r^2)})$ in time $O\left(m^{1+O(1/r)+o(1)} \cdot (\log m)^{O(r^2)}\right)$.*

Finally, we adapt the previous theorem to interface more conveniently with our statements.

Corollary 3.10 (Expander Decomposition). *There is a deterministic algorithm $\text{EXPDECOMP}(G, \gamma)$ that, given an undirected and unweighted graph $G = (V, E)$ with m edges and a constant $\gamma \in (0, 1)$, computes a $(1/2, \frac{1}{\text{Exp}((\log n)^\gamma)})$ -expander decomposition in time $m^{1+o(1)}$.*

Proof. Follows directly from Theorem 3.9. \square

3.6 Preconditioned Richardson

The next lemma analyses preconditioned Richardson (Algorithm 1) for asymmetric matrices.

Lemma 3.11 (Preconditioned Richardson, Lemma 4.2 in [CKP⁺16b]). *Let $\mathbf{b} \in \mathbb{R}^n$ and $\mathbf{M}, \mathbf{Z}, \mathbf{U} \in \mathbb{R}^{n \times n}$ such that \mathbf{U} is symmetric positive definite, $\ker(\mathbf{U}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M}^T) = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^T)$, and $\mathbf{b} \in \text{im}(\mathbf{M})$. Then N iterations of preconditioned Richardson with step size $\eta > 0$, result in a vector $\mathbf{x}_N = \text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \mathbf{b}, \eta, N)$ so that*

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

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

The previous lemma leads to the notion of an approximate pseudoinverse by measuring the suitability of a matrix as a preconditioner.

Definition 3.12 (Approximate Pseudoinverse, Definition 4.3 in [CKP⁺16b]). *Matrix \mathbf{Z} is an ϵ -approximate-pseudoinverse of matrix \mathbf{M} with respect to a PSD matrix \mathbf{U} , if $\ker(\mathbf{U}) \subseteq \ker(\mathbf{M}) = \ker(\mathbf{M})^T = \ker(\mathbf{Z}) = \ker(\mathbf{Z}^T)$, and*

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

Algorithm 1: PRECONRICHARDSON($\mathbf{M}, \mathbf{Z}, \mathbf{b}, \eta, N$)

```

1  $\mathbf{x}_0 = \mathbf{0}$ 
2 for  $i = 0, \dots, N - 1$  do
3    $\mathbf{x}_{i+1} = \mathbf{x}_i + \eta \mathbf{Z}(\mathbf{b} - \mathbf{M} \mathbf{x}_i)$ 
4 end
5 return  $\mathbf{x}_N$ 

```

Finally, we state three more lemmas that are crucial for arguing about approximate pseudoinverses. The first two are often applied consecutively to upper bound $\|\mathbf{I}_{\text{im}(\mathbf{M})} - \mathbf{Z}\mathbf{M}\|_{\mathbf{U}_Z \rightarrow \mathbf{U}_Z}$ with a variational form.

Lemma 3.13 (Part of Lemma B.9 in [CKP⁺16b]). *If \mathbf{L} is a matrix with $\ker(\mathbf{L}) = \ker(\mathbf{L}^T) = \ker(\mathbf{U}_L)$, and \mathbf{U}_L is positive semidefinite, then for any matrix \mathbf{A} with the same left and right kernels as \mathbf{L} we have*

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

Lemma 3.14 (Part of Lemma B.2 in [CKP⁺16b]). *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 $\ker(\mathbf{M}) \subseteq \ker(\mathbf{A}^T)$ and $\ker(\mathbf{N}) \subseteq \ker(\mathbf{A})$ we have*

$$\left\| \mathbf{M}^{+/2} \mathbf{A} \mathbf{N}^{+/2} \right\|_2 = 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T \mathbf{A} \mathbf{y}}{\mathbf{x}^T \mathbf{M} \mathbf{x} + \mathbf{y}^T \mathbf{N} \mathbf{y}}$$

where we define $0/0$ to be 0.

Lemma 3.15 (Part of Lemma B.4 in [CKP⁺16b]). *For a PSD diagonal matrix \mathbf{D} and any matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$*

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

4 Global Sparsification for Directed Laplacians

In this section we describe our low accuracy global sparsification routine. This constitutes the backbone of our algorithm. We first formally define the β -partial-symmetrization of an Eulerian graph $\underline{\mathcal{G}}$. See Figure 4 for an illustration.

Definition 4.1. *For an Eulerian directed graph $\underline{\mathcal{G}}$ we call $\mathcal{U}^{(\beta)}(\underline{\mathcal{G}}) := \beta \cdot \mathcal{U}(\underline{\mathcal{G}}) + \underline{\mathcal{G}}$ its β -partial-symmetrization.*

Remark 4.2. $\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})} = \beta \mathbf{U}_{\mathbf{L}_{\underline{G}}} + \mathbf{L}_{\underline{G}}$.

There are three steps in our sparsification procedure.

1. The first step relies on what may be the most crucial observation. Given an Eulerian directed graph $\underline{G} = \underline{G}_0$, we let $\underline{G}_1 = \mathcal{U}^{(\beta)}(\underline{G}) = \beta \cdot \mathcal{U}(\underline{G}) + \underline{G}$ be the graph obtained from \underline{G}_0 by β -partial symmetrization. Then, surprisingly, $\mathbf{L}_{\underline{G}_1}$ can be used as a preconditioner for solving linear equations in $\mathbf{L}_{\underline{G}_0}$ in time $O(\beta)$ using Richardson. We call $\mathcal{U}^{(\beta)}(\underline{G})$ the β -partial-symmetrization of \underline{G} .
2. A partial-symmetrization is naturally interpreted as the sum of an undirected graph $\beta \cdot \mathcal{U}(\underline{G})$ and a directed graph \underline{G} . We expander decompose the undirected graph $\beta \cdot \mathcal{U}(\underline{G})$ into parts V_1, V_2, \dots, V_k . Then a simple greedy patching scheme can be used to sparsify the induced subgraphs $\underline{G}[V_i]$ leveraging the expander structure for error control. In our actual algorithm we additionally bucket by edge weight and layer expander decompositions. The former allows us to treat the graph as unweighted and the latter ensures that every edge is in an expander after $O(\log n)$ layers. The Laplacian of the obtained graph $\underline{G}_2 = \beta \cdot \mathcal{U}(\underline{G}) + \underline{G}$ can then be used as a preconditioner for $\mathbf{L}_{\underline{G}_1}$.
3. Lastly, the undirected graph $\beta \cdot \mathcal{U}(\underline{G})$ can be sparsified via previously known deterministic algorithms presented in [CGL⁺20b]. We obtain $\underline{G}_3 = \frac{\beta}{\eta} \tilde{G} + \underline{R}$ which in turn is a preconditioner for \underline{G}_2 .

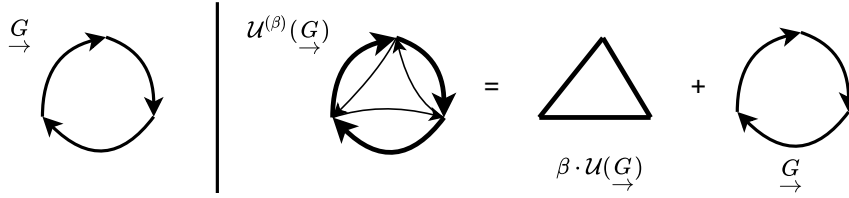


Figure 4: This drawing illustrates the concept of an β -partial-symmetrization. We use boldness to roughly indicate edge weights. On the left hand side an Eulerian graph \underline{G} is depicted. On the right hand side its β -partial-symmetrization is drawn, alongside its natural split into an undirected and directed part.

We define pseudoinverse sparsification quadruples. Constructing these is at the core of our global sparsification routine.

Definition 4.3. (*Sparsification quadruple*) We call strongly connected n -vertex Eulerian graphs $\underline{G}_0, \underline{G}_1, \underline{G}_2, \underline{G}_3$ a (γ, β, η) -quadruple for some constant $\gamma \in (0, 1)$ if

1. $\mathbf{L}_{\underline{G}_i}^+$ is a $\left(1 - \frac{1}{\text{Exp}(O((\log n)^\gamma))}\right)$ -approximate pseudoinverse of $\mathbf{L}_{\underline{G}_{i-1}}$ with respect to $\mathbf{U}_{\mathbf{L}_{\underline{G}_i}}$ for $i = 1, 2, 3$.
2. $\frac{1}{\text{Exp}(O((\log n)^\gamma))} \mathbf{U}_{\mathbf{L}_{\underline{G}_{i-1}}} \preceq \mathbf{U}_{\mathbf{L}_{\underline{G}_i}} \preceq \text{Exp}(O((\log n)^\gamma)) \mathbf{U}_{\mathbf{L}_{\underline{G}_{i-1}}}$ for $i \in \{1, 2, 3\}$.
3. $|E(\underline{G}_3)| = \tilde{O}(n)$. $|E(\underline{G}_i)| \leq 2|E(\underline{G}_0)| + \tilde{O}(n)$ for $i = 1, 2$.

4. For all vertices v : $\deg_{\underline{G}_0}^+(v) = \deg_{\underline{G}_0}^-(v) = (1 + \beta) \deg_{\underline{G}_i}^+(v) = (1 + \beta) \deg_{\underline{G}_i}^-(v)$ for $i = 1, 2$
and $\deg_{\underline{G}_0}^+(v) = (1 + \frac{\beta}{\eta}) \deg_{\underline{G}_3}^+(v) = (1 + \frac{\beta}{\eta}) \deg_{\underline{G}_3}^-(v)$.

We then state the main lemma of this section. It shows that it is possible to construct a (γ, β, η) -quadruple in almost linear time.

Lemma 4.4 (Global Sparsification). *For every m -edge, strongly connected Eulerian graph $\underline{G} = \underline{G}_0$ and a constant $\gamma \in (0, 1)$ the routine $\underline{G}_1, \underline{G}_2, \underline{G}_3 = \text{GLOBALSPARSIFICATION}(\underline{G}, \gamma)$ yields a $(\gamma, \beta = \text{Exp}(O((\log n)^\gamma)), \eta = \text{Exp}(-3(\log n)^\gamma))$ -quadruple $\underline{G}_0, \underline{G}_1, \underline{G}_2, \underline{G}_3$. The runtime is $m^{1+o(1)}$.*

Remark 4.5. While in the description of our algorithm, β scales linearly in $O\left(\log\left(\frac{\omega_{\underline{G}}^{\max}}{\omega_{\underline{G}}^{\min}}\right)\right)$, we assume throughout the paper that $\beta = \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma)$ is fixed to a global upper bound as $\log\left(\frac{\omega_{\underline{G}}^{\max}}{\omega_{\underline{G}}^{\min}}\right) = \tilde{O}(1)$ for all graphs \underline{G} we work with. This avoids clutter in the analysis.

Algorithm 2: GLOBALSPARSIFICATION(\underline{G}, γ)

- 1 $\beta = L \cdot \text{Exp}(2 \cdot (\log n)^\gamma)$ for $L = 128 \cdot 20 \cdot P \cdot \log n$ and $P = \left\lceil \log\left(\frac{\omega_{\underline{G}}^{\max}}{\omega_{\underline{G}}^{\min}}\right) \right\rceil$.
 - 2 $\eta = \text{Exp}(-3 \cdot (\log n)^\gamma)$
 - 3 $\underline{G}_1 = \beta \mathcal{U}(\underline{G}) + \underline{G}$; // Note that $\underline{G}_1 = \mathcal{U}^{(\beta)}(\underline{G})$.
 - 4 $\underline{R} = \text{SPARSIFYDIRECTED}(\underline{G}, \gamma)$
 - 5 $\underline{G}_2 = \beta \mathcal{U}(\underline{G}) + \underline{R}$
 - 6 $\tilde{G} = \text{SPECTRALSPARSIFYDEG}(\mathcal{U}(\underline{G}), \gamma)$
 - 7 $\underline{G}_3 = \frac{\beta}{\eta} \tilde{G} + \underline{R}$
 - 8 **return** $\underline{G}_1, \underline{G}_2, \underline{G}_3$
-

4.1 Preconditioning with the Partial-Symmetrization

Our next lemma shows that $\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}$ is a good preconditioner in terms of Lemma 3.11.

Lemma 4.6. *For every Eulerian Laplacian $\mathbf{L}_{\underline{G}_0}$ the matrix $\mathbf{L}_{\underline{G}_1}^+$ is an $(1 - \frac{1}{1+\beta})$ -approximate pseudoinverse of $\mathbf{L}_{\underline{G}_0}$ with respect to $\mathbf{U}_{\underline{G}_1}$ if $\underline{G}_1 = \mathcal{U}^{(\beta)}(\underline{G}_0)$.*

Proof. First recall that $\mathbf{L}_{\underline{G}_1}^+ = \mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}^+$. We have

$$\left\| \mathbf{I}_{\text{im}(\mathbf{L}_{\underline{G}})} - \mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}^+ \mathbf{L}_{\underline{G}} \right\|_{\mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}} \rightarrow \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}} \leq \left\| \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} (\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})} - \mathbf{L}_{\underline{G}}) \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} \right\|_2$$

by Lemma 3.13. With Remark 4.2 we conclude

$$\begin{aligned}
\left\| \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} (\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})} - \mathbf{L}_{\underline{G}}) \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} \right\|_2 &= \beta \left\| \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} \mathbf{U}_{\mathbf{L}_{\underline{G}}} \mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}}^{+/2} \right\|_2 \\
&= \frac{\beta}{1+\beta} \left\| \mathbf{U}_{\mathbf{L}_{\underline{G}}}^{+/2} \mathbf{U}_{\mathbf{L}_{\underline{G}}} \mathbf{U}_{\mathbf{L}_{\underline{G}}}^{+/2} \right\|_2 \\
&= \frac{\beta}{1+\beta} = 1 - \frac{1}{1+\beta}
\end{aligned}$$

where we use that $\mathbf{U}_{\mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}} = \mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})} = (1+\beta) \mathbf{U}_{\mathbf{L}_{\underline{G}}}$. The lemma follows from chaining the calculations. \square

4.2 Sparsifying the Directed Part

Given $\mathbf{L}_{\underline{G}_1} = \beta \mathbf{U}_{\mathbf{L}_{\underline{G}}} + \mathbf{L}_{\underline{G}}$, we aim to obtain a sparse directed graph \underline{R} with the same in- and out-degrees as \underline{G} so that the directed Laplacian $\mathbf{L}_{\underline{G}_2} = \beta \mathbf{U}_{\mathbf{L}_{\underline{G}}} + \mathbf{L}_{\underline{R}}$ preconditions $\mathbf{L}_{\underline{G}}$. Our strategy closely follows common strategies for sparsifying undirected graphs via expander decompositions. First we get rid of most of the weighted structure by bucketing by edge weight. We obtain $\tilde{O}(1)$ graphs $\underline{G}^{(i)}$ with close to uniform edge weight such that $\sum_i \underline{G}^{(i)} = \underline{G}$.

We let $H^{(i)}$ denote the unweighted and undirected graph with the same edges as $\underline{G}^{(i)}$. Then we layer $j = 1, \dots, O(\log n)$ undirected and unweighted expander decompositions on this graph, where each of them peels off at least $1/2$ of the remaining edges $E_r^{(i,j)}$. This procedure computes $O(\log n)$ partitions $V_1^{(i,j)}, \dots, V_{k(i,j)}^{(i,j)}$ of the vertex set, such that for each component $V_p^{(i,j)}$ the graph $H^{(i)}[V_p^{(i,j)}]$ is an expander. In the j -th layer, we put the remaining edges of the directed graph $\underline{G}^{(i)}$ that do not go across sets in the partition $V_1^{(i,j)}, \dots, V_{k(i,j)}^{(i,j)}$ into the graph $\underline{G}^{(i,j)}$ and remove them from the set of remaining edges.

The expander structure allows us to sparsify $\underline{G}^{(i,j)}$ via a greedy patching scheme obtaining $\tilde{\underline{G}}^{(i,j)}$. Finally, we sum up across layers and buckets and obtain $\underline{R} = \sum_{i,j} \tilde{\underline{G}}^{(i,j)}$. We leverage the robustness introduced by partial symmetrization to bound the error. See Algorithm 3 for detailed pseudocode. We first state the main lemma of this subsection, which analyses this algorithm.

Lemma 4.7. *Let $\underline{R} = \text{SPARSIFYDIRECTED}(\underline{G}, \gamma)$ for $\gamma \in (0, 1)$ constant. Then $\mathbf{L}_{\underline{G}_2}^+ = (\beta \mathbf{U}_{\mathbf{L}_{\underline{G}}} + \mathbf{L}_{\underline{R}})^+$ is a $1/2$ -approximate pseudoinverse of $\mathbf{L}_{\underline{G}_1} = \mathbf{L}_{\mathcal{U}^{(\beta)}(\underline{G})}$ with respect to $\mathbf{U}_{\mathbf{L}_{\underline{G}_2}}$ for $\beta = \tilde{O}(1) \cdot \text{Exp}(2 \cdot (\log n)^\gamma)$. Further, the graph \underline{R} has $\tilde{O}(n)$ edges and the same in- and out-degrees as \underline{G} .*

The proof of Lemma 4.7 relies on analysing the error incurred by sparsifying each individual expander decomposition, i.e. the cost of replacing $\underline{G}^{(i,j)}$ with $\tilde{\underline{G}}^{(i,j)}$. Then we conclude by just summing up the error. The next lemma carefully analyses the amount of error sparsifying such an expander decomposition creates. It crucially relies on the fact that the expander parts form a disjoint partition, and therefore the error does not scale in the number of expanders.

Lemma 4.8. *In the context of SPARSIFY() in Algorithm 3 we have*

$$\left\| \mathbf{U}_{\mathbf{L}_{\underline{G}}}^{+/2} (\underline{G}^{(i,j)} - \tilde{\underline{G}}^{(i,j)}) \mathbf{U}_{\mathbf{L}_{\underline{G}}}^{+/2} \right\|_2 \leq 128 \cdot \text{Exp}(2 \cdot (\log n)^\gamma)$$

for every edge weight bucket i and expander decomposition layer j .

Algorithm 3: SPARSIFYDIRECTED(\underline{G}, γ) and subroutines SPARSIFY() and PATCH()

```

1 Algorithm SPARSIFYDIRECTED( $\underline{G}, \gamma$ )
2    $P = \left\lceil \log \left( \frac{\omega_{\underline{G}}^{\max}}{\omega_{\underline{G}}^{\min}} \right) \right\rceil$ 
3   for  $i = 1, \dots, P$  do
4      $E^{(i)} = \{e \in E(\underline{G}) : \omega_{\underline{G}}^{\min} \cdot 2^{i-1} \leq \omega_{\underline{G}}(e) < \omega_{\underline{G}}^{\min} \cdot 2^i\}$ 
5      $\tilde{\underline{G}}^{(i)} = \text{SPARSIFY}(\underline{G}^{(i)} = (V(\underline{G}), E^{(i)}, \omega_{\underline{G}}))$ 
6   end
7   return  $\underline{R} = \sum_{i=1}^P \tilde{\underline{G}}^{(i)}$ 
8 Procedure SPARSIFY( $\underline{G}^{(i)}$ )
9   Let  $H^{(i)}$  denote the unweighted and undirected graph with the same edges as  $\mathcal{U}(\underline{G}^{(i)})$ 
10  for  $j = 1, \dots, 10 \log n$  do
11     $E_r^{(i,j)} = E(H^{(i)}) - \bigcup_{l=1}^{j-1} E(H^{(i,l)}); \underline{E}_r^{(i,j)} = E(\underline{G}^{(i)}) - \bigcup_{l=1}^{j-1} E(\underline{G}^{(i,l)})$ 
12     $V_1^{(i,j)}, \dots, V_{k(i,j)}^{(i,j)} = \text{EXPDECOMP}((V(H^{(i)}), E_r^{(i,j)}), \gamma)$ 
13     $E(H^{(i,j)}) = \bigcup_{p=1}^{k(i,j)} \{(u, v) \in E_r^{(i,j)} : u \in V_p^{(i,j)} \wedge v \in V_p^{(i,j)}\}$ 
14     $E(\underline{G}^{(i,j)}) = \bigcup_{p=1}^{k(i,j)} \{(u, v) \in \underline{E}_r^{(i,j)} : u \in V_p^{(i,j)} \wedge v \in V_p^{(i,j)}\}$ 
15     $H^{(i,j)} = (V(H^{(i)}), E(H^{(i,j)})); \underline{G}^{(i,j)} = (V(\underline{G}^{(i)}), E(\underline{G}^{(i,j)}), \omega_{\underline{G}^{(i)}})$ 
16     $\tilde{\underline{G}}^{(i,j)} = \sum_{p=1}^{k(i,j)} \text{PATCH}(\underline{G}^{(i,j)}[V_p^{(i,j)}])$ 
17  end
18  return  $\tilde{\underline{G}}^{(i)} = \sum_{j=1}^{10 \log n} \tilde{\underline{G}}^{(i,j)}$ 
19 Procedure PATCH( $\underline{H}$ )
20  Let  $\mathbf{a}, \mathbf{b} \in \mathbf{R}_{\geq 0}^n$  so that  $\mathbf{a}(v) = \deg_{\underline{H}}^+(v)$  and  $\mathbf{b}(v) = \deg_{\underline{H}}^-(v)$ . // Note  $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1$ .
21   $E(\tilde{\underline{H}}) = \emptyset; \omega_{\tilde{\underline{H}}}(e) = 0$  for all  $e$ .
22  while  $\|\mathbf{a}\| \neq 0$  do
23    Let  $i$  and  $j$  be arbitrary such that  $\mathbf{a}(i) > 0$  and  $\mathbf{b}(j) > 0$ .
24     $w = \min\{\mathbf{a}(i), \mathbf{b}(j)\}; \mathbf{a}(i) = \mathbf{a}(i) - w; \mathbf{b}(j) = \mathbf{b}(j) - w$ 
25     $E(\tilde{\underline{H}}) = E(\tilde{\underline{H}}) \cup \{(i, j)\}; \omega(i, j) = w$ 
26  end
27  return  $\tilde{\underline{H}}$ 

```

Proof. $\underline{G}^{(i,j)}$ and $\tilde{\underline{G}}^{(i,j)}$ are directed graphs with the same in and out degrees since PATCH() in Algorithm 3 preserves degrees exactly. Therefore $\mathbf{1}$ is in both the left and right kernel of $\underline{G}^{(i,j)} - \tilde{\underline{G}}^{(i,j)}$. We apply Lemma 3.14 twice and obtain

$$\begin{aligned}
\left\| \mathbf{U}_{L_{\underline{G}}}^{+/2} (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{U}_{L_{\underline{G}}}^{+/2} \right\|_2 &= 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{y}}{\mathbf{x}^T \mathbf{U}_{L_{\underline{G}}} \mathbf{x} + \mathbf{y}^T \mathbf{U}_{L_{\underline{G}}} \mathbf{y}} \\
&\stackrel{i)}{\leq} 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{\mathcal{U}(\underline{G}^{(i,j)})} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{\mathcal{U}(\underline{G}^{(i,j)})} \mathbf{y}} \\
&\stackrel{ii)}{\leq} \frac{2}{\omega_{\underline{G}}^{\min} \cdot 2^{i-2}} \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{H^{(i,j)}} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{H^{(i,j)}} \mathbf{y}} \\
&= \frac{1}{\omega_{\underline{G}}^{\min} \cdot 2^{i-2}} \left\| \mathbf{L}_{H^{(i,j)}}^{+/2} (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{L}_{H^{(i,j)}}^{+/2} \right\|_2 \quad (2)
\end{aligned}$$

where i) follows since $\mathbf{L}_{\mathcal{U}(\underline{G}^{(i,j)})} \preceq \mathbf{U}_{L_{\underline{G}}}$ because $\mathcal{U}(\underline{G}^{(i,j)})$ is a subgraph of $\mathcal{U}(\underline{G})$ and ii) uses that all edge weights in $\mathcal{U}(\underline{G}^{(i,j)})$ are in $[\omega_{\underline{G}}^{\min} \cdot 2^{i-2}, \omega_{\underline{G}}^{\min} \cdot 2^i]$. Next we can use that the expander parts $V_p^{(i,j)}$ are disjoint in both $\underline{G}^{(i,j)}$ and $\tilde{\underline{G}}^{(i,j)}$ to bound

$$\left\| \mathbf{L}_{H^{(i,j)}}^{+/2} (\mathbf{L}_{\underline{G}^{(i,j)}} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}}) \mathbf{L}_{H^{(i,j)}}^{+/2} \right\|_2 \leq \max_p \left\| \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]}^{+/2} (\mathbf{L}_{\underline{G}^{(i,j)}[V_p^{(i,j)}]} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}[V_p^{(i,j)}]}) \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]}^{+/2} \right\|_2 \quad (3)$$

since the spectral norm of a block diagonal matrix is upper bounded by the maximum spectral norm of a block (See Fact A.1). Then, for every $p \in \{1, \dots, k(i, j)\}$ we have

$$\begin{aligned}
\left\| \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]}^{+/2} (\mathbf{L}_{\underline{G}^{(i,j)}[V_p^{(i,j)}]} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}[V_p^{(i,j)}]}) \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]}^{+/2} \right\|_2 &\stackrel{i)}{=} 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{G}^{(i,j)}[V_p^{(i,j)}]} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}[V_p^{(i,j)}]}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{H^{(i,j)}[V_p^{(i,j)}]} \mathbf{y}} \\
&\stackrel{ii)}{\leq} 8 \cdot 2^{2(\log n)^\gamma} \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{G}^{(i,j)}[V_p^{(i,j)}]} - \mathbf{L}_{\tilde{\underline{G}}^{(i,j)}[V_p^{(i,j)}]}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{G(\mathbf{d}_p^{(i,j)})} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{G(\mathbf{d}_p^{(i,j)})} \mathbf{y}} \quad (4)
\end{aligned}$$

for $\mathbf{d}_p^{(i,j)}$ being the degree vector of $H^{(i,j)}[V_p^{(i,j)}]$, where i) is by Lemma 3.14 and ii) is by the expansion of $H^{(i,j)}[V_p^{(i,j)}]$ and Corollary 3.7. Let $\mathbf{x}, \mathbf{y} \perp \mathbf{1}$ be maximizing the right hand side of the previous inequality. Then, also $\mathbf{x}' = \mathbf{x} - \frac{\mathbf{x}^T \mathbf{d}_p^{(i,j)}}{\|\mathbf{d}_p^{(i,j)}\|_2} \mathbf{1}$ and $\mathbf{y}' = \mathbf{y} - \frac{\mathbf{y}^T \mathbf{d}_p^{(i,j)}}{\|\mathbf{d}_p^{(i,j)}\|_2} \mathbf{1}$ are maximizing. We

obtain

$$\begin{aligned}
2 \frac{\mathbf{x}^T (\mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} - \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{G(d_p^{(i,j)})} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{G(d_p^{(i,j)})} \mathbf{y}} &= 2 \frac{\mathbf{x}'^T (\mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} - \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}}) \mathbf{y}'}{\mathbf{x}'^T \mathbf{L}_{G(d_p^{(i,j)})} \mathbf{x}' + \mathbf{y}'^T \mathbf{L}_{G(d_p^{(i,j)})} \mathbf{y}'} \\
&= 2 \frac{\mathbf{x}'^T (\mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} - \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}}) \mathbf{y}'}{\mathbf{x}'^T \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}} \mathbf{x}' + \mathbf{y}'^T \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}} \mathbf{y}'} \\
&= \left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} (\mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} - \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}}) \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\| \quad (5)
\end{aligned}$$

where the last equality is by Lemma 3.14. Next we upper bound

$$\left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\|$$

and

$$\left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\|.$$

By Lemma 3.15 we have

$$\left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\| \leq \max \left\{ \left\| \mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^+ \right\|_1, \left\| \mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}}^T \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^+ \right\|_1 \right\}.$$

Since for every v , the undirected graph $\omega_{\vec{G}}^{\min} \cdot 2^{i+1} \cdot H^{(i,j)}[V_p^{(i,j)}]$ satisfies

$$\deg_{\omega_{\vec{G}}^{\min} \cdot 2^{i+1} \cdot H^{(i,j)}[V_p^{(i,j)]}}(v) \geq \max \left\{ \deg_{\vec{G}^{(i,j)}[V_p^{(i,j)]}}^+(v), \deg_{\vec{G}^{(i,j)}[V_p^{(i,j)]}}^-(v) \right\}$$

we have

$$\left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \mathbf{L}_{\vec{G}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\| \leq \omega_{\vec{G}}^{\min} \cdot 2^{i+2}. \quad (6)$$

Since we only used the in and out degrees of $\vec{G}^{(i,j)}[V_p^{(i,j)}]$ in the above, and $\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)}]$ has exactly the same degrees, we analogously conclude

$$\left\| \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \mathbf{L}_{\tilde{\vec{G}}^{(i,j)}[V_p^{(i,j)]}} \mathbf{D}_{H^{(i,j)}[V_p^{(i,j)]}}^{+/2} \right\| \leq \omega_{\vec{G}}^{\min} \cdot 2^{i+2}. \quad (7)$$

Chaining inequalities (2), (3), (4), (5), (6) and (7) yields

$$\left\| (\mathbf{U}_{L_{\vec{G}}})^{+/2} (\vec{G}^{(i,j)} - \tilde{\vec{G}}^{(i,j)}) (\mathbf{U}_{L_{\vec{G}}})^{+/2} \right\|_2 \leq 128 \cdot \text{Exp}(2 \cdot (\log n)^\gamma)$$

which concludes our proof. \square

Next we analyse the number of edges of $\tilde{\vec{G}}^{(i,j)}$

Lemma 4.9. $|E(\tilde{G}^{(i,j)})| = O(n)$.

Proof. It is easy to see that the patching routine adds at most $2|V_p^{(i,j)}|$ edges to graph $\tilde{G}^{(i,j)}[V_p^{(i,j)}]$, since each added edge repairs either the desired in-degree or the desired out-degree of a vertex. The result follows since $\sum_p |V_p^{(i,j)}| = n$. \square

We show the main lemma of this subsection by summing up the parts.

Proof of Lemma 4.7. Notice that each expander decomposition peels off half of the edges, and thus every edge is part of a unique expander part by the end of the procedure SPARSIFY() in Algorithm 3. Thus our algorithm exactly preserves the in- and out-degrees of $\underline{G}_1 = \beta \cdot \mathcal{U}(\underline{G}) + \underline{G}$, since each individual patching exactly preserves degrees. Since the graph $\underline{G}_2 = \beta \cdot \mathcal{U}(\underline{G}) + \underline{R}$ remains connected the null-spaces are unaltered. Further, since \underline{R} is the sum of $\tilde{O}(1)$ graphs with $O(n)$ edges the total amount of edges of \underline{R} is bounded by $\tilde{O}(n)$.

Finally, we show the approximation bound. By Lemma 3.13 we have

$$\left\| \mathbf{I}_{\text{im}(\mathcal{U}^{(\beta)}(\underline{G}))} - \mathbf{L}_{\underline{G}_2}^+ \mathbf{L}_{\underline{G}_1} \right\|_{\mathbf{U}_{\underline{G}_2} \rightarrow \mathbf{U}_{\underline{G}_2}} \leq \left\| \mathbf{U}_{\underline{G}_2}^{+/2} (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{U}_{\underline{G}_2}^{+/2} \right\|.$$

We use Lemma 3.14 to obtain

$$\begin{aligned} \left\| \mathbf{U}_{\underline{G}_2}^{+/2} (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{U}_{\underline{G}_2}^{+/2} \right\| &= 2 \max_{\mathbf{x}, \mathbf{y} \neq 0} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{y}}{\mathbf{x}^T (\beta \mathbf{U}_{\underline{L}_{\underline{G}}} + \mathbf{U}_{\underline{L}_{\underline{R}}}) \mathbf{x} + \mathbf{y}^T (\beta \mathbf{U}_{\underline{L}_{\underline{G}}} + \mathbf{U}_{\underline{L}_{\underline{R}}}) \mathbf{y}} \\ &\leq 2 \max_{\mathbf{x}, \mathbf{y} \neq 0} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{y}}{\beta \mathbf{x}^T \mathbf{U}_{\underline{L}_{\underline{G}}} \mathbf{x} + \beta \mathbf{y}^T \mathbf{U}_{\underline{L}_{\underline{G}}} \mathbf{y}}. \end{aligned}$$

Using Lemma 3.14 again we have

$$\begin{aligned} 2 \max_{\mathbf{x}, \mathbf{y} \neq 0} \frac{\mathbf{x}^T (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{y}}{\beta \mathbf{x}^T \mathbf{U}_{\underline{L}_{\underline{G}}} \mathbf{x} + \beta \mathbf{y}^T \mathbf{U}_{\underline{L}_{\underline{G}}} \mathbf{y}} &= \frac{1}{\beta} \left\| \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} (\mathbf{L}_{\underline{R}} - \mathbf{L}_{\underline{G}}) \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} \right\|_2 \\ &= \frac{1}{\beta} \left\| \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} \sum_{i,j} (\mathbf{L}_{\tilde{G}^{(i,j)}} - \mathbf{L}_{\underline{G}^{(i,j)}}) \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} \right\|_2 \\ &\leq \frac{1}{\beta} \sum_{i,j} \left\| \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} (\mathbf{L}_{\tilde{G}^{(i,j)}} - \mathbf{L}_{\underline{G}^{(i,j)}}) \mathbf{U}_{\underline{L}_{\underline{G}}}^{+/2} \right\|_2 \\ &\stackrel{i)}{\leq} \frac{10 \cdot P \cdot \log n \cdot 128 \cdot \text{Exp}(2 \cdot (\log n)^\gamma)}{\beta} \stackrel{ii)}{\leq} \frac{1}{2} \end{aligned}$$

where i) follows from Lemma 4.8 and ii) is by $\beta \geq 128 \cdot 20 \cdot P \cdot \log n \cdot \text{Exp}(2 \cdot (\log n)^\gamma)$. Chaining the inequalities shows the approximation statement and concludes our proof. \square

4.3 Sparsifying the Undirected Part

The final task we have left is to sparsify the undirected graph $\mathcal{U}(\underline{G})$. This can be more or less directly achieved by employing a sparsification theorem presented in [CGL⁺20b]. Mainly for notational convenience in our algorithm, we adapt this sparsification technique to be degree preserving in Appendix B and state the resulting lemma here. See Lemma B.2 for the proof.

Lemma 4.10. (*Degree Preserving Sparsification*) *There is a deterministic algorithm $\text{SPECTRALSPARSIFYDEG}(G, \gamma)$ (Algorithm 10) that given a parameter $\gamma \in (0, 1)$ and an undirected graph $G = (V, E, \omega)$ with n vertices and m edges such that $P := \frac{\max_{e \in E} \omega(e)}{\min_{e \in E} \omega(e)} = \text{poly}(n)$ computes \tilde{G} satisfying*

1. $\text{Exp}(-(\log n)^\gamma) \mathbf{L}_G \preceq \mathbf{L}_{\tilde{G}} \preceq \text{Exp}((\log n)^\gamma) \mathbf{L}_G$
2. $\text{nnz}(\mathbf{A}) = \tilde{O}(n)$

in time

$$\tilde{O}(m^{1+O(1/(\log n)^{\gamma/2})} \cdot (\log m)^{O((\log n)^\gamma)}) = m^{1+o(1)}.$$

The graph \tilde{G} has self loops and exactly the same degrees as G .

Next we apply degree preserving sparsification together with an appropriate scaling to sparsify the undirected part.

Lemma 4.11. *There exists a routine $\tilde{G} = \text{SPECTRALSPARSIFYDEG}(\mathcal{U}(\underline{G}), \gamma)$ (Algorithm 10) that given the undirected graph $\mathcal{U}(\underline{G})$ computes an undirected graph \tilde{G} with $\tilde{O}(n)$ edges so that $\mathbf{L}_{\tilde{G}_3}^+ = (\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{L}_{\underline{R}})^+$ is an $\left(1 - \frac{1}{2 \cdot \text{Exp}(4 \cdot (\log n)^\gamma)}\right)$ -approximate pseudoinverse of $\mathbf{L}_{\underline{G}_2} = \beta \mathbf{U}_{\underline{L}_{\underline{G}}} + \mathbf{L}_{\underline{R}}$ with respect to $\mathbf{U}_{\underline{L}_{\underline{G}_3}}$. The degrees of \tilde{G} and $\mathcal{U}(\underline{G})$ are the same.*

Proof. The sparsity and degree preservation follows directly from Lemma 4.10. To show the approximation property, we use Lemma 3.13 and obtain

$$\left\| \mathbf{I}_{\text{im}(\mathbf{L}_{\underline{G}})} - \mathbf{L}_{\underline{G}_3}^+ \mathbf{L}_{\underline{G}_2} \right\|_{\mathbf{U}_{\underline{L}_{\underline{G}_3}} \rightarrow \mathbf{U}_{\underline{L}_{\underline{G}_3}}} \leq \left\| \mathbf{U}_{\underline{L}_{\underline{G}_3}}^{+/2} \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} - \beta \mathbf{L}_{\mathcal{U}(\underline{G})} \right) \mathbf{U}_{\underline{L}_{\underline{G}_3}}^{+/2} \right\|_2. \quad (8)$$

Then we use Lemma 3.14 twice with $\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} \preceq \frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{U}_{\underline{L}_{\underline{R}}}$ to obtain

$$\begin{aligned} \left\| \mathbf{U}_{\underline{L}_{\underline{G}_3}}^{+/2} \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} - \beta \mathbf{L}_{\mathcal{U}(\underline{G})} \right) \mathbf{U}_{\underline{L}_{\underline{G}_3}}^{+/2} \right\|_2 &= 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} - \beta \mathbf{L}_{\mathcal{U}(\underline{G})} \right) \mathbf{y}}{\mathbf{x}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{U}_{\underline{L}_{\underline{R}}} \right) \mathbf{x} + \mathbf{y}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{U}_{\underline{L}_{\underline{R}}} \right) \mathbf{y}} \\ &\leq 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} - \beta \mathbf{L}_{\mathcal{U}(\underline{G})} \right) \mathbf{y}}{\mathbf{x}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} \right) \mathbf{x} + \mathbf{y}^T \left(\frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} \right) \mathbf{y}} \\ &= \left\| \mathbf{L}_{\tilde{G}}^{+/2} (\mathbf{L}_{\tilde{G}} - \eta \mathbf{L}_{\mathcal{U}(\underline{G})}) \mathbf{L}_{\tilde{G}}^{+/2} \right\|. \end{aligned} \quad (9)$$

Next we compute

$$\left\| \mathbf{L}_{\tilde{G}}^{+/2} (\mathbf{L}_{\tilde{G}} - \eta \mathbf{L}_{\mathcal{U}(\underline{G})}) \mathbf{L}_{\tilde{G}}^{+/2} \right\|_2 = \left\| \mathbf{I}_{\text{im}(\mathbf{L}_{\tilde{G}})} - \eta \mathbf{L}_{\tilde{G}}^{+/2} \mathbf{L}_{\mathcal{U}(\underline{G})} \mathbf{L}_{\tilde{G}}^{+/2} \right\|_2. \quad (10)$$

We use the standard strategy of bounding the square. Let $\mathbf{M} := \mathbf{L}_{\tilde{G}}^{+/2} \mathbf{L}_{\mathcal{U}(\underline{G})} \mathbf{L}_{\tilde{G}}^{+/2}$ be a shorthand. Then we have

$$\begin{aligned} \left\| \mathbf{I}_{\text{im}(\mathbf{L}_{\tilde{G}})} - \eta \mathbf{M} \right\|_2 &= \max_{\mathbf{x} \in \text{im}(\mathbf{L}_{\tilde{G}}): \|\mathbf{x}\|_2=1} \mathbf{x}^T (\mathbf{I}_{\text{im}(\mathbf{L}_{\tilde{G}})} - \eta \mathbf{M})^T (\mathbf{I}_{\text{im}(\mathbf{L}_{\tilde{G}})} - \eta \mathbf{M}) \mathbf{x} \\ &= 1 + \max_{\mathbf{x} \in \text{im}(\mathbf{L}_{\tilde{G}}): \|\mathbf{x}\|_2=1} -2\eta \mathbf{x}^T \mathbf{M} \mathbf{x} + \eta^2 \mathbf{x}^T \mathbf{M}^2 \mathbf{x} \\ &\leq 1 - 2\eta \lambda_*(\mathbf{M}) + \eta^2 \|\mathbf{M}\|_2^2 \end{aligned}$$

where $\lambda_*(\mathbf{M})$ denotes the smallest non-zero eigenvalue of \mathbf{M} . We first lower bound $\lambda_*(\mathbf{M})$. By Lemma 4.10 we have

$$\begin{aligned} \frac{1}{2^{(\log n)^\gamma}} \mathbf{L}_{\tilde{G}} &\preceq \mathbf{L}_{\mathcal{U}(\underline{G})} \\ \frac{1}{2^{(\log n)^\gamma}} \mathbf{I}_{\text{im}(\tilde{G})} &\preceq \mathbf{L}_{\tilde{G}}^{+/2} \mathbf{L}_{\mathcal{U}(\underline{G})} \mathbf{L}_{\tilde{G}}^{+/2} \end{aligned}$$

and thus $\text{Exp}(-(\log n)^\gamma) \leq \lambda_*(\mathbf{M})$. We obtain $\|\mathbf{M}\|_2 \leq \text{Exp}(2 \cdot (\log n)^\gamma)$ in an analogous way from Lemma 4.10. Then, we use $\eta = \text{Exp}(-3(\log n)^\gamma) \leq \frac{\lambda_*(\mathbf{M})}{\|\mathbf{M}\|_2^2}$ to conclude.

$$\left\| \mathbf{I}_{\text{im}(\mathbf{L}_{\tilde{G}})} - \eta \mathbf{M} \right\|_2 \leq 1 - \frac{1}{2 \cdot \text{Exp}(4 \cdot (\log n)^\gamma)} \quad (11)$$

where we use $\sqrt{1-\epsilon} \leq 1 - \epsilon/2$ for $\epsilon \in (0, 1)$. Chaining inequalities (8), (9), (10) and (11) shows the desired approximate pseudoinverse property and finishes the proof. \square

4.4 Loss in Condition Number

The overarching goal of the recursion of a squaring solver is to reduce the condition number of the undirectification of the graph. When a constant condition number is reached, the recursion terminates. However, since our sparsification has relatively low accuracy, we have to ensure that our losses do not outweigh our gains when we call it. In this subsection we bound the loss in condition number incurred by our global sparsification routine. We address the loss in condition number incurred by all of the three steps. To do so we consider spectral bounds.

By Partial Symmetrization. The first part is the easiest. Since $\mathbf{U}_{\mathbf{L}_{\mathcal{U}(\beta)(\underline{G})}} = (1 + \beta) \mathbf{U}_{\mathbf{L}_{\underline{G}}}$ β -partial-symmetrization does not change the condition number. Further we observe

Observation 4.12. $\mathbf{U}_{\mathbf{L}_{\underline{G}}} \preceq \mathbf{U}_{\mathbf{L}_{\mathcal{U}(\beta)(\underline{G})}} \preceq (1 + \beta) \mathbf{U}_{\mathbf{L}_{\underline{G}}}$

By Sparsifying the Directed Part. Next we analyse the loss incurred by patching undirected expanders. This part crucially relies on a spectral upper bound on $\mathbf{U}_{\mathbf{L}_{\vec{R}}}$. We show the following lemma

Lemma 4.13. $\beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} \preceq \beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}} \preceq (\beta + \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma)) \mathbf{U}_{\mathbf{L}_{\vec{G}}}$

Proof. The first inequality, $\beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} \preceq \beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}}$, is clear since $\mathbf{0} \preceq \mathbf{L}_{\mathcal{U}(\vec{R})} = \mathbf{U}_{\mathbf{L}_{\vec{R}}}$. To upper bound $\mathbf{U}_{\mathbf{L}_{\vec{R}}}$ we compute

$$\begin{aligned} \left\| \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} (\mathbf{U}_{\mathbf{L}_{\vec{R}}} - \mathbf{U}_{\mathbf{L}_{\vec{G}}}) \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \right\|_2 &\leq 2 \left\| \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} (\mathbf{L}_{\vec{G}} - \mathbf{L}_{\vec{R}}) \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \right\|_2 \\ &= 2 \left\| \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \sum_{i,j} (\mathbf{L}_{\vec{G}}^{(i,j)} - \mathbf{L}_{\vec{G}}^{(i,j)}) \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \right\|_2 \\ &\leq 2 \sum_{i,j} \left\| \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} (\mathbf{L}_{\vec{G}}^{(i,j)} - \mathbf{L}_{\vec{G}}^{(i,j)}) \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \right\|_2 \\ &\leq \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma) \end{aligned}$$

where we use that the sum is over $\tilde{O}(1)$ edge weight buckets i and $\tilde{O}(1)$ expander decomposition layers j . We conclude

$$\begin{aligned} \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} (\mathbf{U}_{\mathbf{L}_{\vec{R}}} - \mathbf{U}_{\mathbf{L}_{\vec{G}}}) \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} &\preceq \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma) \mathbf{I}_{\text{im}(\mathbf{L}_{\vec{G}})} \\ \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} \mathbf{U}_{\mathbf{L}_{\vec{R}}} \mathbf{U}_{\mathbf{L}_{\vec{G}}}^{+/2} - \mathbf{I}_{\text{im}(\mathbf{L}_{\vec{G}})} &\preceq \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma) \mathbf{I}_{\text{im}(\mathbf{L}_{\vec{G}})} \\ \mathbf{U}_{\mathbf{L}_{\vec{R}}} &\preceq \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma) \mathbf{U}_{\mathbf{L}_{\vec{G}}}. \end{aligned}$$

This concludes our proof. \square

By Sparsifying the Undirected Part. Finally, we analyse the loss incurred by undirected sparsification. This part directly follows from the approximation guarantee of Lemma 4.10.

Lemma 4.14. For $\tilde{G} = \text{SPECTRALSPARSIFYDEG}(\mathcal{U}(\vec{G}), \gamma)$ we have

$$\text{Exp}(-4(\log n)^\gamma) \mathbf{U}_{\mathbf{L}_{\vec{G}_2}} \preceq \mathbf{U}_{\mathbf{L}_{\vec{G}_3}} \preceq \text{Exp}(4(\log n)^\gamma) \mathbf{U}_{\mathbf{L}_{\vec{G}_2}}$$

Proof. Recall that $\mathbf{U}_{\mathbf{L}_{\vec{G}_2}} = \beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}}$ and $\mathbf{U}_{\mathbf{L}_{\vec{G}_3}} = \frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}}$. We show the following stronger statement

$$\text{Exp}(-4(\log n)^\gamma) \beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}} \preceq \frac{\beta}{\eta} \mathbf{L}_{\tilde{G}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}} \preceq \text{Exp}(4(\log n)^\gamma) \beta \mathbf{U}_{\mathbf{L}_{\vec{G}}} + \mathbf{U}_{\mathbf{L}_{\vec{R}}}.$$

After subtracting $\mathbf{U}_{\mathbf{L}_{\vec{R}}}$ and dividing by β we have

$$\text{Exp}(-4(\log n)^\gamma) \mathbf{U}_{\mathbf{L}_{\vec{G}}} \preceq \frac{1}{\eta} \mathbf{L}_{\tilde{G}} \preceq \text{Exp}(4(\log n)^\gamma) \mathbf{U}_{\mathbf{L}_{\vec{G}}}$$

which directly follows from Lemma 4.10. \square

4.5 Proof of Lemma 4.4

Now that we have assembled the pieces we are ready to prove Lemma 4.4, the main lemma of this section.

Proof of Lemma 4.4. We show each point in the enumeration separately.

1. The statement for $i = 1$ is by Lemma 4.6 and $\beta = \tilde{O}(1) \cdot \text{Exp}(2(\log n)^\gamma)$, for $i = 2$ it is by Lemma 4.7 and for $i = 3$ it is by Lemma 4.11.
2. The statement for $i = 1$ is by Observation 4.12, for $i = 2$ it is by Lemma 4.13 and for $i = 3$ it is by Lemma 4.14.
3. Clearly, \underline{G}_1 has at most twice as many edges as $\underline{G} = \underline{G}_0$. Further, $|E(\underline{G}_2)| \leq |E(\underline{G}_1)| + |E(\underline{R})|$ and $|E(\underline{R})| = \tilde{O}(n)$ by Lemma 4.7. Finally $|E(\underline{G}_3)| \leq |E(\tilde{G})| + |E(\underline{R})| = \tilde{O}(n)$ by Lemma 4.11 and Lemma 4.7. .
4. Adding β times the undirectification to an Eulerian graph increases the in- and out- degrees by exactly a factor of $(1 + \beta)$. Then the statement follows from the degree preservation of our sparsification routines shown in Lemma 4.7 and Lemma 4.11 and the extra scaling by $\frac{1}{\eta}$ of the undirected part of \underline{G}_3

Finally, the runtime follows from Corollary 3.10. \square

5 Approximately Squaring Directed Laplacians

Given a directed Eulerian Laplacian $\mathbf{L}_{\underline{G}} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T$ we call $\mathbf{L}_{\underline{G}^2} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T \mathbf{D}_{\underline{G}}^{-1} \mathbf{A}_{\underline{G}}^T$ its square. The squaring operation not only preserves degrees, but also Eulerianness. In this section we introduce a deterministic algorithm for high accuracy sparsification of the square of an Eulerian Laplacian in roughly $\text{nnz}(\mathbf{L}_{\underline{G}})$ time. Since squaring can drastically increase the density, the runtime of this routine sometimes has to be sub-linear in the size of the graph it is approximating. Thus it is clear that we can only afford work with implicit representations of $\mathbf{L}_{\underline{G}^2}$. Our routines are based on and inspired by [KLP⁺15] and [PS21]. A similar sparsification routine is developed in [AKM⁺20]. We first state the main lemma of this section.

Lemma 5.1. *For every strongly connected Eulerian Laplacian $\mathbf{L}_{\underline{G}} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T$ the algorithm $\text{SPARSE SQUARE}(\underline{G}, \epsilon)$ computes a strongly connected Eulerian graph $\tilde{\underline{G}}^2$ in time $O(\text{nnz}(\mathbf{L}_{\underline{G}})^{1+o(1)}/\epsilon^4)$ so that $\mathbf{L}_{\tilde{\underline{G}}^2} = \mathbf{D}_{\tilde{\underline{G}}^2} - \mathbf{A}_{\tilde{\underline{G}}^2}^T$ is an ϵ -graph-approximation of $\mathbf{L}_{\underline{G}^2} = \mathbf{D}_{\underline{G}} - \mathbf{A}_{\underline{G}}^T \mathbf{D}_{\underline{G}}^{-1} \mathbf{A}_{\underline{G}}^T$, $\text{nnz}(\mathbf{L}_{\tilde{\underline{G}}^2}) = O(\text{nnz}(\mathbf{L}_{\underline{G}})/\epsilon^4)$ and degrees are preserved.*

As evident from the pseudocode of $\text{SPARSE SQUARE}()$ (Algorithm 4) our approach to implicitly sparsifying $\mathbf{L}_{\underline{G}^2}$ breaks down its adjacency matrix into rank one contributions

$$\mathbf{A}_{\underline{G}}^T \mathbf{D}_{\underline{G}}^{-1} \mathbf{A}_{\underline{G}}^T = \sum_{i=1}^n \underbrace{\mathbf{D}_{\underline{G}}^{-1}(i, i) (\mathbf{A}_{\underline{G}}(i, :))^T \cdot (\mathbf{A}_{\underline{G}}(:, i))^T}_{=: \mathbf{A}_{\tilde{\underline{G}}^2, i}}$$

as commonplace in randomized squaring sparsification algorithms [PS21]. Then SPARSEPRODUCT() (Algorithm 4) computes $\mathbf{A}_{\underline{H}_i}$, a suitable sparse approximation of $\mathbf{A}_{\underline{G}_i}$. The full adjacency matrix is obtained by simply summing up these contributions. Note that SPARSEPRODUCT() will preserve in- and out-degrees exactly, ensuring that the resulting approximation preserves degrees, which in turn ensures $\mathbf{L}_{\underline{G}^2}$ being Eulerian.

Algorithm 4: SPARSE SQUARE(\underline{G}, ϵ)

```

1 for  $i = 1, \dots, n$  do
2   |  $\mathbf{A}_{\underline{H}_i} = \text{SPARSEPRODUCT}(\mathbf{A}_{\underline{G}}(i, :), \mathbf{A}_{\underline{G}}(:, i), \epsilon)$ 
3 end
4  $\mathbf{A}_{\underline{\tilde{G}}^2} = \sum_{i=1}^n \mathbf{A}_{\underline{H}_i}$ 
5 return  $\underline{\tilde{G}}^2$ 

```

5.1 Deterministically Sparsifying Bipartite Product Graphs

High accuracy spectral sparsifiers for undirected bipartite product graphs are known to the literature [KLP⁺15]. We will first adapt these to our setting by showing that a simple greedy patching scheme can be employed to ensure the conservation of degrees. Then we will directly use sparsified bipartite product graphs to build SPARSEPRODUCT() (Algorithm 4), our routine for sparsifying $\mathbf{L}_{\underline{G}_i}$. We start by defining the central object of this subsection.

Definition 5.2. For $\mathbf{a} \in \mathbb{R}_{\geq 0}^n$ and $\mathbf{b} \in \mathbb{R}_{\geq 0}^n$ positive vectors so that $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1 = d$, let

$$\mathbf{L}_{G(\mathbf{a}, \mathbf{b})} := \begin{pmatrix} \text{diag}(\mathbf{a}) & \mathbf{0} \\ \mathbf{0} & \text{diag}(\mathbf{b}) \end{pmatrix} - \begin{pmatrix} \mathbf{0} & \frac{\mathbf{b}\mathbf{a}^T}{d} \\ \frac{\mathbf{a}\mathbf{b}^T}{d} & \mathbf{0} \end{pmatrix}$$

be the Laplacian of the undirected bipartite product graph $G(\mathbf{a}, \mathbf{b})$ of \mathbf{a} and \mathbf{b} .

Definition 5.3. We call an undirected graph Laplacian

$$\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} = \begin{pmatrix} \text{diag}(\mathbf{a}) & \mathbf{0} \\ \mathbf{0} & \text{diag}(\mathbf{b}) \end{pmatrix} - \begin{pmatrix} \mathbf{0} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{0} \end{pmatrix}$$

with $\mathbf{C} \in \mathbb{R}_{\geq 0}^{n \times n}$ an ϵ -bipartite-sparsifier of $\mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$ if

$$(1 - \epsilon)\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} \preceq \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \preceq (1 + \epsilon)\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$$

and $\text{nnz}(\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) = O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))/\epsilon^4)$.

Next we introduce the previously known spectral sparsification lemma we will use.

Lemma 5.4 (Lemma G.16 in [KLP⁺15]). *There is a routine WEIGHTEDBIPARTITEEXPANDER($\mathbf{a}, \mathbf{b}, \epsilon$) such that for any vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$ and a parameter $\epsilon \in (0, 1/2]$, it returns in time $O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))\epsilon^{-4})$ a bipartite graph $\tilde{G}(\mathbf{a}, \mathbf{b})$ on the same bipartition as $G(\mathbf{a}, \mathbf{b})$ with $O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))\epsilon^{-4})$ edges such that*

$$(1 - \epsilon)\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} \preceq \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \preceq (1 + \epsilon)\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$$

The routine `SparseBipartite()` (Algorithm 5) combines `WeightedBipartiteExpander()` from Lemma 5.4 with the greedy patching procedure `BipartitePatching()` (Algorithm 5) to construct an ϵ -bipartite-sparsifier. We first show the desired runtime and preservation of degrees.

Algorithm 5: `SparseBipartite`($\mathbf{a}, \mathbf{b}, \epsilon$) and `BipartitePatching`($\mathbf{a}, \mathbf{b}, \mathbf{d}^A, \mathbf{d}^B$)

```

1 Algorithm SparseBipartite( $\mathbf{a}, \mathbf{b}, \epsilon$ )
2    $\epsilon' = \epsilon/128$ 
3    $H_1 = \text{WeightedBipartiteExpander}(\mathbf{a}, \mathbf{b}, \epsilon')$ 
4    $H_2 = H_1/(1 + \epsilon')$ 
5    $\mathbf{d} = \text{diag}(\mathbf{D}_{H_2})$ 
6    $\mathbf{L}_R = \text{BipartitePatching}(\mathbf{a}, \mathbf{b}, \mathbf{d}(1:n), \mathbf{d}((n+1):2n))$ 
7   return  $\mathbf{L}_{H_2} + \mathbf{L}_R$ 
8 Procedure BipartitePatching( $\mathbf{a}, \mathbf{b}, \mathbf{d}^A, \mathbf{d}^B$ )
9    $\tilde{\mathbf{a}} = \mathbf{a} - \mathbf{d}^A$ 
10   $\tilde{\mathbf{b}} = \mathbf{b} - \mathbf{d}^B$ 
11   $\mathbf{L}_R = \mathbf{0}^{2n \times 2n}$ 
12  while  $\tilde{\mathbf{a}} \neq \mathbf{0}$  do
13    Let  $i$  and  $j$  be arbitrary such that  $\tilde{\mathbf{a}}(i) > 0$  and  $\tilde{\mathbf{b}}(j) > 0$ .
14     $w = \min\{\tilde{\mathbf{a}}(i), \tilde{\mathbf{b}}(j)\}$ 
15     $\tilde{\mathbf{a}}(i) = \tilde{\mathbf{a}}(i) - w$ 
16     $\tilde{\mathbf{b}}(j) = \tilde{\mathbf{b}}(j) - w$ 
17     $\mathbf{L}_R = \mathbf{L}_R + w(\mathbf{e}_i - \mathbf{e}_{j+n})(\mathbf{e}_i - \mathbf{e}_{j+n})^T$ 
18  end
19  return  $\mathbf{L}_R$ 

```

Lemma 5.5. *The routine $\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} = \text{SparseBipartite}(\mathbf{a}, \mathbf{b}, \epsilon)$ with $\mathbf{a}, \mathbf{b} \in R_{\geq 0}^n$ so that $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1$, and $\epsilon \leq 1/2$ terminates in time $O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))/\epsilon^4)$ and ensures $\mathbf{D}_{\tilde{G}(\mathbf{a}, \mathbf{b})} = \mathbf{D}_{G(\mathbf{a}, \mathbf{b})}$.*

Proof. Since for any graph G we have $\mathbf{e}_v^T \mathbf{L}_G \mathbf{e}_v = \deg_G(v)$ for all v it is easy to see that the spectral approximation of `WeightedBipartiteExpander()` from Lemma 5.4 preserves degrees in the sense that for all v

$$(1 - \epsilon') \deg_{G(\mathbf{a}, \mathbf{b})}(v) \leq \deg_{H_1} \leq (1 + \epsilon') \deg_{G(\mathbf{a}, \mathbf{b})}.$$

From this it is immediate that $\deg_{H_2}(v) \leq \deg_{G(\mathbf{a}, \mathbf{b})}(v)$. Then the routine `BipartitePatching()` iteratively computes the Laplacian of a bipartite patching graph so that $\deg_R(v) = \deg_{G(\mathbf{a}, \mathbf{b})}(v) - \deg_{H_2}(v)$ for all v . It is clear that both H_2 and $G(\mathbf{a}, \mathbf{b})$ being bipartite ensures the invariant that $\|\tilde{\mathbf{a}}\|_1 = \|\tilde{\mathbf{b}}\|_1$ throughout the execution of `BipartitePatching()` and $\tilde{\mathbf{a}} \neq \mathbf{0} \implies \tilde{\mathbf{b}} \neq \mathbf{0}$. Since each edge (u, v) added to R either ensures $\tilde{\mathbf{a}}(u) = 0$ or $\tilde{\mathbf{b}}(v) = 0$ the routine terminates after at most $\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b})$ edges are added since $\text{nnz}(\tilde{\mathbf{a}}) + \text{nnz}(\tilde{\mathbf{b}}) \leq \text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b})$ initially. The result follows from these observations and Lemma 5.4. \square

To conclude this subsection we would like to use Lemma 5.4 to show that `SparseBipartite()` (Algorithm 5) computes an ϵ -bipartite-sparsifier of $\mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$. The remaining obstruction is the quan-

tification of the error introduced by the `BIPARTITEPATCHING()` (Algorithm 5). Our analysis of this error relies on the high expansion of $G(\mathbf{a}, \mathbf{b})$.

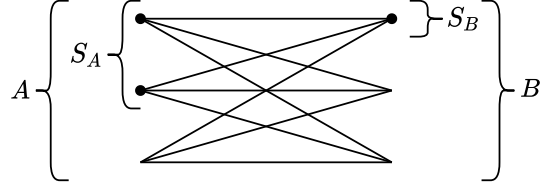


Figure 5: This figure depicts a bipartite graph $G(\mathbf{a}, \mathbf{b})$, where the set S as defined in Lemma 5.6 consists of the vertices highlighted with black dots. Then the set is split into parts S_A and S_B as indicated.

Lemma 5.6. $G(\mathbf{a}, \mathbf{b})$ is a $1/2$ -expander.

Proof. Consider any cut $\emptyset \subset S \subset V$. Without loss of generality, let $\text{vol}_{G(\mathbf{a}, \mathbf{b})}(S) \leq \text{vol}_{G(\mathbf{a}, \mathbf{b})}(V \setminus S)$. We distinguish between the two parts A and B of the bipartite graph and let $S_A \subseteq A$ and $S_B \subseteq B$ so that $S_A \cup S_B = S$. We first introduce some terminology that will prove useful:

$$d := \|\mathbf{a}\|_1 (= \|\mathbf{b}\|_1) \quad t_{S_A}^A := \sum_{v \in S_A} \mathbf{a}(v) \quad t_{S_B}^B := \sum_{v \in S_B} \mathbf{b}(v).$$

See Figure 5 for a drawing of the situation. Then we have

$$\begin{aligned} \delta_{G(\mathbf{a}, \mathbf{b})}(S) &= \sum_{u \in S_A} \sum_{v \in B \setminus S_B} \frac{\mathbf{a}(u)\mathbf{b}(v)}{\|\mathbf{a}\|_1} + \sum_{v \in S_B} \sum_{u \in A \setminus S_A} \frac{\mathbf{a}(u)\mathbf{b}(v)}{\|\mathbf{a}\|_1} \\ &= \frac{1}{d} (t_{S_A}^A (d - t_{S_B}^B) + t_{S_B}^B (d - t_{S_A}^A)) \end{aligned}$$

and

$$\begin{aligned} \text{vol}_{G(\mathbf{a}, \mathbf{b})}(S) &= \sum_{u \in S_A} \sum_{v \in B} \frac{\mathbf{a}(u)\mathbf{b}(v)}{\|\mathbf{a}\|_1} + \sum_{v \in S_B} \sum_{u \in A} \frac{\mathbf{a}(u)\mathbf{b}(v)}{\|\mathbf{a}\|_1} \\ &= t_{S_A}^A + t_{S_B}^B. \end{aligned}$$

Finally we show

$$\begin{aligned} \frac{\delta_{G(\mathbf{a}, \mathbf{b})}(S)}{\text{vol}_{G(\mathbf{a}, \mathbf{b})}(S)} \geq 1/2 &\iff (t_{S_A}^A (d - t_{S_B}^B) + t_{S_B}^B (d - t_{S_A}^A)) \geq \frac{d}{2} (t_{S_A}^A + t_{S_B}^B) \\ &\iff \frac{d}{2} (t_{S_A}^A + t_{S_B}^B) \geq 2t_{S_B}^B t_{S_A}^A. \end{aligned}$$

Using $\text{vol}_{G(\mathbf{a}, \mathbf{b})}(S) \leq \text{vol}_{G(\mathbf{a}, \mathbf{b})}(V \setminus S)$ and $d = \text{vol}_{G(\mathbf{a}, \mathbf{b})}(V)$ we have $d \geq t_{S_A}^A + t_{S_B}^B$ and thus

$$d(t_{S_A}^A + t_{S_B}^B) \geq (t_{S_A}^A)^2 + (t_{S_B}^B)^2 + 2t_{S_A}^A t_{S_B}^B \geq 4t_{S_A}^A t_{S_B}^B$$

by since for any real numbers a and b we have $a^2 + b^2 \geq 2ab$. □

Using the expansion of $\mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$, we give a spectral upper bound on the error by patching next.

Lemma 5.7. *In the context of $\text{SPARSEBIPARTITE}(\mathbf{a}, \mathbf{b}, \epsilon)$ with $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$, $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1$, we have $\mathbf{L}_R \preceq 128\epsilon' \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$.*

Proof. As observed in the proof of Lemma 5.5 we have $\deg_R(v) = \deg_{G(\mathbf{a}, \mathbf{b})}(v) - \deg_{H_2}(v)$ for all v . Since $\deg_{H_2}(v) \geq \frac{1-\epsilon'}{1+\epsilon'} \deg_{G(\mathbf{a}, \mathbf{b})}(v)$ by Lemma 5.4 we have

$$\deg_R(v) \leq \deg_{G(\mathbf{a}, \mathbf{b})} - \frac{1-\epsilon'}{1+\epsilon'} \deg_{G(\mathbf{a}, \mathbf{b})}(v) \leq 4\epsilon' \deg_{G(\mathbf{a}, \mathbf{b})}(v). \quad (12)$$

In the following, let $\mathbf{d} = \text{diag}(\mathbf{D}_{G(\mathbf{a}, \mathbf{b})})$ be the degree vector of $G(\mathbf{a}, \mathbf{b})$. We use Lemma 3.14 and arrive at

$$\left\| \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}^{+/2} \mathbf{L}_R \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}^{+/2} \right\|_2 = 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T \mathbf{L}_R \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \mathbf{y}} \quad (13)$$

Let \mathbf{x} and \mathbf{y} be maximizing the right hand side of (13) and $\mathbf{x}, \mathbf{y} \perp \mathbf{1}$. Then, $\mathbf{x}' = \mathbf{x} - \frac{\mathbf{x}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}$ and $\mathbf{y}' = \mathbf{y} - \frac{\mathbf{y}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}$ are also maximizing since the all-ones vector is in the kernel of Laplacian matrices. By Corollary 3.7 and Lemma 5.6 we have $\frac{1}{16}(\text{diag}(\mathbf{d}) - \frac{\mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|_1}) \preceq \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$. We calculate

$$\begin{aligned} \left\| \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}^{+/2} \mathbf{L}_R \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}^{+/2} \right\|_2 &= 2 \frac{\mathbf{x}'^T \mathbf{L}_R \mathbf{y}'}{\mathbf{x}'^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \mathbf{x}' + \mathbf{y}'^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \mathbf{y}'} \\ &\leq 32 \frac{\mathbf{x}'^T \mathbf{L}_R \mathbf{y}'}{\mathbf{x}'^T (\text{diag}(\mathbf{d}) - \frac{\mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|_1}) \mathbf{x}' + \mathbf{y}'^T (\text{diag}(\mathbf{d}) - \frac{\mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|_1}) \mathbf{y}'} \\ &= 32 \frac{\mathbf{x}'^T \mathbf{L}_R \mathbf{y}'}{\mathbf{x}'^T \text{diag}(\mathbf{d}) \mathbf{x}' + \mathbf{y}'^T \text{diag}(\mathbf{d}) \mathbf{y}'} \\ &\leq 16 \left\| \text{diag}(\mathbf{d})^{+/2} \mathbf{L}_R \text{diag}(\mathbf{d})^{+/2} \right\| \end{aligned}$$

where the last line follows from another application of Lemma 3.14. By Lemma 3.15 and (12) we have

$$\left\| \text{diag}(\mathbf{d})^{+/2} \mathbf{L}_R \text{diag}(\mathbf{d})^{+/2} \right\| \leq \left\| \mathbf{L}_R \mathbf{D}_{G(\mathbf{a}, \mathbf{b})} \right\|_1 \leq 8\epsilon'.$$

We conclude

$$\mathbf{L}_R \preceq 128\epsilon' \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}.$$

□

Finally, we assemble the pieces and show that $\text{SPARSEBIPARTITE}()$ sparsifies $G(\mathbf{a}, \mathbf{b})$ while preserving degrees.

Lemma 5.8. *There is a routine $\text{SPARSEBIPARTITE}(\mathbf{a}, \mathbf{b}, \epsilon)$ that given $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$ and a parameter $\epsilon \leq 1/2$ computes in $O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))\epsilon^{-4})$ time an ϵ -bipartite sparsifier $\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$ of $\mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$.*

Proof. The runtime and degree preservation is clear from Lemma 5.5. Further, we have

$$\frac{1 - \epsilon'}{1 + \epsilon'} \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \preceq \mathbf{L}_{H_2} \preceq \mathbf{L}_{H_2} + \mathbf{L}_R = \mathbf{L}_H$$

and $\mathbf{L}_{H_2} \preceq \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$ by Lemma 5.4. From Lemma 5.7 we conclude

$$\frac{1 - \epsilon'}{1 + \epsilon'} \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \preceq \mathbf{L}_H \preceq (1 + 128\epsilon') \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}.$$

The desired approximation follows since $\epsilon' = \epsilon/128$. \square

5.2 Sparsifying Directed Product Graphs via Bipartite Product Graphs

The subroutine SPARSEPRODUCT() (Algorithm 4) exploits relationships between directed graphs and bipartite graphs, allowing us to reduce our problem to the sparsification problem on bipartite product graphs we solved in the previous subsection. Our object of interest in this subsection is defined as follows.

Definition 5.9. For $\mathbf{a} \in \mathbb{R}_{\geq 0}^n$ and $\mathbf{b} \in \mathbb{R}_{\geq 0}^n$ positive vectors so that $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1 = d$, let

$$\mathbf{L}_{\vec{G}(\mathbf{a}, \mathbf{b})} := \text{diag}(\mathbf{b}) - \frac{\mathbf{a}\mathbf{b}^T}{d}$$

denote the directed product graph of \mathbf{a} and \mathbf{b} .

Inspecting $\mathbf{L}_{\vec{G}(\mathbf{a}, \mathbf{b})}$ from Definition 5.2 yields a natural approach to sparsifying $\mathbf{L}_{\vec{G}(\mathbf{a}, \mathbf{b})}$: Compute a ϵ -bipartite sparsifier \mathbf{L}_H and then simply retrieve the adjacency matrix from its bottom left block. This is exactly what SPARSEPRODUCT() (Algorithm 4) does. Somewhat surprisingly, this approach directly yields a high accuracy approximation of $\mathbf{L}_{\vec{G}(\mathbf{a}, \mathbf{b})}$ with a mere constant factor loss in approximation. The argument relies on the fact that $G(\mathbf{a}, \mathbf{b})$ is a constant expander (Lemma 5.6), which would be a serious obstruction for extending this strategy to sparsifying general directed graphs. For a graphical explanation of the relationship between directed graphs and bipartite graphs see Figure 6.

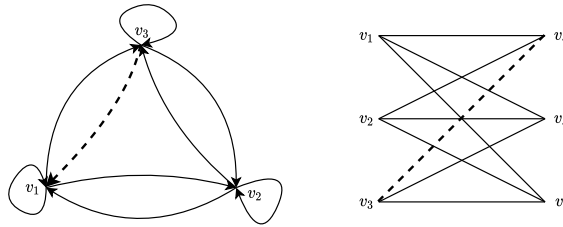


Figure 6: A directed graph naturally has an associated bipartite graph on twice as many vertices. The directed edge (i, j) is then associated with the undirected edge $(i, j + |V|)$. For example, in the figure the two dashed edges are associated.

Lemma 5.10. For $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$ so that $\|\mathbf{a}\|_1 = \|\mathbf{b}\|_1$ and $\epsilon \leq 1/2$ the routine SPARSEPRODUCT($\mathbf{a}, \mathbf{b}, \epsilon$) (Algorithm 4) yields an ϵ -graph-approximation $\mathbf{L}_{\tilde{\vec{G}}(\mathbf{a}, \mathbf{b})}$ of $\mathbf{L}_{\vec{G}(\mathbf{a}, \mathbf{b})}$ with $\text{nnz}(\mathbf{L}_{\tilde{\vec{G}}(\mathbf{a}, \mathbf{b})}) = O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))\epsilon^{-4})$ in time $O((\text{nnz}(\mathbf{a}) + \text{nnz}(\mathbf{b}))\epsilon^{-4})$. The in- and out-degrees of $\tilde{\vec{G}}(\mathbf{a}, \mathbf{b})$ and $\vec{G}(\mathbf{a}, \mathbf{b})$ are identical.

Algorithm 6: SPARSEPRODUCT($\mathbf{a}, \mathbf{b}, \epsilon$)

- 1 $d = \|\mathbf{a}\|_1 (= \|\mathbf{b}\|)$
 - 2 $\mathbf{A}_{\tilde{G}(\mathbf{a}, \mathbf{b})} = \text{SPARSEBIPARTITE}(\mathbf{a}, \mathbf{b}, \epsilon/128)$
 - 3 $\mathbf{A}_{\tilde{G}(\mathbf{a}, \mathbf{b})}^T = \mathbf{A}_{\tilde{G}(\mathbf{a}, \mathbf{b})}((n+1) : 2n, 1 : n)$
 - 4 **return** $\mathbf{A}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$
-

Proof. The runtime and sparsity is clear from Lemma 5.8. The preservation of in- and out-degrees follows from the preservation of degrees in the bipartite graph. It remains to show that

$$\left\| \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} \right\| \leq \epsilon.$$

By Lemma 3.14 we have

$$\left\| \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} \right\| = 2 \max_{\mathbf{x}, \mathbf{y} \neq \mathbf{0}} \frac{\mathbf{x}^T (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{y}}. \quad (14)$$

Since $\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$ and $\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}$ have identical in- and out-degrees, it is clear that $\mathbf{1}$ is in both the left and right kernel of $(\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})})$. We let $\mathbf{x}, \mathbf{y} \perp \mathbf{1}$ be maximizing the right hand side of (14). Further, we let $\mathbf{d} := \text{diag}(\mathbf{D}_{G(\mathbf{a}, \mathbf{b})})$ and

$$\begin{aligned} \hat{\mathbf{x}} &:= \begin{pmatrix} \mathbf{x} \\ \mathbf{x} \end{pmatrix} & \hat{\mathbf{y}} &:= \begin{pmatrix} \mathbf{y} \\ \mathbf{y} \end{pmatrix} & \hat{\mathbf{x}}' &:= \hat{\mathbf{x}} - \frac{\hat{\mathbf{x}}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}_{2n} & \hat{\mathbf{y}}' &:= \hat{\mathbf{y}} - \frac{\hat{\mathbf{y}}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}_{2n} \\ \mathbf{x}' &:= \mathbf{x} - \frac{\hat{\mathbf{x}}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}_n & \mathbf{y}' &:= \mathbf{y} - \frac{\hat{\mathbf{y}}^T \mathbf{d}}{\|\mathbf{d}\|_2} \mathbf{1}_n & \tilde{\mathbf{x}} &= \begin{pmatrix} \mathbf{0} \\ \mathbf{x}' \end{pmatrix} & \tilde{\mathbf{y}} &= \begin{pmatrix} \mathbf{y}' \\ \mathbf{0} \end{pmatrix}. \end{aligned}$$

Then we have

$$\begin{aligned} \left\| \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))}^{+/2} \right\| &= 2 \frac{\mathbf{x}^T (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{y}}{\mathbf{x}^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{y}} \\ &= 2 \frac{\mathbf{x}'^T (\mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \mathbf{y}'}{\mathbf{x}'^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{x} + \mathbf{y}'^T \mathbf{L}_{\mathcal{U}(\tilde{G}(\mathbf{a}, \mathbf{b}))} \mathbf{y}} \\ &= 4 \frac{\tilde{\mathbf{x}}^T (\mathbf{L}_{G(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \tilde{\mathbf{y}}}{\hat{\mathbf{x}}'^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \hat{\mathbf{x}}' + \hat{\mathbf{y}}'^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \hat{\mathbf{y}}'} \\ &\stackrel{i)}{\leq} 64 \frac{\tilde{\mathbf{x}}^T (\mathbf{L}_{G(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \tilde{\mathbf{y}}}{\hat{\mathbf{x}}'^T \text{diag}(\mathbf{d}) \hat{\mathbf{x}}' + \hat{\mathbf{y}}'^T \text{diag}(\mathbf{d}) \hat{\mathbf{y}}'} \\ &\stackrel{ii)}{\leq} 64 \frac{\tilde{\mathbf{x}}^T (\mathbf{L}_{G(\mathbf{a}, \mathbf{b})} - \mathbf{L}_{\tilde{G}(\mathbf{a}, \mathbf{b})}) \tilde{\mathbf{y}}}{\tilde{\mathbf{x}}^T \text{diag}(\mathbf{d}) \tilde{\mathbf{x}} + \tilde{\mathbf{y}}^T \text{diag}(\mathbf{d}) \tilde{\mathbf{y}}} \end{aligned}$$

where i) holds since $\mathbf{L}_{G(\mathbf{d})} \preceq 16 \mathbf{L}_{G(\mathbf{a}, \mathbf{b})}$ by Corollary 3.7 and $\mathbf{x}'^T \mathbf{L}_{G(\mathbf{d})} \mathbf{x}' = \mathbf{x}'^T \mathbf{D}_{G(\mathbf{d})} \mathbf{x}'$ by the choice of \mathbf{x}' . Analogously we have $\mathbf{y}'^T \text{diag}(\mathbf{d}) \mathbf{y}' \leq 16 \mathbf{y}'^T \mathbf{L}_{G(\mathbf{a}, \mathbf{b})} \mathbf{y}'$. Further, ii) holds since setting

parts of the vectors \mathbf{x}' and \mathbf{y}' to zero only decreases the value of the quadratic form with a positive diagonal matrix. Using $\mathbf{L}_{G(\mathbf{a},b)} \preceq 2\mathbf{D}_{G(\mathbf{a},b)} = 2\text{diag}(\mathbf{d})$ and Lemma 3.14 we conclude

$$\begin{aligned} 64 \frac{\tilde{\mathbf{x}}^T (\mathbf{L}_{G(\mathbf{a},b)} - \mathbf{L}_{\tilde{G}(\mathbf{a},b)}) \tilde{\mathbf{y}}}{\tilde{\mathbf{x}}^T \text{diag}(\mathbf{d}) \tilde{\mathbf{x}} + \tilde{\mathbf{y}}^T \text{diag}(\mathbf{d}) \tilde{\mathbf{y}}} &\leq 128 \frac{\tilde{\mathbf{x}}^T (\mathbf{L}_{G(\mathbf{a},b)} - \mathbf{L}_{\tilde{G}(\mathbf{a},b)}) \tilde{\mathbf{y}}}{\tilde{\mathbf{x}}^T \mathbf{L}_{G(\mathbf{a},b)} \tilde{\mathbf{x}} + \tilde{\mathbf{y}}^T \mathbf{L}_{G(\mathbf{a},b)} \tilde{\mathbf{y}}} \\ &= 128 \left\| \mathbf{L}_{G(\mathbf{a},b)}^{+/2} (\mathbf{L}_{G(\mathbf{a},b)} - \mathbf{L}_{\tilde{G}(\mathbf{a},b)}) \mathbf{L}_{G(\mathbf{a},b)}^{+/2} \right\|_2 \leq \epsilon \end{aligned}$$

where the last inequality follows from Lemma 5.8 since we call SPARSEBIPARTITE() with accuracy $\epsilon/128$. This concludes the proof. \square

Finally, we use Lemma 5.10 to prove Lemma 5.1.

Proof of Lemma 5.1. By Lemma 5.10 each individual call to SPARSEPRODUCT() conserves in- and out-degrees, and thus $\mathbf{L}_{\tilde{G}^2}$ and $\mathbf{L}_{\tilde{G}^2}$ are Eulerian Laplacians with the same in- and out-degrees. Further, by Lemma 5.10, we have that

$$\left\| \mathbf{L}_{\mathcal{U}(\tilde{G}_i)}^{+/2} (\mathbf{L}_{\tilde{G}_i} - \mathbf{L}_{\tilde{H}_i}) \mathbf{L}_{\mathcal{U}(\tilde{G}_i)}^{+/2} \right\|_2 \leq \epsilon$$

By Lemma 3.14 we conclude that for all $\mathbf{x}, \mathbf{y} \neq \mathbf{0}$

$$\mathbf{x}^T (\mathbf{L}_{\tilde{G}_i} - \mathbf{L}_{\tilde{H}_i}) \mathbf{y} \leq \frac{\epsilon}{2} (\mathbf{x}^T \mathbf{L}_{\mathcal{U}(\tilde{G}_i)} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{\mathcal{U}(\tilde{G}_i)} \mathbf{y}).$$

Finally, using Lemma 3.14 again, we have for some vectors \mathbf{x}, \mathbf{y}

$$\begin{aligned} \left\| \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}}^{+/2} (\mathbf{L}_{\tilde{G}^2} - \mathbf{L}_{\tilde{G}^2}) \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}}^{+/2} \right\|_2 &= \left\| \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}}^{+/2} \left(\sum_{i=1}^n \mathbf{L}_{\tilde{G}_i} - \mathbf{L}_{\tilde{H}_i} \right) \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}}^{+/2} \right\|_2 \\ &= 2 \frac{\mathbf{x}^T (\sum_{i=1}^n \mathbf{L}_{\tilde{G}_i} - \mathbf{L}_{\tilde{H}_i}) \mathbf{y}}{\mathbf{x}^T \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}} \mathbf{x} + \mathbf{y}^T \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}} \mathbf{y}} \\ &\leq \epsilon \frac{\sum_{i=1}^n (\mathbf{x}^T \mathbf{L}_{\mathcal{U}(\tilde{G}_i)} \mathbf{x} + \mathbf{y}^T \mathbf{L}_{\mathcal{U}(\tilde{G}_i)} \mathbf{y})}{\mathbf{x}^T \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}} \mathbf{x} + \mathbf{y}^T \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}} \mathbf{y}} = \epsilon \end{aligned}$$

where we used that $\sum_{i=1}^n \mathbf{L}_{\mathcal{U}(\tilde{G}_i)} = \mathbf{U}_{\mathbf{L}_{\tilde{G}^2}}$. Clearly, this also shows that

The runtime and sparsity directly follow from Lemma 5.10. \square

6 Deterministic Square Sparsifier Chains

In this section we define the collection of $\tilde{O}(1)$ sparse matrices our algorithm operates on. This collection will be computed once, not recursively, and relies on square sparsifier chains.

Definition 6.1 (Square Sparsifier Chain, Definition 4.6 in [CKP⁺16b]). *We call sequences of matrices $\mathcal{S} = \mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d \in \mathbb{R}^{n \times n}$ a square sparsifier chain of length d with parameter $0 < \alpha \leq \frac{1}{2}$ and error $\epsilon \leq \frac{1}{2}$ (or a (d, ϵ, α) -square chain for short) if under the definitions $\mathbf{L}_i = \mathbf{I} - \mathcal{A}_i$ and $\mathcal{A}^{(\alpha)} = \alpha \mathbf{I} + (1 - \alpha) \mathcal{A}_i$ the following hold*

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

Since we cannot afford to run our global sparsification routine after each squaring operation while still achieving an almost-linear runtime, we compute chains of depth $d = O((\log n)^\delta)$ for $\delta = 1/3$ and then globally sparsify. These global sparsifications will naturally correspond to branching points of our recursive algorithm, since we have to rectify the loss due to global sparsification. We first define pseudoinverse chains.

Definition 6.2 (Pseudoinverse Chain). *We call a collection $\mathcal{C} = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=0}^k$ with $\mathcal{S}^{(i)} = \{\mathcal{A}_0^{(i)}, \mathcal{A}_1^{(i)}, \dots, \mathcal{A}_d^{(i)}\}$ a $(k, d, \alpha, \epsilon, \gamma)$ -pseudoinverse chain if*

1. $\mathcal{S}^{(i)}$ is a (d, ϵ, α) -square chain for all i .
2. $\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}$ is a γ -quadruple for all i .
3. $\frac{1}{1+\frac{\epsilon}{\eta}} \mathbf{D}_{\underline{G}}^{+/2} \mathbf{A}_{\underline{G}_3^{(i)}}^T \mathbf{D}_{\underline{G}}^{+/2} = \mathcal{A}_0^{(i)}$ for all i .
4. $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{A}_{\underline{G}_0^{(i+1)}}^T \mathbf{D}_{\underline{G}}^{+/2} = \mathcal{A}_d^{(i)}$ for all $i < k$.

where $\underline{G} = \underline{G}_0^{(0)}$ and $\frac{\eta}{\beta}$ is as in Algorithm 2 and $\mathbf{D}_{\underline{G}_0^{(i)}} = \mathbf{D}_{\underline{G}}$ for all i .

We then state some previous work that will be useful for showing that Algorithm 7 creates a pseudoinverse chain. The following lemma shows that ϵ -approximations behave well under diagonal rescalings.

Lemma 6.3 (Lemma 3.7 in [CKP⁺16b]). *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 $\ker(\mathbf{M}^T) \subseteq \ker(\mathbf{U}_A)$, then $\mathbf{M}^T \tilde{\mathbf{A}} \mathbf{M}$ is an ϵ -approximation of $\mathbf{M}^T \mathbf{A} \mathbf{M}$.*

And the next lemma will be used to show the first point in Definition 6.1.

Lemma 6.4 (Lemma 4.7 in [CKP⁺16b]). *If for $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{D} = \text{diag}(\mathbf{A}\mathbf{1})$ the matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}^T$ is an Eulerian Laplacian associated with a strongly connected graph then, $\left\| \mathbf{D}^{-1/2} \mathbf{A}^T \mathbf{D}^{-1/2} \right\|_2 \leq 1$ and $\ker(\mathbf{L}) = \ker(\mathbf{L}^T) = \ker(\mathbf{U}_L)$*

Given these results, we first show a lemma about the subroutine SQUARECHAIN() in Algorithm 7.

Lemma 6.5. *Given a strongly connected Eulerian graph $\frac{1}{1+\frac{\epsilon}{\eta}} \underline{G}_3^{(i)}$ with $m = \tilde{O}(n)$ edges and degree matrix $\mathbf{D}_{\underline{G}}$, as well as parameters $\alpha \in (0, 1)$ and $\delta \in (0, 1/2)$ the subroutine $\underline{G}_0^{(i+1)}, \mathcal{S}^{(i)} = \text{SQUARECHAIN}\left(\frac{1}{1+\frac{\epsilon}{\eta}} \underline{G}_3^{(i)}, d = \Theta((\log n)^\delta), \alpha, \epsilon = \frac{1}{30 \cdot \text{Exp}(5d)}\right)$ computes a (d, ϵ, α) -square chain $\mathcal{S}^{(i)}$ and a graph $\underline{G}_0^{(i+1)}$ with degree matrix $\mathbf{D}_{\underline{G}}$ so that*

Algorithm 7: CHAINCONSTRUCTION($\underline{G}, k, d, \alpha, \epsilon, \gamma$) and SQUARECHAIN($\hat{\underline{G}}^{(i)}, d, \alpha, \epsilon$)

```

1 Algorithm CHAINCONSTRUCTION( $\underline{G}_0^{(0)} = \underline{G}, k, d, \alpha, \epsilon, \gamma$ )
2   for  $i = 0, \dots, k$  do
3      $\underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)} = \text{GLOBALSPARSIFICATION}(\underline{G}_0^{(i)}, \gamma)$  ;           // See Algorithm 2.
4     // Scale back degrees by  $\frac{\eta}{\beta}$  before building the next chain
5      $\underline{G}_0^{(i+1)}, \mathcal{S}^{(i)} = \text{SQUARECHAIN}(\frac{1}{1+\frac{\beta}{\eta}} \underline{G}_3^{(i)}, d, \alpha, \epsilon)$ 
6   end
7   return  $\mathcal{C} = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=0}^k$ 
8 Procedure SQUARECHAIN( $\underline{G}^{(i,0)} = \frac{1}{1+\frac{\beta}{\eta}} \underline{G}_3^{(i)}, d, \alpha, \epsilon$ )
9   for  $j = 1, \dots, d$  do
10    Let  $G_D$  denote the graph with adjacency matrix  $\mathbf{D}_{\underline{G}}$ , only consisting of self-loops.
11     $\underline{G}^{(i,j)} = \text{SPARSE SQUARE}((1 - \alpha)\underline{G}^{(i,j-1)} + \alpha G_D, \epsilon)$  ;           // See Algorithm 4.
12  end
13  // We use  $\mathbf{D}_{\underline{G}} = \mathbf{D}_{\underline{G}^{(i,0)}} = \mathbf{D}_{\underline{G}^{(i,j)}}$  here since we preserve degrees.
14  return  $\underline{G}_0^{(i+1)} = \underline{G}^{(i,d)}, \mathcal{S}^{(i)} = \{\mathcal{A}_j^{(i)} = \mathbf{D}_{\underline{G}}^{+/2} \mathbf{A}_{\underline{G}^{(i,j)}}^T \mathbf{D}_{\underline{G}}^{+/2}\}_{j=0}^d$ 

```

1. $\text{nnz}(\mathcal{A}_j^{(i)}) = n^{1+o(1)}$ for $j = 0, \dots, d$.

2. $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{A}_{\underline{G}^{(i,0)}}^T \mathbf{D}_{\underline{G}}^{+/2} = \mathcal{A}_0^{(i)}$ and $\frac{1}{1+\frac{\beta}{\eta}} \mathbf{D}_{\underline{G}}^{+/2} \mathbf{A}_{\underline{G}_3^{(i)}}^T \mathbf{D}_{\underline{G}}^{+/2} = \mathcal{A}_d^{(i)}$

Proof. Notice that $\mathbf{D}_{\underline{G}} = \mathbf{D}_{\underline{G}^{(i,0)}} = \mathbf{D}_{\underline{G}^{(i,d)}}$ since degrees are exactly preserved by SPARSE SQUARE(). We first show that $\mathcal{S}^{(i)}$ is a (d, ϵ, α) -square chain. Since the graph $(1 - \alpha)\underline{G}^{(i,j-1)} + \alpha G_D$ has the adjacency matrix $(1 - \alpha)\mathbf{A}_{\underline{G}^{(i,j-1)}} + \alpha \mathbf{D}_{\underline{G}^{(i,j-1)}}$, its square contains a multiple of $\underline{G}^{(i,j-1)}$ as a subgraph and is therefore strongly connected, if $\underline{G}^{(i,j-1)}$ is strongly connected. Then by Lemma 5.1 $\underline{G}^{(i,j)}$ is strongly connected. Together with Lemma 6.3 and Lemma 6.4 this yields the properties of a (d, ϵ, α) -square chain by induction. It remains to show sparsity.

Initially, we have that $\text{nnz}(\mathcal{A}_0^{(i)}) = \tilde{O}(n)$. By Lemma 5.1 we have

$$\text{nnz}(\mathcal{A}_{j+1}^{(i)}) \leq C \text{nnz}(\mathcal{A}_j^{(i)}) / \epsilon^4 = 30C \cdot \text{Exp}(20(\log n)^\delta) \cdot \text{nnz}(\mathcal{A}_j^{(i)})$$

for some constant C . By induction, we conclude

$$\text{nnz}(\mathcal{A}_j^{(i)}) \leq (30C)^j \cdot \text{Exp}(20j(\log n)^\delta) \cdot \text{nnz}(\mathcal{A}_0^{(i)}) \leq \text{Exp}(O((\log n)^{2\delta})) \cdot \tilde{O}(n) = n^{1+o(1)}$$

where we used that $\delta < 1/2$ is a constant. \square

Next we show that CHAINCONSTRUCTION() in Algorithm 7 deterministically computes a pseudoinverse chain in almost linear time, using the machinery we developed.

Lemma 6.6. For constants $\delta, \gamma \in (0, 1/2)$, $\gamma < \delta$, $k \geq 1$, $\alpha = 1/4$ and a strongly connected Eulerian graph \underline{G} with m edges the algorithm $\mathcal{C} = \text{CHAINCONSTRUCTION}(\underline{G}, k, d = O((\log n)^\delta), \alpha, \epsilon = 1/(30 \cdot \text{Exp}(5d)), \gamma)$ computes a $(k, d, \alpha, \epsilon, \gamma)$ -pseudoinverse chain \mathcal{C} in time $m^{1+o(1)}$ so that

1. $|E(\underline{G}_j^{(0)})| \leq 2m + \tilde{O}(n)$ for $j = 1, 2$, $|E(\underline{G}_3^{(0)})| = \tilde{O}(n)$.
2. $|E(\underline{G}_j^{(i)})| = n^{1+o(1)}$ for $i > 0$, $j = 0, 1, 2, 3$.
3. $\text{nnz}(\mathcal{A}_j^{(i)}) = n^{1+o(1)}$ for all i, j .

Proof. Follows directly from Lemma 6.5 and Lemma 4.4. \square

Before concluding with a lemma about the improvement in condition number of a pseudoinverse chain, we show a convenient corollary about the quadruplets introduced in Section 4 and state a lemma about the improvement in condition number due to squaring.

Corollary 6.7. Given a $(\gamma, \beta = \text{Exp}(O((\log n)^\gamma)), \eta = \text{Exp}(-3(\log n)^\gamma))$ -quadruple $\underline{G}_0, \underline{G}_1, \underline{G}_2, \underline{G}_3$ for $\gamma \in (0, 1)$ constant, we have

$$\text{Exp}(-O((\log n)^\gamma)) \lambda_*(\underline{D}_{\underline{G}_0}^{+/2} \underline{L}_{\underline{G}_0} \underline{D}_{\underline{G}_0}^{+/2}) \leq \frac{\eta}{\beta} \lambda_*(\underline{D}_{\underline{G}_0}^{+/2} \underline{L}_{\underline{G}_3} \underline{D}_{\underline{G}_0}^{+/2})$$

Proof. By the second property of Definition 4.3 we have

$$\text{Exp}(-O((\log n)^\gamma)) \underline{D}_{\underline{G}_0}^{+/2} \underline{L}_{\underline{G}_0} \underline{D}_{\underline{G}_0}^{+/2} \preceq \underline{D}_{\underline{G}_0}^{+/2} \underline{L}_{\underline{G}_3} \underline{D}_{\underline{G}_0}^{+/2}.$$

Since $\frac{\eta}{\beta} \geq \text{Exp}(-O((\log n)^\gamma))$ we conclude the lemma. \square

Lemma 6.8 (Lemma 4.9 in [CKP⁺16b]). For length $d \geq 1$, parameter $\alpha = 1/4$ and error $\epsilon \in (1, 1/2)$ and a (d, α, ϵ) -chain $\mathcal{A}_0, \mathcal{A}_1, \dots, \mathcal{A}_d$ the following hold:

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

Finally, we show that a deep enough pseudoinverse chain reaches a constant condition number.

Lemma 6.9. Given $\gamma, \delta, \epsilon, \alpha, \mathcal{C}$ and \underline{G} as in Lemma 6.6 where we set $k = \Theta(\log(1/\hat{\lambda}_*)/(\log n)^\delta)$ for $1/\text{poly}(n) \leq \hat{\lambda}_* \leq \lambda_*(\underline{D}_{\underline{G}}^{+/2} \underline{U}_{\underline{L}_{\underline{G}}} \underline{D}_{\underline{G}}^{+/2})$ we have $\lambda_*(\underline{D}_{\underline{G}}^{+/2} \underline{U}_{\underline{L}_{\underline{G}}^{(k)}} \underline{D}_{\underline{G}}^{+/2}) \geq 1/4$

Proof. By Corollary 6.7 we have for every $i = 0, 1, \dots, k$

$$\text{Exp}(-O((\log n)^\gamma)) \cdot \lambda_*(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_0^{(i)}} \underline{D}_{\underline{G}}^{+/2}) \leq \frac{\eta}{\beta} \cdot \lambda_*(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_3^{(i)}} \underline{D}_{\underline{G}}^{+/2})$$

and by Lemma 6.8 we have

$$\min\{1.125^d \lambda_*(\frac{\eta}{\beta} \underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_3^{(i)}} \underline{D}_{\underline{G}}^{+/2}), 1/4\} \leq \lambda_*(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_0^{(i+1)}} \underline{D}_{\underline{G}}^{+/2}).$$

Therefore each of the k chains improves the minimum eigenvalue by $\frac{1.125^d}{\text{Exp}(O((\log n)^\gamma))} = \text{Exp}(\Omega((\log n)^\delta))$ for $\delta > \lambda$ and $d = \Theta((\log n)^\delta)$ big enough. After $k = \Theta(\log(\hat{\lambda}_*)/d)$ iterations this ensures the threshold $1/4$ is reached. \square

7 A Deterministic Squaring Solver

In this section, we build our recursive squaring solver on top of the pseudoinverse chain constructed in the previous section. It is inspired by and closely follows the solver presented in [CKP⁺16b]. In our algorithm, the global sparsifications of the pseudoinverse chain correspond to branching points of the recursion. This is necessary since we incur relatively high sub-polynomial losses when globally sparsifying. We state the main theorem of this section.

Theorem 7.1 (Deterministic Eulerian Solver). *Given the Laplacian $\mathbf{L}_{\underline{G}}$ of a strongly connected Eulerian graph \underline{G} with m edges and polynomially bounded edge weights, a parameter $\epsilon \in (0, 1)$ and a lower bound $\hat{\lambda}_*$ on $\lambda_*(\mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}} \mathbf{D}_{\underline{G}}^{+/2})$ so that $\hat{\lambda}_* \geq 1/\text{poly}(n)$ the algorithm $\text{SOLVEEULERIAN}(\mathbf{L}_{\underline{G}}, \epsilon, \hat{\lambda}_*)$ (Algorithm 9) deterministically computes a vector \mathbf{x} so that*

$$\left\| \mathbf{x} - \mathbf{L}_{\underline{G}}^+ \mathbf{b} \right\|_{U_{\underline{G}}} \leq \epsilon \left\| \mathbf{L}_{\underline{G}}^+ \mathbf{b} \right\|_{U_{\underline{G}}}$$

in time $m^{1+o(1)} \cdot \log \frac{1}{\epsilon}$ for a vector $\mathbf{b} \in \text{im}(\mathbf{L}_{\underline{G}})$.

The following lemma re-interprets preconditioned Richardson in terms of augmenting the quality of a pseudoinverse.

Lemma 7.2 (Pseudoinverse Improvement, Lemma 4.4 in [CKP⁺16b]). *If \mathbf{Z} is an ϵ -approximate-pseudoinverse of \mathbf{M} with respect to \mathbf{U} , for $\epsilon \in (0, 1)$, $\mathbf{b} \in \text{im}(\mathbf{M})$ and $N \geq 0$, then $\mathbf{x}_N = \text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \mathbf{b}, 1, N)$ computes $\mathbf{x}_N = \mathbf{Z}_N \mathbf{b}$ 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} .*

Our algorithm frequently uses preconditioned Richardson to boost the quality of pseudoinverses. Following [CKP⁺16b] we let $\mathbf{Z}_N = \text{PRECONRICHARDSON}(\mathbf{M}, \mathbf{Z}, \cdot, \eta, N)$ denote the implicit matrix \mathbf{Z}_N as in Lemma 7.2. To apply it to a vector \mathbf{b} , we need to apply the (possibly implicit) matrices \mathbf{M} and \mathbf{Z} a total of N times. Before we show the main technical lemma of this section, we reference previous work that we use in the proof. We first state a lemma about solving well conditioned linear equations with Richardson which we will use to establish the base case of our recursion

Lemma 7.3 (Lemma 4.5 in [CKP⁺16b]). *Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ such that $\mathbf{U}_{\mathbf{M}}$ is PSD with $\ker(\mathbf{M}) = \ker(\mathbf{M}^T)$. Then for parameters $\eta \leq \lambda_*(\mathbf{U}_{\mathbf{M}})/\|\mathbf{M}\|_2^2$ and $N > 0$ we have that the resulting matrix $\mathbf{Z}_N = \text{PRECONRICHARDSON}(\mathbf{M}, \eta \mathbf{I}_{\text{im}(\mathbf{M})}, \cdot, 1, N)$ is an $\text{Exp}(-N\eta\lambda_*(\mathbf{U}_{\mathbf{M}})/2)$ -approximate pseudoinverse.*

Then we state a lemma that formalises obtaining preconditioners via squaring as introduced in the overview.

Lemma 7.4 (Lemma 4.13 in [CKP⁺16b]). *Let the sequence $\mathcal{A}_0, \dots, \mathcal{A}_d$ be an (d, ϵ, α) -chain as specified in Definition 6.1, with $\epsilon \leq 1/2$ and $\alpha \leq 1/4$. Using the notation from Definition 6.1, consider the matrix*

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

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

Finally, we need two lemmas about composing approximate pseudoinverses and changing norms.

Lemma 7.5 (Lemma 4.10 in [CKP⁺16b]). *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} .*

Lemma 7.6 (Lemma 4.12 in [CKP⁺16b]). *Let $\mathbf{Z}, \mathbf{M}, \mathbf{U} \in \mathbb{R}^{n \times n}$ be matrices such that \mathbf{U} is PSD, and $\ker(\mathbf{Z}) = \ker(\mathbf{Z}^T) = \ker(\mathbf{M}) = \ker(\mathbf{M}^T) \supseteq \ker(\mathbf{U})$. Then the following holds:*

1. (Preserved under right multiplication) *Let $\mathbf{C} \in \mathbb{R}^{n \times n}$ such that both \mathbf{C} and \mathbf{C}^T are invariant on the kernel of \mathbf{M} in the sense that $\mathbf{x} \in \ker(\mathbf{M})$ if and only if $\mathbf{C}\mathbf{x} \in \ker(\mathbf{M})$, and similarly for \mathbf{C}^\perp . 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 PSD matrix $\widetilde{\mathbf{U}}$, such that $\ker(\widetilde{\mathbf{U}}) = \ker(\mathbf{U})$, \mathbf{Z} is an $(\epsilon\sqrt{\kappa(\widetilde{\mathbf{U}}, \mathbf{U})})$ -approximate pseudoinverse of \mathbf{M} with respect to $\widetilde{\mathbf{U}}$.*

These results, together with the properties of pseudoinverse chains, allow us to show the following main technical lemma of this section, which establishes correctness of the algorithm SOLVERECURSIVE() (Algorithm 8).

Lemma 7.7. *Given the tail $\mathcal{C} = \{\underline{\mathcal{G}}_0^{(i)}, \underline{\mathcal{G}}_1^{(i)}, \underline{\mathcal{G}}_2^{(i)}, \underline{\mathcal{G}}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l}^k$ of a $(k, d = \Theta((\log n)^{1/3}), \alpha = 1/4, \epsilon_0 = \frac{1}{30\text{Exp}(5d)}, \gamma = 1/10)$ -pseudoinverse chain as in Lemma 6.9, and a parameter $\epsilon \in (0, 1/2)$ the algorithm SOLVERECURSIVE(\mathcal{C}, ϵ) computes an ϵ -approximate pseudoinverse $\mathbf{Z}_0^{(l)}$ of $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{L}_{\underline{\mathcal{G}}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ with respect to $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{U}_{\mathbf{L}_{\underline{\mathcal{G}}}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ where $\mathbf{D}_{\underline{\mathcal{G}}} = \mathbf{D}_{\underline{\mathcal{G}}_0^{(l)}}$.*

Proof. We use that $\mathbf{D}_{\underline{\mathcal{G}}} = \mathbf{D}_{\underline{\mathcal{G}}_0^{(i)}}$ for all i by Definition 6.2. The proof is by induction on $k - l$.

- *Base case:* $k - l = 0$. We land in the if case of the algorithm SOLVERECURSIVE() and denote $\mathbf{I} - \mathcal{A} = \mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{L}_{\underline{\mathcal{G}}_0^{(k)}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ as in the algorithm. Then we have $\lambda_*(\mathbf{I} - \mathbf{U}\mathcal{A}) \geq 1/4$ by Lemma 6.9, which is exactly what we set out to achieve with squaring. By Lemma 6.4 we further have $\|\mathbf{I} - \mathcal{A}\|_2^2 \leq 4$. Then the base case follows directly from Lemma 7.3.
- *Step case:* $k - l > 0$. We land in the else case of the algorithm SOLVERECURSIVE() and let $\mathcal{C}' = \{\underline{\mathcal{G}}_0^{(i)}, \underline{\mathcal{G}}_1^{(i)}, \underline{\mathcal{G}}_2^{(i)}, \underline{\mathcal{G}}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l+1}^k$. Then the induction hypothesis is that SOLVERECURSIVE(\mathcal{C}', ϵ) returns an ϵ -approximate pseudoinverse $\mathbf{Z}_0^{(l+1)}$ of $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{L}_{\underline{\mathcal{G}}_0^{(l+1)}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ with respect to $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{U}_{\mathbf{L}_{\underline{\mathcal{G}}_0^{(l+1)}}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$. To establish the step case we rely on a second, nested induction of constant depth, showing the correctness of the subroutine PEEL(). Namely, we show that $\mathbf{Z}_p^{(l)} = \text{PEEL}(\{\underline{\mathcal{G}}_j^{(l)}\}_{j=p}^3, \mathcal{S}^{(l)}, \mathcal{C}', \epsilon)$ returns an ϵ -approximate pseudoinverse $\mathbf{Z}_p^{(l)}$ of $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{L}_{\underline{\mathcal{G}}_p^{(l)}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ with respect to $\mathbf{D}_{\underline{\mathcal{G}}}^{+/2} \mathbf{U}_{\mathbf{L}_{\underline{\mathcal{G}}_p^{(l)}}} \mathbf{D}_{\underline{\mathcal{G}}}^{+/2}$ for $p = 0, 1, 2, 3$. The nested induction is on $3 - p$.

Algorithm 8: SOLVERECURSIVE($\mathcal{C} = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l}^k, \epsilon$) and PEEL()

```

// We let  $\underline{G}_0^{(l)} = \underline{G}$  in these algorithms.
1 Algorithm SOLVERECURSIVE( $\mathcal{C} = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l}^k, \epsilon$ )
2   if  $l = k$  then
3     // Leaf of Recursion
4      $\mathcal{A} = \underline{D}_{\underline{G}}^{+/2} \underline{A}_{\underline{G}_0^{(k)}}^T \underline{D}_{\underline{G}}^{+/2}$ 
5      $\underline{Z}_0^{(k)} = \text{PRECONRICHARDSON}(\underline{I} - \mathcal{A}, \frac{1}{16} \underline{I}_{\text{im}(\underline{I} - \mathcal{A})}, \cdot, 1, 256 \log(1/\epsilon))$ 
6   else
7      $\underline{Z}_0^{(l)} = \text{PEEL}(\{\underline{G}_j^{(l)}\}_{j=0}^3, \mathcal{S}^{(l)}, \mathcal{C}' = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l+1}^k, \epsilon)$ 
8   end
9   return  $\underline{Z}_0^{(l)}$ 
10 Procedure PEEL( $\{\underline{G}_j^{(l)}\}_{j=p}^3, \mathcal{S}^{(l)}, \mathcal{C}' = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l+1}^k, \epsilon$ )
11   if  $p = 3$  then
12     // We have  $\mathcal{S}^{(l)} = \mathcal{A}_0^{(l)}, \dots, \mathcal{A}_d^{(l)}$  and we let  $\mathcal{A}_t^{(l,1/4)} = \frac{3}{4} \mathcal{A}_t^{(l)} + \frac{1}{4} \underline{I}$ 
13      $\underline{Z}_0^{(l+1)} = \text{SOLVERECURSIVE}(\mathcal{C}', \frac{1}{30 \cdot \text{Exp}(5d)})$ 
14      $\underline{Z} = (3/4)^{d+1} \underline{Z}_0^{(l+1)} (\underline{I} + \mathcal{A}_{d-1}^{(l,1/4)}) \dots (\underline{I} + \mathcal{A}_0^{(l,1/4)})$ 
15      $\underline{Z}_3^{(l)} = \text{PRECONRICHARDSON}(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_p^{(l)}} \underline{D}_{\underline{G}}^{+/2}, (1 + \frac{\beta}{\eta}) \underline{Z}, \cdot, 1, \log(1/\epsilon))$ 
16   else
17      $\underline{Z}_{p+1}^{(l)} = \text{PEEL}(\{\underline{G}_j^{(l)}\}_{j=p+1}^3, \mathcal{S}^{(l)}, \mathcal{C}', \frac{1}{\text{Exp}(\Theta((\log n)^\gamma))})$ 
18      $\underline{Z}_p^{(l)} = \text{PRECONRICHARDSON}(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}_p^{(l)}} \underline{D}_{\underline{G}}^{+/2}, \underline{Z}_{p+1}^{(l)}, \cdot, 1, \text{Exp}(\Theta((\log n)^\gamma)) \cdot \log(1/\epsilon))$ 
19   end
20   return  $\underline{Z}_p^{(l)}$ 

```

- *Nested base case:* $3-p=0$. We land in the if case of the procedure $\text{PEEL}()$. Noting that $\mathbf{I} - \mathcal{A}_d^{(l)} = \mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}_0^{(l+1)}} \mathbf{D}_{\underline{G}}^{+/2}$ by Definition 6.2, we have that $\mathbf{Z}_0^{(l+1)}$ is an $\frac{1}{30 \cdot \text{Exp}(5d)}$ -approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_d^{(l)}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_d^{(l)}}$ by the induction hypothesis of the main induction. Then, by Lemma 7.4 \mathbf{Z} is an $1/30$ approximate pseudoinverse of $\mathbf{I} - \mathcal{A}_0^{(l)}$ with respect to $\mathbf{I} - \mathbf{U}_{\mathcal{A}_0^{(l)}}$. Recall that $\mathbf{I} - \mathcal{A}_0^{(l)} = (1 + \frac{\beta}{\eta})^{-1} \mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}_3^{(l)}} \mathbf{D}_{\underline{G}}^{+/2}$. By Lemma 7.2 the matrix $\mathbf{Z}_3^{(l)} = \text{PEEL}(\{\underline{G}_3^{(l)}\}_{j=p}^3, \mathcal{S}^{(l)}, \mathcal{C}', \epsilon)$ is an ϵ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}_3^{(l)}} \mathbf{D}_{\underline{G}}^{+/2}$ with respect to $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{U}_{\underline{L}_{\underline{G}_3^{(l)}}} \mathbf{D}_{\underline{G}}^{+/2}$ since the factor $(1 + \frac{\beta}{\eta})$ cancels. This establishes the base case.
- *Nested step case:* $p < 3$. We land in the else case of the procedure $\text{PEEL}()$. By the induction hypothesis $\mathbf{Z}_{p+1}^{(l)}$ is a $(\epsilon_1 = \frac{1}{\text{Exp}(\Theta((\log n)^\gamma))})$ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}_{p+1}^{(l)}} \mathbf{D}_{\underline{G}}^{+/2}$ with respect to $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{U}_{\underline{L}_{\underline{G}_{p+1}^{(l)}}} \mathbf{D}_{\underline{G}}^{+/2}$. Further, we have that $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{L}_{\underline{G}_{p+1}^{(l)}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ is an $(\epsilon_2 = 1 - \frac{1}{\text{Exp}(O((\log n)^\gamma))})$ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{L}_{\underline{G}_p^{(l)}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ with respect to $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{U}_{\underline{L}_{\underline{G}_{p+1}^{(l)}}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ by Definition 4.3. Combining the two with Lemma 7.5 yields that $\mathbf{Z}_{p+1}^{(l)}$ is an $(\epsilon_1 + \epsilon_2 + \epsilon_1 \epsilon_2 = 1 - \frac{1}{\text{Exp}(O((\log n)^\gamma))})$ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{L}_{\underline{G}_p^{(l)}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ with respect to $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{U}_{\underline{L}_{\underline{G}_{p+1}^{(l)}}}^+ \mathbf{D}_{\underline{G}}^{1/2}$, where we choose ϵ_1 small enough. By Lemma 7.2, $\text{Exp}(\Theta((\log n)^\gamma) \log(1/\epsilon))$ iterations of preconditioned Richardson suffice to reduce the error to ϵ . However, the resulting approximate pseudoinverse is with respect to $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{U}_{\underline{L}_{\underline{G}_{p+1}^{(l)}}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ instead of $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{U}_{\underline{L}_{\underline{G}_p^{(l)}}}^+ \mathbf{D}_{\underline{G}}^{1/2}$. By the second point in Definition 4.3 and Lemma 7.6 another factor of $\tilde{O}(1)$ in the iteration count suffices to translate between norms. This factor can be subsumed in the iteration count $\text{Exp}(\Theta((\log n)^\gamma) \log(1/\epsilon))$. Therefore, $\mathbf{Z}_p^{(l)}$ is an ϵ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{L}_{\underline{G}_p^{(l)}}^+ \mathbf{D}_{\underline{G}}^{1/2}$ with respect to $\mathbf{D}_{\underline{G}}^{1/2} \mathbf{U}_{\underline{L}_{\underline{G}_p^{(l)}}}^+ \mathbf{D}_{\underline{G}}^{1/2}$, which concludes the step case.

The nested induction establishes that $\mathbf{Z}_0^{(l)}$ is an ϵ -approximate pseudoinverse of $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{L}_{\underline{G}_0^{(l)}} \mathbf{D}_{\underline{G}}^{+/2}$ with respect to $\mathbf{D}_{\underline{G}}^{+/2} \mathbf{U}_{\underline{L}_{\underline{G}_0^{(l)}}} \mathbf{D}_{\underline{G}}^{+/2}$. This concludes the step case, since $\text{SOLVERECURSIVE}(\mathcal{C} = \{\underline{G}_0^{(i)}, \underline{G}_1^{(i)}, \underline{G}_2^{(i)}, \underline{G}_3^{(i)}, \mathcal{S}^{(i)}\}_{i=l}^k, \epsilon)$ returns $\mathbf{Z}_0^{(l)}$. The lemma follows by induction. \square

Before we conclude with the main theorem of this section, we analyse the runtime of $\text{SOLVEEULERIAN}(\mathbf{L}_{\underline{G}}, \mathbf{b}, \epsilon, \hat{\lambda}_*)$.

Lemma 7.8. *Given $\hat{\lambda}_* \geq 1/\text{poly}(n)$, the Laplacian of an Eulerian graph \underline{G} with n -vertices and m -edges, a vector $\mathbf{b} \in \mathbb{R}^n$ so that $\mathbf{b} \in \text{im}(\mathbf{L}_{\underline{G}})$ and a parameter $\epsilon \in (0, 1)$, $\text{SOLVEEULERIAN}(\mathbf{L}_{\underline{G}}, \mathbf{b}, \epsilon, \hat{\lambda}_*)$ runs in time $m^{1+o(1)} \log(1/\epsilon)$.*

Algorithm 9: SOLVEEULERIAN($\underline{L}_{\underline{G}}, \mathbf{b}, \epsilon, \hat{\lambda}_*$)

```

1  $d = \Theta((\log n)^{1/3}); k = C \log(1/\hat{\lambda}_*)/d \in O((\log n)^{2/3}); \epsilon_0 = \frac{1}{30 \cdot \text{Exp}(5d)}$ 
2  $\mathcal{C} = \text{CHAINCONSTRUCTION}(\underline{G}, k, d, \delta = 1/3, \epsilon_0, \gamma = 1/10)$ 
3  $\mathbf{Z} = \text{SOLVERECURSIVE}(\mathcal{C}, 1/10)$ 
4  $\hat{\mathbf{Z}} = \text{PRECONRICHARDSON}(\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}} \underline{D}_{\underline{G}}^{+/2}, \mathbf{Z}, \cdot, 1, \log(1/\epsilon))$ 
5 return  $\underline{D}_{\underline{G}}^{+/2} \hat{\mathbf{Z}} \underline{D}_{\underline{G}}^{+/2} \mathbf{b}$ 

```

Proof. The pseudoinverse chain is built in $m^{1+o(1)}$ time by Lemma 6.6. By Definition 6.2 all matrices involved have at most $m^{1+o(1)}$ edges, and Richardson only ever multiplies with these matrices. Therefore it suffices to show that the total branching of our recursion SOLVERECURSIVE() is $n^{o(1)}$. There are 4 branching points for each level $l \in [k]$ for $k = \Theta((\log n)^{2/3})$ in the procedure PEEL() in Algorithm 8. Each of these branches by a factor at most $f = \text{Exp}(\Theta((\log n)^{1/10}))$. Therefore the total branching is at most $f^k \leq O(1) \cdot \text{Exp}((\log n)^{4/5}) = n^{o(1)}$. Finally SOLVEEULERIAN() contributes another factor of $O(\log(1/\epsilon))$. This concludes the proof. \square

Finally, we put the pieces together and show Theorem 7.1.

Proof of Theorem 7.1. The runtime follows from Lemma 7.8. \mathcal{C} is a pseudoinverse chain by Lemma 6.6. Then, by Lemma 7.7 we have that \mathbf{Z} is a $1/10$ approximate pseudoinverse of $\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}} \underline{D}_{\underline{G}}^{+/2}$ with respect to $\underline{D}_{\underline{G}}^{+/2} \underline{U}_{\underline{L}_{\underline{G}}} \underline{D}_{\underline{G}}^{+/2}$. Therefore $\hat{\mathbf{Z}}$ in SOLVEEULERIAN() is an ϵ -approximate pseudoinverse of $\underline{D}_{\underline{G}}^{+/2} \underline{L}_{\underline{G}} \underline{D}_{\underline{G}}^{+/2}$ with respect to $\underline{D}_{\underline{G}}^{+/2} \underline{U}_{\underline{L}_{\underline{G}}} \underline{D}_{\underline{G}}^{+/2}$ by Lemma 7.2. The result follows from Lemma 3.11 and the diagonal rescaling. \square

References

- [AJSS19] AmirMahdi Ahmadi, Arun Jambulapati, Amin Saberi, and Aaron Sidford. *Perron-Frobenius Theory in Nearly Linear Time: Positive Eigenvectors, M-matrices, Graph Kernels, and Other Applications*, pages 1387–1404. 2019. [9](#)
- [AKM⁺20] AmirMahdi Ahmadi, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision estimation of random walks in small space. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1295–1306, 2020. [2](#), [7](#), [24](#)
- [BSS12] Joshua Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012. [2](#)
- [CGL⁺20a] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1158–1167. IEEE, 2020. [2](#), [5](#)
- [CGL⁺20b] Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond, 2020. [6](#), [11](#), [12](#), [14](#), [21](#), [43](#)
- [CKK⁺18] Michael B. Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. Solving directed laplacian systems in nearly-linear time through sparse lu factorizations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 898–909, 2018. [1](#), [2](#), [9](#)
- [CKP⁺14] Michael B. Cohen, Rasmus Kyng, Jakub W. Pachocki, Richard Peng, and Anup B. Rao. Preconditioning in expectation. *CoRR*, abs/1401.6236, 2014. [1](#)
- [CKP⁺16a] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Aaron Sidford, and Adrian Vladu. Faster algorithms for computing the stationary distribution, simulating random walks, and more. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 583–592, 2016. [1](#), [2](#)
- [CKP⁺16b] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. *CoRR*, abs/1611.00755, 2016. [7](#), [9](#), [11](#), [12](#), [13](#), [31](#), [32](#), [34](#), [35](#), [36](#), [45](#)
- [CKP⁺16c] 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. *CoRR*, abs/1608.03270, 2016. [2](#), [3](#), [9](#)
- [CKP⁺17] Michael B. Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM*

- SIGACT Symposium on Theory of Computing*, STOC 2017, page 410–419, New York, NY, USA, 2017. Association for Computing Machinery. , 1, 2, 4
- [GR99] Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999. 12
 - [JS21] Arun Jambulapati and Aaron Sidford. Ultrasparse ultrasparsifiers and faster laplacian system solvers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 540–559, 2021. 1
 - [KLP⁺15] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. *CoRR*, abs/1512.01892, 2015. 7, 9, 24, 25, 44, 45
 - [KLP⁺16] 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 Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC ’16, page 842–850, New York, NY, USA, 2016. Association for Computing Machinery. 1, 9
 - [KMP11] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-m log n time solver for sdd linear systems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 590–598, 2011. 1
 - [KMP14] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. *SIAM Journal on Computing*, 43(1):337–354, 2014. 1
 - [KOSZ13] 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 Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC ’13, page 911–920, New York, NY, USA, 2013. Association for Computing Machinery. 1
 - [KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582, 2016. 1, 9
 - [KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, may 2004. 12
 - [Pee19] John Peebles. Fast spectral primitives for directed graphs. PhD thesis, Massachusetts Institute of Technology, 2019. 9
 - [PS14] Richard Peng and Daniel A. Spielman. An efficient parallel solver for sdd linear systems. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC ’14, page 333–342, New York, NY, USA, 2014. Association for Computing Machinery. 1, 7
 - [PS21] Richard Peng and Zhuoqing Song. Sparsified block elimination for directed laplacians. *CoRR*, abs/2111.10257, 2021. 24, 25, 44, 45

- [PS22] Richard Peng and Zhuoqing Song. Sparsified block elimination for directed laplacians. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 557–567, New York, NY, USA, 2022. Association for Computing Machinery. , [1](#), [2](#), [4](#), [9](#)
- [RV05] Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 436–447, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [2](#), [7](#)
- [SS11] Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. [2](#)
- [ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 81–90, New York, NY, USA, 2004. Association for Computing Machinery. [1](#)
- [ST11] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. [1](#), [2](#)
- [ST13] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013. [1](#)
- [ST14] 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. [1](#)

A Linear Algebra

Fact A.1. For a block diagonal matrix $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_k \end{pmatrix}$ we have $\|\mathbf{A}\|_2 = \max_i \{\|\mathbf{A}_i\|_2\}$

Proof. Consider the matrix $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \\ & \mathbf{A}_2 \end{pmatrix}$. Then $\|\mathbf{A}\|_2 = \max_{\mathbf{v}: \|\mathbf{v}\|=1} \|\mathbf{A}\mathbf{v}\|_2$. Now every \mathbf{v} is decomposable into $a\mathbf{x}$ and $b\mathbf{y}$ so that $\mathbf{A}\mathbf{v} = \begin{pmatrix} a\mathbf{A}_1\mathbf{x} \\ b\mathbf{A}_2\mathbf{y} \end{pmatrix}$ and $\|x\|_2 = 1$, $\|y\|_2 = 1$ and $a^2 + b^2 = 1$.

But then $\left\| \begin{pmatrix} a\mathbf{A}_1\mathbf{x} \\ b\mathbf{A}_2\mathbf{y} \end{pmatrix} \right\|_2^2 = a^2 \|\mathbf{A}_1\mathbf{x}\|_2^2 + b^2 \|\mathbf{A}_2\mathbf{y}\|_2^2 \leq (a^2 + b^2) \max\{\|\mathbf{A}_1\mathbf{x}\|_2^2, \|\mathbf{A}_2\mathbf{y}\|_2^2\}$. For $k > 2$ the fact follows by induction. \square

B Degree Preserving Undirected Sparsification

We adapt the sparsification routine provided in Section 6.3 of [CGL⁺20b] to be degree preserving. We first state their result.

Theorem B.1 (Corollary 6.4 in [CGL⁺20b]). *There is a deterministic algorithm called SPECTRALSPARSIFY(G, r) that, given an undirected n -node m -edge graph $G = (V, E, \omega)$ with integer edge weights ω bounded by U , and a parameter $1 \leq r \leq O(\log m)$, computes a $(\log m)^{C \cdot r^2}$ -spectral sparsifier H for G , with $|E(H)| \leq O(n \log n \log U)$ for some constant C , in time*

$$O(m^{1+O(1/r)} \cdot (\log m)^{O(r^2)} \log U).$$

Algorithm 10: SPECTRALSPARSIFYDEG(G, γ)

- 1 $r = ((\log n)^{\gamma/2} / \sqrt{2C}) / (\log \log m)$
 - 2 let $\omega_{\min} = \min_{e \in E(G)} \omega(e)$ denote the minimum weight
 - 3 let \hat{G} be
 - 4 $H = \text{SPECTRALSPARSIFY}(G, r)$
 - 5 $\mathbf{A}_{\hat{G}} = \mathbf{A}_H / \text{Exp}(\frac{1}{2} \cdot (\log n)^\gamma) + \underbrace{\mathbf{D}_G - \mathbf{D}_H / \text{Exp}(\frac{1}{2} \cdot (\log n)^\gamma)}_{\text{self loops}}$
 - 6 return \tilde{G}
-

To make the sparsification degree preserving, we simply add self loops.

Lemma B.2. (Degree Preserving Sparsification, Lemma 4.10 restated) *There is a deterministic algorithm*

SPECTRALSPARSIFYDEG(G, γ) that given a parameter $\gamma \in (0, 1)$ and an undirected graph $G = (V, E, \omega)$ with n vertices and m edges such that that $P := \frac{\max_{e \in E} \omega(e)}{\min_{e \in E} \omega(e)} = \text{poly}(n)$ computes \tilde{G} satisfying

1. $\text{Exp}(-(\log n)^\gamma) \mathbf{L}_G \preceq \mathbf{L}_{\tilde{G}} \preceq \text{Exp}((\log n)^\gamma) \mathbf{L}_G$
2. $\text{nnz}(\mathbf{A}) = \tilde{O}(n)$

in time

$$\tilde{O}(m^{1+O(1/(\log n)^{\gamma/2})} \cdot (\log m)^{O((\log n)^\gamma)}) = m^{1+o(1)}.$$

The graph \tilde{G} has self loops and exactly the same degrees as G .

Proof. The runtime directly follows from the runtime of Theorem B.1. Further, since for our choice of r we have $(\log m)^{C \cdot r^2} \leq \text{Exp}(\frac{1}{2} \cdot (\log n)^\gamma)$, we obtain

$$\text{Exp}(-\frac{1}{2} \cdot (\log n)^\gamma) \mathbf{L}_G \preceq \mathbf{L}_H \preceq \text{Exp}(\frac{1}{2} \cdot (\log n)^\gamma) \mathbf{L}_G.$$

This directly allows us to conclude that for all v : $\deg_G(v) \leq \deg_{\tilde{G}}$, and thus the self loops we add are valid and ensure the preservation of degrees. Finally, we have

$$\text{Exp}(-(\log n)^\gamma) \mathbf{L}_G \preceq \mathbf{L}_{\tilde{G}} \preceq \mathbf{L}_G$$

since self loops cancel and thus do not change the directed Laplacian. This proves the desired approximation and concludes our proof. \square

C Sketching the Cholesky Solver

Very recently, new techniques for analysing the error accumulation in sparsified-cholesky-solvers for directed Laplacians lead to an algorithm with almost optimal dependence on the runtime of the sparsification routine [PS21]. In this section, we sketch that our deterministic sparsification routines can also be used to derandomize that framework. We first summarize the framework of [KLP⁺15, PS21].

C.1 Sparsified-Cholesky for Directed Laplacians

Given a bi-partition (F, C) of the vertex set V , the block Cholesky decomposition of an Eulerian Laplacian $\mathbf{L} = \mathbf{L}_{\tilde{G}}$ is given by

$$\mathbf{L} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{L}_{CF} \mathbf{L}_{FF}^{-1} & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{L}_{FF} & \mathbf{0} \\ \mathbf{0} & \text{Sc}[\mathbf{L}]_F \end{pmatrix} \cdot \begin{pmatrix} \mathbf{I} & \mathbf{L}_{FF}^{-1} \mathbf{L}_{FC} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

where $\text{Sc}[\mathbf{L}]_F := \mathbf{L}_{CC} - \mathbf{L}_{CF} \mathbf{L}_{FF}^{-1} \mathbf{L}_{FC}$. The algorithm of [PS21] selects a ρ -RCDD (row-column-diagonally-dominant) block \mathbf{L}_{FF} , and then computes the above decomposition, where the inverse of \mathbf{L}_{FF} is not explicitly computed. Given that it is easy to approximately invert ρ -RCDD blocks using iterative schemes, the main obstruction to apply the inverse of \mathbf{L} is to apply the inverse of $\text{Sc}[\mathbf{L}]_F$.

The augmented matrix view of partial block elimination introduced by [PS21] shows that $\text{Sc}[\mathbf{L}]_F$ can be explicitly approximated using $O(\log \log n)$ -approximate squaring steps with moderate accuracy $\epsilon \approx \frac{1}{\log \log n}$. Since $\text{Sc}[\mathbf{L}]_F$ is another Eulerian Laplacian, and through careful patching its explicitly computed sparse approximation retains this property, the above decomposition can be iterated until C has constant size. Since $|F| = \Omega(|F| + |C|)$, $\Theta(\log n)$ steps suffice.

C.2 Derandomizing the Sparsified-Cholesky Solver

There are three randomized pieces in [PS21].

- The *sparsified squaring* routines from [CKP⁺16b] are used to approximate $\text{Sc}[\mathbf{L}]_F$. We replace these calls with our deterministic sparsified squaring routine (see Section 5).
- In [PS21] *global sparsification* is invoked at the start, and after each squaring to avoid any build up of density. We cannot match this strategy using our global sparsification techniques. Therefore, we only occasionally globally sparsify and recurse (see Section 4 and Section 6).
- The routine for selecting a ρ -RCDD subset is randomized [KLP⁺15, PS21]. We show that this can be done deterministically in Appendix C.3.

Schur complement chains. Since we run our sparsified squaring algorithm without directly following up with global sparsification, the density increases by a factor of $\frac{1}{\epsilon^d}$ after d block Cholesky decomposition steps. Therefore, as in the squaring framework, we cannot afford to go to full depth $\Theta(\log n)$, but have to limit the depth to say $\Theta((\log n)^{1/2})$, such that $\frac{1}{\epsilon^d} = n^{o(1)}$. Then, we invoke our global sparsification technique, and continue on the globally sparsified schur complement.

The recursive algorithm. We apply our decomposition just like [PS21], but whenever we reach a global sparsification point, we have to recursively branch to rectify the error this induced. If we set the global sparsification error to $\text{Exp}(O((\log n)^{1/10}))$ as in the squaring algorithm, we obtain a branching factor of $\text{Exp}(O((\log n)^{1/10}))$ and $\Theta((\log n)^{1/2})$ depth. Therefore, the total branching is bounded by $n^{o(1)}$ and the algorithm runs in almost linear time, since all the matrices involved have an almost linear amount of entries if we globally sparsify at the start.

C.3 Deterministically Finding a ρ -RCDD Subset

Given a directed and Eulerian Graph $\underline{G} = (V, E, \omega)$ on n vertices, we aim to find a set $S \subseteq V$ so that $|S| > \Omega(n)$ and for each vertex s in S

$$\sum_{(v,s) \in E, v \notin S} \omega(v, s) \geq \rho \deg_{\underline{G}}^-(s)$$

and

$$\sum_{(s,v) \in E, v \notin S} \omega(s, v) \geq \rho \deg_{\underline{G}}^+(s).$$

Namely, a ρ -fraction of its (weighted) in-neighbours and a ρ -fraction of its (weighted) out-neighbours are not in S . Such a set S is called a ρ -RCDD subset, since it corresponds to a ρ -RCDD block of the Eulerian Laplacian $\mathbf{L}_{\underline{G}}$. See Figure 7 for an illustration.

One Condition suffices. We first reduce the problem to finding an algorithm that satisfies one of the two conditions.

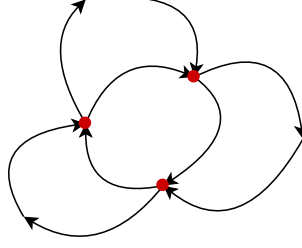


Figure 7: The dark red vertices form a $1/2$ -RCDD subset S , since for each dark red vertex, half the out-edges leave S , and half the in-edges come from outside S .

Lemma C.1. *Given an algorithm for finding a subset S of V for a directed graph \underline{G} such that*

$$\sum_{(v,s) \in E, v \notin S} \omega((v,s)) \geq \rho \deg_{\underline{G}}^-(s)$$

and $|S| = \Omega(n)$ we can find an ρ -RCDD subset S' with $|S'| = \Omega(n)$.

Proof. Use the algorithm to find set S . Then look at the induced subgraph $\underline{G}' = \underline{G}[S]$. Reverse the direction of all edges in \underline{G}' , and use the algorithm again. \square

Satisfying the in-edge Condition. Given the previous section, we are left with the task of eliminating a constant weighted fraction of the in-edges. To do so, we define a potential for every set $S \subset V$.

Definition C.2. Let $\Psi(S) = \sum_{v \in S} \sum_{(u,v) \in E, u \notin S} \frac{\omega((u,v))}{\deg_{\underline{G}}^-(v)}$.

We first make a simple observation.

Lemma C.3. *Given a set S such that $\Psi(S)/|S| \leq \frac{1}{2}$, there is a subset $S' \subseteq S$ with $|S'| \geq \frac{1}{4}|S|$ and for each $s \in S'$*

$$\sum_{(v,s) \in E, v \notin S} \omega((v,s)) \geq \frac{1}{4} \deg_{\underline{G}}^-(s)$$

Proof. We have

$$\begin{aligned} \Psi(S) &= \sum_{v \in S} \sum_{(v,u) \in E, u \notin S} \frac{\omega((u,v))}{\deg_{\underline{G}}^-(v)} \leq \frac{1}{2}|S| \\ \frac{1}{|S|} \sum_{v \in S} \sum_{(v,u) \in E, u \notin S} \frac{\omega((u,v))}{\deg_{\underline{G}}^-(v)} &\leq \frac{1}{2}. \end{aligned}$$

and we aim to show that there exists a set $S' \subseteq S$ so that for $s \in S'$

$$\sum_{(s,u) \in E, u \notin S} \frac{\omega((u,s))}{\deg_{\underline{G}}^-(s)} \leq \frac{3}{4}.$$

But since the average value is below $\frac{1}{2}$ at least a $\frac{1}{4}$ -fraction of the values

$$\sum_{(v,u) \in E, u \in S} \frac{\omega((u,v))}{\deg_{\vec{G}}^-(v)}$$

are below $\frac{3}{4}$ which shows our claim. \square

We are left with having to construct a set S as in Lemma C.3. We define an importance score for each vertex, which we will use as a greedy criterion.

Definition C.4. Let $\text{score}(v, S) = \sum_{(v,u) \in E, u \in S} \frac{\omega((v,u))}{\deg_{\vec{G}}^-(u)} - \sum_{(u,v) \in E, u \notin S} \frac{\omega((u,v))}{\deg_{\vec{G}}^-(v)}$ for $v \in S$.

Lemma C.5. For $S \subseteq V$ we have

$$\sum_{v \in S} \text{score}(v, S) = \Psi(S)$$

Proof. We prove the lemma by induction. Initially, we have

$$\Psi(V) = \sum_{v \in V} \sum_{(u,v) \in E} \frac{\omega((u,v))}{\deg_{\vec{G}}^-(v)} = n = \sum_{v \in V} \text{score}(v, V).$$

Now let's assume our lemma holds for S , and we show it holds for $S \setminus \{l\}$.

$$\begin{aligned} \Psi(S \setminus \{l\}) &= \Psi(S) - \sum_{(u,l) \in E, u \in S} \frac{\omega(u,l)}{\deg^-(l)} - \sum_{(l,u) \in E, u \in S} \frac{\omega(l,u)}{\deg_{\vec{G}}^-(u)} \\ &= \sum_{v \in S} \text{score}(v, S) - \sum_{(u,l) \in E, u \in S} \frac{\omega(u,l)}{\deg^-(l)} + \sum_{(l,u) \in E, u \in S} \frac{\omega(l,u)}{\deg_{\vec{G}}^-(u)} \\ &= \sum_{v \in S \setminus \{l\}} \text{score}(v, S \setminus \{l\}). \end{aligned}$$

Therefore the claim holds by induction on the size of S . \square

Next we introduce our greedy algorithm based on the scores.

Lemma C.6. If Algorithm 11 terminates it returns a set S' so that for all $v \in S$

$$\sum_{(v,u) \in E, u \in S_i} \frac{\omega((u,v))}{\deg_{\vec{G}}^-(v)} \leq \frac{3}{4}$$

and $|S'| \geq \frac{1}{4}|S_i|$.

Proof. Immediately follows from the description of Algorithm 11 and Lemma C.3. \square

Lemma C.7. For all i : $\Psi(S_i) \leq \Psi(S_{i-1}) - \frac{3}{2}$

Algorithm 11: FINDDD(\underline{G})

```

1  $S_0 = V$ 
2  $i = 0$ 
3 while  $\Psi(S_i) > 0.5|S_i|$  do
4    $v_i = \arg \max_{v \in S_i} \text{score}(v, S_i)$ 
5    $S_{i+1} = S_i \setminus \{v_i\}$ 
6    $i = i + 1$ 
7 end
8  $S = S_i; S' = S$ 
9 for  $v \in S'$  do
10  if  $\sum_{(v,u) \in E, u \in S_i} \frac{\omega((u,v))}{\deg_{\underline{G}}^-(v)} > \frac{3}{4}$  then
11     $S' = S' \setminus \{v\}$ 
12  end
13 end
14 return  $S'$ 

```

Proof. For any S we have

$$\begin{aligned}
\Psi(S \setminus \{l\}) &= \Psi(S) - \sum_{(u,l) \in E, u \in S} \frac{\omega(u,l)}{\deg_{\underline{G}}^-(l)} - \sum_{(l,u) \in E, u \in S} \frac{\omega(l,u)}{\deg_{\underline{G}}^-(u)} \\
&= \Psi(S) - \text{score}(l, S) - 1
\end{aligned}$$

and thus the lemma follows from the maximum being at least the average. \square

Lemma C.8. For the set S as in Algorithm 11 we have $|S| \geq \frac{1}{2}|V|$.

Proof. Assume the contrary. Then more than $n/2$ vertices got eliminated, but in iteration $n/2$

$$\Psi(S_{n/2}) \leq \Psi(V) - \frac{3}{2}n < 0.$$

So the while loop must have stopped then, which is a contradiction. \square

Lemma C.9. We can deterministically find a $\frac{1}{4}$ -RCDD subset of an n -vertex m -edge Eulerian graph \underline{G} with at least $\frac{1}{64}n$ vertices in $\tilde{O}(m)$ time.

Proof. Follows from Lemma C.8, Lemma C.6 and the proof of Lemma C.1. \square

D Preconditioning a Cycle with its Transpose Fails

In this section we show that preconditioning the Laplacian of a length 5 cycle with its transpose (the directed Laplacian of the graph with edges reversed) does not lead to converging behaviour on

some inputs. To this end, we consider the Laplacian

$$\mathbf{L} = \begin{pmatrix} 1 & & & & -1 \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & -1 & 1 & \\ & & & -1 & 1 \end{pmatrix}$$

of said cycle. Computing the eigenvalues of $(\mathbf{L}^T)^+ \mathbf{L}$ yields one eigenvalue $\lambda' \approx -0.3 - 0.95i$ (See Figure 8). We conclude that $\mathbf{I}_{\text{im}(\mathbf{L})} - \eta(\mathbf{L}^T)^+ \mathbf{L}$ has an eigenvalue that is approximately $1 + \eta \cdot 0.3 + \eta \cdot 0.95i$, which is strictly larger than 1 in magnitude for any step size $\eta > 0$ ⁶. From this we conclude that $\rho(\mathbf{I}_{\text{im}(\mathbf{L})} - \eta(\mathbf{L}^T)^+ \mathbf{L}) > 1$. For $\mathbf{E} = \mathbf{I}_{\text{im}(\mathbf{L})} - \eta(\mathbf{L}^T)^+ \mathbf{L}$ preconditioned Richardson approximates

$$\mathbf{L}^+ \mathbf{b} = (\mathbf{I}_{\text{im}(\mathbf{L})} + \mathbf{E} + \mathbf{E}^2 + \mathbf{E}^3 \dots)(\mathbf{L}^T)^+ \mathbf{b}$$

by truncating the sum above. From our previous derivation we know that there is a complex vector $\mathbf{v} = (\mathbf{L}^T)^+ \hat{\mathbf{b}}$ (note that \mathbf{L} and \mathbf{L}^T and their inverses all have the same image) for which the sum does not converge. To show that this behaviour is also exhibited by a real vector, we decompose $\mathbf{v} = \mathbf{v}_1 + i\mathbf{v}_2$ into its real and imaginary part and conclude

$$\mathbf{E}\mathbf{v} = \mathbf{E}\mathbf{v}_1 + i\mathbf{E}\mathbf{v}_2.$$

Notice that \mathbf{E} is a real valued matrix, and thus $\mathbf{E}\mathbf{v}_1$ is the real part of $\mathbf{E}\mathbf{v}$. However, from this we can conclude that if $\mathbf{E}^k \mathbf{v}$ is large for some k , then either $\mathbf{E}^k \mathbf{v}_1$ or $\mathbf{E}^k \mathbf{v}_2$ must be large by the triangle inequality.

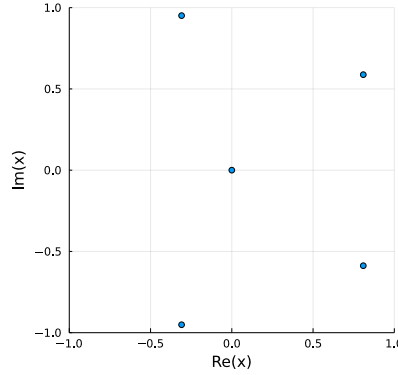


Figure 8: Plot of the eigenvalues of $(\mathbf{L}^T)^+ \mathbf{L}$, where \mathbf{L} is the Laplacian of a directed cycle of length 5 with unit edge weight.

Since \mathbf{L}^T is the Laplacian of the same cycle as \mathbf{L} with reversed edge directions this shows that there are eulerian Laplacians that cannot be used as preconditioners with any step size. This is a significant obstruction for developing a notion of high error sparsification for directed graphs.

⁶A similar argument can show that no complex step size η leads to a converging behaviour by realizing that $(\mathbf{L}^T)^+ \mathbf{L}$ has an eigenvalue in each of the quadrants of the convex plane (See Figure 8).