Derandomizing Directed Random Walks in Almost-Linear Time

Rasmus Kyng
Department of Computer Science
ETH Zurich
Zurich, Switzerland
kyng@inf.ethz.ch

Simon Meierhans
Department of Computer Science
ETH Zurich
Zurich, Switzerland
mesimon@inf.ethz.ch

Maximilian Probst Gutenberg
Department of Computer Science
ETH Zurich
Zurich, Switzerland
maxprobst@inf.ethz.ch

***Abstract*—In this article, we present the first *deterministic* directed Laplacian L systems solver that runs in time almost-linear in the number of non-zero entries of 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 convergence 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 [1] and the sparsified Cholesky-based framework of [2].

***Index Terms*—Algorithms, Algorithm design and analysis, Derandomization, Directed Laplacians, Random Walks**

I. 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 [3], that has since been split into three parts [4]–[6], 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 analyzing 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 [7] 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 [1], 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 [8]. The runtime was improved to nearly-linear time by Cohen-Kelner-Kyng-Peebles-Peng-Rao-Sidford [9] and further improved by Peng and Song [2] very recently. Both operate within sparsified-Cholesky frameworks for solving undirected Laplacian linear equations: [9] uses the framework of Kyng and Sachdeva [10] and [2] uses the framework of Kyng-Lee-Peng-Sachdeva-Spielman [11].

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 [1][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 preconditoner 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 [1] and [2] with an almost-linear runtime.

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

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

2575-8454/22/$31.00 ©2022 IEEE
DOI 10.1109/FOCS54457.2022.00046

A. Prior Work

a) Undirected Laplacian Solvers: The first nearly-linear time Laplacian solver [4]–[6] sparked the development of a field producing a whole host of distinct algorithms for solving Laplacian linear equations [8], [10]–[16]. Of these, our algorithm is most similar to [8] and its directed counterpart [1], which works by repeatedly squaring the adjacency matrix. We also derandomize an alternative Cholesky-factorization based framework, which is motivated by [11] in the undirected setting and was recently translated to the directed setting by [2].

b) 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 [5], and later strengthened and simplified by Spielman and Srivastava [17]. 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 [18]. However, no deterministic almost-linear time algorithms are known for $(1 \pm \epsilon)$-spectral sparsification. Recently, Chuzhoy-Gao-Li-Nanongkai-Peng-Saranurak [19] 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.

c) Directed Laplacian Solvers: Before [1], [7] 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 [1] or a sparsified-Cholesky approach [2], [9]. Both rely on spectral sparsification techniques developed in [1].

d) 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 [20]. They show that the Rozeman and Vadhan [21] 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.

e) Applications of Directed Laplacian Solvers: There are numerous applications of directed Laplacian solvers given in Section 7 of [22], most of which are deterministic reductions to directed Laplacian system solving. The deterministic ones include

- solving large classes of linear systems,
- computing personalized 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 problems[2].

B. Our Contributions

We introduce the first notion of crude approximation for Eulerian Laplacians $\boldsymbol{L}$. It is defined via the suitability as a preconditioner in the Richardson iteration and sparse approximations are constructed using *partial symmetrization* to increase robustness. This technique allows us to trade off additional Richardson iterations for a behavior 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 I.1 (Informal version of our main result)**.** *Given an Eulerian Laplacian $\boldsymbol{L}_{\vec{G}}$ associated with a strongly connected Eulerian Graph $\vec{G}$ with n vertices and m edges, a vector $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{L}_{\vec{G}})$, and a parameter $\epsilon \in (0,1)$ the algorithm* SolveEulerian$(\boldsymbol{L}_{\vec{G}}, \epsilon)$ *in time $m^{1+o(1)} \log \epsilon^{-1}$ computes a vector $\boldsymbol{x}$ satisfying*

$$\left\| \boldsymbol{x} - \boldsymbol{L}_{\vec{G}}^{+} \boldsymbol{b} \right\|_{\boldsymbol{U}_{\boldsymbol{L}_{\vec{G}}}} \leq \epsilon \left\| \boldsymbol{L}_{\vec{G}}^{+} \boldsymbol{b} \right\|_{\boldsymbol{U}_{\boldsymbol{L}_{\vec{G}}}}$$

where $\boldsymbol{U}_{\boldsymbol{A}} = (\boldsymbol{A} + \boldsymbol{A}^T)/2$ for any square matrix $\boldsymbol{A}$.

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 I.2. *Given a directed Laplacian $\boldsymbol{L}_{\vec{G}} = \boldsymbol{D}_{\vec{G}} - \boldsymbol{A}_{\vec{G}}^T$ associated with a directed Graph $\vec{G}$ with n vertices and m edges, a vector $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{L}_{\vec{G}})$, a parameter $\epsilon \in (0,1)$ and an upper bound $\kappa \geq \max(\kappa(\boldsymbol{D}_{\vec{G}}), \kappa(\boldsymbol{L}_{\vec{G}}))$ there is an algorithm* SolveFull$(\boldsymbol{L}_{\vec{G}}, \epsilon)$ *that in time $m^{1+o(1)} \log^{O(1)}(\kappa\epsilon^{-1})$ computes a vector $\vec{\boldsymbol{x}}$ satisfying*

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

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

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 [22].

a) 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

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

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.

b) 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. We include such a construction in the appendix of the full version [23] of our article.

II. 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 $\underrightarrow{G}$ with m-edges in time close to m and 2) *sparsifying the square graph* $\underrightarrow{G}^2$ in time close to m. The square graph $\underrightarrow{G}^2$ is the graph with adjacency matrix $\boldsymbol{A}_{\underrightarrow{G}} \boldsymbol{D}_{\underrightarrow{G}}^{-1} \boldsymbol{A}_{\underrightarrow{G}}$ where $\boldsymbol{A}_{\underrightarrow{G}}$ denotes the adjacency matrix of $\underrightarrow{G}$ and $\boldsymbol{D}_{\underrightarrow{G}}$ denotes the diagonal matrix containing the degrees of $\underrightarrow{G}$. For some graphs the running time of 2) is sublinear in the number of edges of the sparsified graph $\underrightarrow{G}^2$ 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 [1], and sketch the sparsified-Cholesky solver [2] in the appendix of our full paper[3].

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 [1] 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 $\boldsymbol{M}$, by preconditioning with a matrix $\boldsymbol{N}$. Roughly speaking, if, in an appropriate norm $\|\cdot\|$, we have $\left\|\boldsymbol{I}_{\mathrm{im}(\boldsymbol{M})} - \boldsymbol{N}^+\boldsymbol{M}\right\| \leq 1 - \alpha$, then we can solve the linear equation in $\boldsymbol{M}$ to high accuracy using $\widetilde{O}(1/\alpha)$ preconditioned Richardson iterations where we apply $\boldsymbol{M}$ and $\boldsymbol{N}^+$ once per iteration. A priori, it may be surprising that our approach relies on distinguishing the case $1 - \left\|\boldsymbol{I}_{\mathrm{im}(\boldsymbol{M})} - \boldsymbol{N}^+\boldsymbol{M}\right\| \geq n^{o(-1)}$ from the case $1 - \left\|\boldsymbol{I}_{\mathrm{im}(\boldsymbol{M})} - \boldsymbol{N}^+\boldsymbol{M}\right\| \approx 0$, 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 $\boldsymbol{A}$ is preconditioned by $\boldsymbol{B}$ and $\boldsymbol{B}$ is preconditioned by $\boldsymbol{C}$, then a solver for $\boldsymbol{A}$ can be constructed with access to $\boldsymbol{B}$ and $\boldsymbol{C}^+$ 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.

A. Global Sparsification

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

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

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

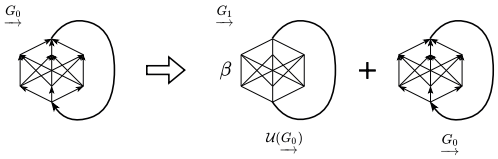

Fig. 1: Obtaining $\underrightarrow{G}_1$ from $\underrightarrow{G}_0$ via β-partial symmetrization. Given the graph $\underrightarrow{G}_0$ as depicted on the left hand side, we add β times its undirectification. This could double the amount of edges, but the graph $\underrightarrow{G}_1$ has much less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

[3] [2] also relies on randomness for finding a linear sized almost independent subset. We provide a simple procedure to derandomize this step in the full version of our paper.

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

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

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

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

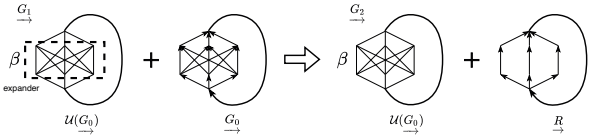

Fig. 2: Obtaining $\underrightarrow{G}_2$ from $\underrightarrow{G}_1$ via patching. The dashed box contains the complete bipartite graph, which is a good expander. We use this information to drastically sparsify the directed part on the same vertex set V', only ensuring that the degrees match up by 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.

b) From $\underrightarrow{G}_1$ to $\underrightarrow{G}_2$: Patching Expander Parts: The graph $\underrightarrow{G}_1 = \beta \cdot \mathcal{U}(\underrightarrow{G}_0) + \underrightarrow{G}_0$ is made up of two parts: an undirected graph $\beta \cdot \mathcal{U}(\underrightarrow{G}_0)$ and a directed graph $\underrightarrow{G}_0$. In this step we aim to sparsify the directed part by constructing a sparse graph $\underrightarrow{R}$ with the same in- and out-degrees as $\underrightarrow{G}_0$. Then $\underrightarrow{G}_2 = \beta \cdot \mathcal{U}(\underrightarrow{G}_0) + \underrightarrow{R}$. Our strategy for obtaining $\underrightarrow{R}$ relies on the structure of the undirected graph $\mathcal{U}(\underrightarrow{G}_0)$ and deterministic expander decompositions as presented in [19]. Our main technical observation here is that given a vertex set V' so that $\mathcal{U}(\underrightarrow{G}_0)[V']$ is a good enough expander, we can replace the directed graph $\underrightarrow{G}_0[V']$ in $\underrightarrow{G}_1$ with *any other* directed graph $\underrightarrow{R}'$, as long as $\underrightarrow{G}_0[V']$ and $\underrightarrow{R}'$ have exactly the same in- and out-degrees, retaining a close approximation between $\underrightarrow{G}_1 - \underrightarrow{G}_0[V'] + \underrightarrow{R}'$ and $\underrightarrow{G}_1$. Our algorithm for constructing $\underrightarrow{R}$ first removes the weighted structure from $\mathcal{U}(\underrightarrow{G}_0)$ 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 $\underrightarrow{R}'$ as introduced above. Summing up all of these yields $\underrightarrow{R}$. 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.

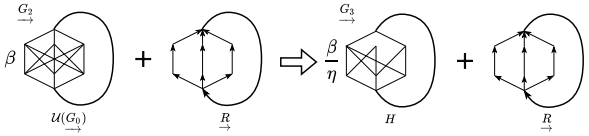

Fig. 3: Obtaining $\underrightarrow{G}_3$ from $\underrightarrow{G}_2$ via undirected sparsification. The undirected part $\beta \cdot \mathcal{U}(\underrightarrow{G})$ of $\underrightarrow{G}_2$ is sparsified using previously known spectral sparsification routines for undirected graphs. Since both remaining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

c) From $\underrightarrow{G}_2$ to $\underrightarrow{G}_3$: Scaling an Undirected Sparsifier: Finally we sparsify the undirected part $\beta \cdot \mathcal{U}(\underrightarrow{G}_0)$ obtaining a sparse undirected graph H and let $\underrightarrow{G}_3 = H + \underrightarrow{R}$. This can be achieved via known theorems for deterministic low-accuracy undirected graph sparsification provided by [24]. To obtain a good preconditioner we have to scale H, but not $\underrightarrow{R}$, 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 $\underrightarrow{R}$, which is a sparse directed graph. See Figure 3 for an illustration.

B. Sparsified Squaring

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

$$\begin{aligned}\boldsymbol{L}_{\underrightarrow{G}^2} &= \boldsymbol{D}_{\underrightarrow{G}} - \underbrace{\boldsymbol{A}^T_{\underrightarrow{G}} \boldsymbol{D}^{-1}_{\underrightarrow{G}} \boldsymbol{A}^T_{\underrightarrow{G}}}_{\text{adjacency matrix}} \\ &= \boldsymbol{D}_{\underrightarrow{G}} - \sum_{i=1}^{n} \underbrace{\frac{1}{\boldsymbol{D}_{\underrightarrow{G}}(i,i)} (\boldsymbol{A}(i,:))^T \cdot (\boldsymbol{A}(:,i))^T}_{\boldsymbol{A}_i^T}.\end{aligned}$$

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

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

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 $\boldsymbol{L}$ to high accuracy ϵ, by adapting a procedure from [11] to be degree preserving. We call the sparse bipartite graph we obtain $\tilde{\boldsymbol{L}}$. Then we obtain a sparsified version $\tilde{\boldsymbol{A}}_i^T$ of $\boldsymbol{A}_i^T$ by simply taking the bottom left block of $\tilde{\boldsymbol{L}}$. We show that $\tilde{\boldsymbol{L}}_i = \boldsymbol{D}_i - \tilde{\boldsymbol{A}}_i^T$ is a ϵ/Φ^2-approximation of $\boldsymbol{L}_i$, 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 $\boldsymbol{L}_{\underrightarrow{\tilde{G}}^2} = \sum_{i=1}^{n} \tilde{\boldsymbol{L}}_i$ of $\boldsymbol{L}_{\underrightarrow{G}^2}$

with $\mathrm{nnz}(\boldsymbol{L}_{\underrightarrow{G}^2}) \leq O(\mathrm{nnz}(\boldsymbol{L}_{\underrightarrow{G}})\epsilon^{-4})$ in almost linear time. Similar deterministic squaring techniques were developed for small space algorithms [20], [21], 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.

C. The Squaring Framework

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

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

for $\boldsymbol{b}$ orthogonal to the kernel of $(\boldsymbol{I} - \boldsymbol{\mathcal{A}})^+$. Given a $1/\operatorname{poly}(n)$ lower bound on the smallest eigenvalue λ_* of $\boldsymbol{I} - \boldsymbol{\mathcal{A}}$, 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

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

for $\boldsymbol{b}$ orthogonal to the kernel of $(\boldsymbol{I} - \boldsymbol{\mathcal{A}})^+$ which is at the center of the squaring mechanism of [25], the algorithm our squaring solver resembles most. Their squaring scheme is in turn inspired by the squaring solver for symmetric Laplacians presented in [8].

This leaves us with the task of solving linear equations in $\boldsymbol{I} - \boldsymbol{\mathcal{A}}^2$, which is another normalized Laplacian. However, this is the normalized Laplacian of the square graph $\underrightarrow{G}^2$, 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 behaviors we consider the normalized adjacency matrix $\boldsymbol{\mathcal{A}}_l^{(\alpha)} := \alpha \boldsymbol{I} + (1-\alpha)\boldsymbol{\mathcal{A}}_j$, which can be interpreted as adding self loops proportional to the out-degrees of $\underrightarrow{G}$.

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

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

as we can ensure that $\boldsymbol{b}$ is orthogonal to the known kernel of $(\boldsymbol{I} - \boldsymbol{\mathcal{A}})^+$. However, the repeated sparsification accumulates an error proportional to ϵe^d, and it is imperative that it stays below 1 such that $\boldsymbol{Z}$ is an approximate pseudoinverse of $\boldsymbol{I} - \boldsymbol{\mathcal{A}}$. Therefore, we have to choose ϵ proportional to e^{-d}. We conclude from the previous subsection that $\mathrm{nnz}(\boldsymbol{A}_d) = O(\mathrm{nnz}(\boldsymbol{\mathcal{A}}) e^{4d^2})$. 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 $e^{4d^2} = n^{o(1)}$.

b) 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 $\boldsymbol{\mathcal{A}}_0^{(0)} = \boldsymbol{D}_{\underrightarrow{G}}^{+/2} \boldsymbol{A}_{\underrightarrow{G}} \boldsymbol{D}_{\underrightarrow{G}}^{+/2}$ for $i = 0, ..., \Theta((\log n)^{2/3})$ we iteratively construct:

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

We discuss these collections of squaring chains linked by global sparsification in section 6 of our full paper. 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.

c) The Recursive Algorithm: Our global sparsification routine allows us to solve linear equations in $\boldsymbol{I} - \boldsymbol{\mathcal{A}}_0^{(i)}$ by solving $\mathrm{Exp}(O((\log n)^{1/10}))$ linear equations in $\boldsymbol{I} - \boldsymbol{\mathcal{A}}_0^{(i+1)}$. Since linear equations in $\boldsymbol{I} - \boldsymbol{\mathcal{A}}_0^{(\Theta(\log n)^{2/3})}$ are easy to solve using standard iterative procedures, this is the depth of our recursion. Therefore, the total amount of branches is $\mathrm{Exp}(O((\log n)^{2/3+1/10})) = n^{o(1)}$. 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.

D. The Sparsified-Cholesky Framework

Very recently [2] showed that the framework of [11] directly works for Eulerian Laplacians by developing new tools for analyzing 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 [9], [10], but [11] and [2] 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

[5]Notice that [11] is concerned with the undirected case.

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 the appendix of the full version of our paper.

E. Reduction to the Eulerian Setting with bounded Condition Number

Previous work [22], [25] reduced solving linear equations in directed Laplacians $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}^T$ 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 $\kappa(\boldsymbol{D})$ and $\kappa(\boldsymbol{L})$ (See Appendix D and F of [25] and Sections 5, 7.1 and 7.3 of [22]). They use that edge weights are polynomially bounded in the proof of Lemma C.3 in Appendix C of [25]. Different reductions to the Eulerian case were presented by Ahmadinejad-Jambulapati-Saberi-Sidford [26] and in the thesis of Peebles [27].

III. Preliminaries

A. Linear Algebra

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

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

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

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

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

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

B. Graphs

a) 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 $\underrightarrow{G} = (V, E, \omega)$, 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 $V(\underrightarrow{G}), E(\underrightarrow{G})$ and $\omega_{\underrightarrow{G}}$ to avoid ambiguity. We let $\omega^{\max}$ and $\omega^{\min}$ denote the maximum and minimum edge weight respectively.

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

c) 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 $\underrightarrow{G}$ we define $\underrightarrow{G}[X]$ analogously.

C. Graph Laplacians

a) General Notation: For a undirected graph $G = (V, E, \omega)$ we denote its (graph) Laplacian as $\boldsymbol{L}_G = \boldsymbol{D}_G - \boldsymbol{A}_G$ where $\boldsymbol{D}_G$ is the diagonal matrices containing the degrees and $\boldsymbol{A}_G(i,j) = \boldsymbol{A}_G(j,i) = \omega((i,j))$. Generalizing this notation to directed graphs $\underrightarrow{G} = (V, E, \omega)$, we let $\boldsymbol{L}_{\underrightarrow{G}} = \boldsymbol{D}_{\underrightarrow{G}} - \boldsymbol{A}^T_{\underrightarrow{G}}$ where the adjacency matrix is given by $\boldsymbol{A}_{\underrightarrow{G}}(i,j) = \omega((i,j))$ and the diagonal matrix $\boldsymbol{D}_{\underrightarrow{G}}(i,i) = \sum_j \boldsymbol{A}_{\underrightarrow{G}}(i,j)$ contains the out degrees. Naturally $\mathbf{1}^T\boldsymbol{L}_{\underrightarrow{G}} = \mathbf{0}$. 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.

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

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

d) Induced subgraphs and submatrices: The reader should note that for a Laplacian $\boldsymbol{L}_{\underrightarrow{G}}$ the matrices $\boldsymbol{L}_{\underrightarrow{G}[X]}$ and $\boldsymbol{L}_{\underrightarrow{G}}[X]$ 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 $\boldsymbol{D}_{\underrightarrow{G}[X]}(i,i) \leq \boldsymbol{D}_{\underrightarrow{G}}[X](i,i)$.

D. Directed Graph Approximation

Definition III.1 (Asymmetric Matrix Approximation, Definition 3.1 in [25])**.** *A (possibly asymmetric) matrix $\tilde{\boldsymbol{A}}$ is said to be an ϵ-matrix-approximation of $\boldsymbol{A}$ if*

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

Remark III.2. *Notice that Definition III.1 is not symmetric.*

E. Expanders

Whenever we use the term expander, we mean expander graphs with respect to *conductance*. We follow the notational conventions of [24].

Definition III.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 $\mathrm{vol}_G(S) := \sum_{v\in S}\sum_{u\in V}\omega(u, v)$. Then we define the conductance*

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

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

Lemma III.5 (Non Uniform Degree Bound, see [24])**.** *Given a Φ-expander G with Laplacian $\boldsymbol{L}_G = \boldsymbol{D}_G - \boldsymbol{A}_G^T$ we have*

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

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

Next we state the definition of the expander decomposition for unweighted and undirected graphs in the notation of [24].

Definition III.6 (See Section 6 of [24], Proposed by [28], [29])**.** *A (ϵ, Φ)-expander decomposition of a undirected and unweighted graph $G = (V, E)$ is a partition $\mathcal{P} = \{V_1, ..., V_k\}$ of the vertex set V such that for all $i \in [k]$ the conductance of $G[V_i]$ is at least Φ and $\sum_{i=1}^k \delta_G(V_i) \leq \epsilon\,\mathrm{vol}_G(V)$.*

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

Theorem III.7 (See Corollary 7.7 in [24])**.** *There is a deterministic algorithm* ExpDecomp*(G, γ) that, given an undirected and unweighted graph $G = (V, E)$ with m edges and a constant $\gamma \in (0, 1)$, computes a $(1/2, \frac{1}{\mathrm{Exp}((\log n)^\gamma)})$-expander decomposition in time $m^{1+o(1)}$.*

F. Preconditioned Richardson

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

Lemma III.8 (Preconditioned Richardson, Lemma 4.2 in [25])**.** *Let $\boldsymbol{b} \in \mathbb{R}^n$ and $\boldsymbol{M}, \boldsymbol{Z}, \boldsymbol{U} \in \mathbb{R}^{n\times n}$ such that $\boldsymbol{U}$ is symmetric positive definite, $\ker(\boldsymbol{U}) \subseteq \ker(\boldsymbol{M}) = \ker(\boldsymbol{M}^T) = \ker(\boldsymbol{Z}) = \ker(\boldsymbol{Z}^T)$, and $\boldsymbol{b} \in \mathrm{im}(\boldsymbol{M})$. Then N iterations of preconditioned Richardson with step size $\eta > 0$, result in a vector $\boldsymbol{x}_N =$ PreconRichardson$(\boldsymbol{M}, \boldsymbol{Z}, \boldsymbol{b}, \eta, N)$ so that*

$$\left\|\boldsymbol{x}_N - \boldsymbol{M}^+\boldsymbol{b}\right\|_{\boldsymbol{U}} \leq \left\|\boldsymbol{I}_{\mathrm{im}(\boldsymbol{M})} - \eta\boldsymbol{Z}\boldsymbol{M}\right\|^N_{\boldsymbol{U}\to\boldsymbol{U}} \left\|\boldsymbol{M}^+\boldsymbol{b}\right\|_{\boldsymbol{U}}.$$

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

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

Definition III.9 (Approximate Pseudoinverse, Definition 4.3 in [25])**.** *Matrix $\boldsymbol{Z}$ is an ϵ-approximate-pseudoinverse of matrix $\boldsymbol{M}$ with respect to a* PSD *matrix $\boldsymbol{U}$, if $\ker(\boldsymbol{U}) \subseteq \ker(\boldsymbol{M}) = \ker(\boldsymbol{M})^T = \ker(\boldsymbol{Z}) = \ker(\boldsymbol{Z}^T)$, and*

$$\left\|\boldsymbol{I}_{\mathrm{im}(\boldsymbol{M})} - \boldsymbol{Z}\boldsymbol{M}\right\|_{\boldsymbol{U}\to\boldsymbol{U}} \leq \epsilon.$$

Algorithm 1: PreconRichardson$(\boldsymbol{M}, \boldsymbol{Z}, \boldsymbol{b}, \eta, N)$

$\boldsymbol{x}_0 = \boldsymbol{0}$
for *i = 0, ..., N - 1* **do** $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \eta\boldsymbol{Z}(\boldsymbol{b} - \boldsymbol{M}\boldsymbol{x}_i)$;
return $\boldsymbol{x}_N$

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

Lemma III.10 (Part of Lemma B.9 in [25])**.** *If $\boldsymbol{L}$ is a matrix with $\ker(\boldsymbol{L}) = \ker(\boldsymbol{L}^T) = \ker(\boldsymbol{U}_{\boldsymbol{L}})$, and $\boldsymbol{U}_{\boldsymbol{L}}$ is positive semidefinite, then for any matrix $\boldsymbol{A}$ with the same left and right kernels as $\boldsymbol{L}$ we have*

$$\|\boldsymbol{A}\|_{\boldsymbol{U}_{\boldsymbol{L}}\to\boldsymbol{U}_{\boldsymbol{L}}} \leq \left\|\boldsymbol{U}_{\boldsymbol{L}}^{+/2}\boldsymbol{L}\boldsymbol{A}\boldsymbol{U}_{\boldsymbol{L}}^{+/2}\right\|_2$$

Lemma III.11 (Part of Lemma B.2 in [25])**.** *For all $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and symmetric* PSD *$\boldsymbol{M}, \boldsymbol{N} \in \mathbb{R}^{n\times n}$ such that $\ker(\boldsymbol{M}) \subseteq \ker(\boldsymbol{A}^T)$ and $\ker(\boldsymbol{N}) \subseteq \ker(\boldsymbol{A})$ we have*

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

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

Lemma III.12 (Part of Lemma B.4 in [25])**.** *For a* PSD *diagonal matrix $\boldsymbol{D}$ and any matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$*

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

IV. 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 $\underrightarrow{G}$.

Definition IV.1. *For an Eulerian directed graph $\underrightarrow{G}$ we call $\mathcal{U}^{(\beta)}(\underrightarrow{G}) := \beta\cdot\mathcal{U}(\underrightarrow{G}) + \underrightarrow{G}$ its β-partial-symmetrization.*

Remark IV.2. $\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})} = \beta \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}}} + \boldsymbol{L}_{\underrightarrow{G}}$.

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 $\underrightarrow{G} = \underrightarrow{G}_0$, we let $\underrightarrow{G}_1 = \mathcal{U}^{(\beta)}(\underrightarrow{G}) = \beta \cdot \mathcal{U}(\underrightarrow{G}) + \underrightarrow{G}$ be the graph obtained from $\underrightarrow{G}_0$ by β-partial symmetrization. Then, surprisingly, $\boldsymbol{L}_{\underrightarrow{G}_1}$ can be used as a preconditioner for solving linear equations in $\boldsymbol{L}_{\underrightarrow{G}_0}$ in time $O(\beta)$ using Richardson. We call $\mathcal{U}^{(\beta)}(\underrightarrow{G})$ the β-partial-symmetrization of $\underrightarrow{G}$.
2) A partial-symmetrization is naturally interpreted as the sum of an undirected graph $\beta \cdot \mathcal{U}(\underrightarrow{G})$ and a directed graph $\underrightarrow{G}$. We expander decompose the undirected graph $\beta \cdot \mathcal{U}(\underrightarrow{G})$ into parts $V_1, V_2, ..., V_k$. Then a simple greedy patching scheme can be used to sparsify the induced subgraphs $\underrightarrow{G}[V_i]$ 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 $\underrightarrow{G}_2 = \beta \cdot \mathcal{U}(\underrightarrow{G}) + \underrightarrow{R}$ can then be used as a preconditioner for $\boldsymbol{L}_{\underrightarrow{G}_1}$
3) Lastly, the undirected graph $\beta \cdot \mathcal{U}(\underrightarrow{G})$ can be sparsified via previously known deterministic algorithms presented in [24]. We obtain $\underrightarrow{G}_3 = \frac{\beta}{\eta}\tilde{G} + \underrightarrow{R}$ which in turn is a preconditioner for $\underrightarrow{G}_2$.

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

Definition IV.3. *(Sparsification quadruple) We call strongly connected n-vertex Eulerian graphs $\underrightarrow{G}_0, \underrightarrow{G}_1, \underrightarrow{G}_2, \underrightarrow{G}_3$ a (γ, β, η)-quadruple for some constant $\gamma \in (0,1)$ if*

1) $\boldsymbol{L}^{+}_{\underrightarrow{G}_i}$ is a $\left(1 - \frac{1}{\mathrm{Exp}(O((\log n)^\gamma))}\right)$-approximate pseudoinverse of $\boldsymbol{L}_{\underrightarrow{G}_{i-1}}$ with respect to $\boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_i}}$ for $i = 1, 2, 3$.

2) $\frac{1}{\mathrm{Exp}(O((\log n)^\gamma))} \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_{i-1}}} \preceq \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_i}} \preceq \mathrm{Exp}(O((\log n)^\gamma)) \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_{i-1}}}$ for $i \in \{1, 2, 3\}$.

3) $|E(\underrightarrow{G}_3)| = \tilde{O}(n)$. $|E(\underrightarrow{G}_i)| \leq 2|E(\underrightarrow{G}_0)| + \tilde{O}(n)$ for $i = 1, 2$.

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

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 IV.4 (Global Sparsification)**.** *For every m-edge, strongly connected Eulerian graph $\underrightarrow{G} = \underrightarrow{G}_0$ and a constant $\gamma \in (0,1)$ the routine $\underrightarrow{G}_1, \underrightarrow{G}_2, \underrightarrow{G}_3 = \textsc{GlobalSparsification}(\underrightarrow{G}, \gamma)$ yields a $(\gamma, \beta = \mathrm{Exp}(O((\log n)^\gamma)), \eta = \mathrm{Exp}(-3(\log n)^\gamma))$-quadruple $\underrightarrow{G}_0, \underrightarrow{G}_1, \underrightarrow{G}_2, \underrightarrow{G}_3$. The runtime is $m^{1+o(1)}$.*

Remark IV.5. *While in the description of our algorithm, β scales linearly in $O\left(\log\left(\frac{\omega^{max}_{\underrightarrow{G}}}{\omega^{min}_{\underrightarrow{G}}}\right)\right)$, we assume throughout the paper that $\beta = \tilde{O}(1) \cdot \mathrm{Exp}(2(\log n)^\gamma)$ is fixed to a global upper bound as $\log\left(\frac{\omega^{max}_{\underrightarrow{G}}}{\omega^{min}_{\underrightarrow{G}}}\right) = \tilde{O}(1)$ for all graphs $\underrightarrow{G}$ we work with. This avoids clutter in the analysis.*

Algorithm 2: $\textsc{GlobalSparsification}(\underrightarrow{G}, \gamma)$

$\beta = L \cdot \mathrm{Exp}(2 \cdot (\log n)^\gamma)$ for $L = 128 \cdot 20 \cdot P \cdot \log n$ and $P = \left\lceil \log\left(\frac{\omega^{\max}_{\underrightarrow{G}}}{\omega^{\min}_{\underrightarrow{G}}}\right)\right\rceil$.
$\eta = \mathrm{Exp}(-3 \cdot (\log n)^\gamma)$
$\underrightarrow{G}_1 = \beta\mathcal{U}(\underrightarrow{G}) + \underrightarrow{G}$; // Note that $\underrightarrow{G}_1 = \mathcal{U}^{(\beta)}(\underrightarrow{G})$.
$\underrightarrow{R} = \textsc{SparsifyDirected}(\underrightarrow{G}, \gamma)$
$\underrightarrow{G}_2 = \beta\mathcal{U}(\underrightarrow{G}) + \underrightarrow{R}$
$\tilde{G} = \textsc{SpectralSparsifyDeg}(\mathcal{U}(\underrightarrow{G}), \gamma)$
$\underrightarrow{G}_3 = \frac{\beta}{\eta}\tilde{G} + \underrightarrow{R}$
return $\underrightarrow{G}_1, \underrightarrow{G}_2, \underrightarrow{G}_3$

A. Preconditioning with the Partial-Symmetrization

Our next lemma shows that $\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})}$ is a good preconditioner in terms of Lemma III.8.

Lemma IV.6. *For every Eulerian Laplacian $\boldsymbol{L}_{\underrightarrow{G}_0}$ the matrix $\boldsymbol{L}^{+}_{\underrightarrow{G}_1}$ is an $(1 - \frac{1}{1+\beta})$-approximate pseudoinverse of $\boldsymbol{L}_{\underrightarrow{G}_0}$ with respect to $\boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_1}}$ if $\underrightarrow{G}_1 = \mathcal{U}^{(\beta)}(\underrightarrow{G}_0)$.*

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

$$\left\| \boldsymbol{I}_{\mathrm{im}(\boldsymbol{L}_{\underrightarrow{G}})} - \boldsymbol{L}^{+}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})} \boldsymbol{L}_{\underrightarrow{G}} \right\|_{\boldsymbol{U}_{\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})}} \to \boldsymbol{U}_{\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})}}} \leq \left\| \boldsymbol{U}^{+/2}_{\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})}} (\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})} - \boldsymbol{L}_{\underrightarrow{G}}) \boldsymbol{U}^{+/2}_{\boldsymbol{L}_{\mathcal{U}^{(\beta)}(\underrightarrow{G})}} \right\|_2$$

by Lemma III.10. With Remark IV.2 we conclude

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

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

B. Sparsifying the Directed Part

Given $\boldsymbol{L}_{\underrightarrow{G}_1} = \beta \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}}} + \boldsymbol{L}_{\underrightarrow{G}}$, we aim to obtain a sparse directed graph $\underrightarrow{R}$ with the same in- and out-degrees as $\underrightarrow{G}$ so that the directed Laplacian $\boldsymbol{L}_{\underrightarrow{G}_2} = \beta \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}}} + \boldsymbol{L}_{\underrightarrow{R}}$ preconditions $\boldsymbol{L}_{\underrightarrow{G}}$. 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 $\underrightarrow{G}^{(i)}$ with close to uniform edge weight such that $\sum_i \underrightarrow{G}^{(i)} = \underrightarrow{G}$.

We let $H^{(i)}$ denote the unweighted and undirected graph with the same edges as $\underrightarrow{G}^{(i)}$. Then we layer $j = 1, ..., O(\log n)$ undirected and unweighted expander decompositions on this graph, where each of them peels of at least $1/2$ of the remaining edges $E_r^{(i,j)}$. This procedure computes $O(\log n)$ partitions $V_1^{(i,j)}, ..., V_{k(i,j)}^{(i,j)}$ 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 $\underrightarrow{G}^{(i)}$ that do not go across sets in the partition $V_1^{(i,j)}, ..., V_{k(i,j)}^{(i,j)}$ into the graph $\underrightarrow{G}^{(i,j)}$ and remove them from the set of remaining edges.

The expander structure allows us to sparsify $\underrightarrow{G}^{(i,j)}$ via a greedy patching scheme obtaining $\underrightarrow{\tilde{G}}^{(i,j)}$. Finally, we sum up across layers and buckets and obtain $\underrightarrow{R} = \sum_{i,j} \underrightarrow{\tilde{G}}^{(i,j)}$. 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 analyzes this algorithm.

Lemma IV.7. *Let* $\underrightarrow{R} = \textsc{SparsifyDirected}(\underrightarrow{G}, \gamma)$ *for* $\gamma \in (0,1)$ *constant. Then* $\boldsymbol{L}^{+}_{\underrightarrow{G}_2} = (\beta \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}}} + \boldsymbol{L}_{\underrightarrow{R}})^{+}$ *is a* $1/2$*-approximate pseudoinverse of* $\boldsymbol{L}_{\underrightarrow{G}_1} = \boldsymbol{L}_{\mathcal{U}^{\beta}(\underrightarrow{G})}$ *with respect to* $\boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_2}}$ *for* $\beta = \tilde{O}(1) \cdot \mathrm{Exp}(2 \cdot (\log n)^{\gamma})$*. Further, the graph* $\underrightarrow{R}$ *has* $\tilde{O}(n)$ *edges and the same in- and out-degrees as* $\underrightarrow{G}$*.*

The proof of Lemma IV.7 relies on analyzing the error incurred by sparsifying each individual expander decomposition, i.e. the cost of replacing $\underrightarrow{G}^{(i,j)}$ with $\underrightarrow{\tilde{G}}^{(i,j)}$. Then we conclude by just summing up the error. The next lemma carefully analyzes 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 IV.8. *In the context of* SPARSIFY*() in Algorithm 3 we have*

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

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

Proof. $\underrightarrow{G}^{(i,j)}$ and $\underrightarrow{\tilde{G}}^{(i,j)}$ 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 $\underrightarrow{G}^{(i,j)} - \underrightarrow{\tilde{G}}^{(i,j)}$. We apply Lemma III.11 twice and obtain

$$\left\| \boldsymbol{U}^{+/2}_{L_{\underrightarrow{G}}} (\boldsymbol{L}_{\underrightarrow{G}^{(i,j)}} - \boldsymbol{L}_{\underrightarrow{\tilde{G}}^{(i,j)}}) \boldsymbol{U}^{+/2}_{L_{\underrightarrow{G}}} \right\|_2$$

Algorithm 3: SPARSIFYDIRECTED($\underrightarrow{G}, \gamma$) and subroutines SPARSIFY() and PATCH()

Algorithm SPARSIFYDIRECTED($\underrightarrow{G}, \gamma$)
  $P = \left\lceil \log \left(\frac{\omega^{\max}_{\underrightarrow{G}}}{\omega^{\min}_{\underrightarrow{G}}} \right) \right\rceil$
  for $i = 1, ..., P$ **do**
    $E^{(i)} = \{e \in E(\underrightarrow{G}) : \omega^{min}_{\underrightarrow{G}} \cdot 2^{i-1} \leq \omega_{\underrightarrow{G}}(e) < \omega^{min}_{\underrightarrow{G}} \cdot 2^{i}\}$
    $\underrightarrow{\tilde{G}}^{(i)} = \textsc{Sparsify}(\underrightarrow{G}^{(i)} = (V(\underrightarrow{G}), E^{(i)}, \omega_{\underrightarrow{G}}))$
  return $\underrightarrow{R} = \sum_{i=1}^{P} \underrightarrow{\tilde{G}}^{(i)}$
Procedure SPARSIFY($\underrightarrow{G}^{(i)}$)
  Let $H^{(i)}$ denote the unweighted and undirected graph with the same edges as $\mathcal{U}(\underrightarrow{G}^{(i)})$
  for $j = 1, ..., 10 \log n$ **do**
    $E_r^{(i,j)} = E(H^{(i)}) - \bigcup_{l=1}^{j-1} E(H^{(i,j)})$; $\underrightarrow{E}_r^{(i,j)} = E(\underrightarrow{G}^{(i)}) - \bigcup_{l=1}^{j-1} E(\underrightarrow{G}^{(i,j)})$
    $V_1^{(i,j)}, ..., V_{k(i,j)}^{(i,j)} = \textsc{ExpDecomp}((\boldsymbol{V}(H^{(i)}), E_r^{(i,j)}), \gamma)$
    $E(H^{(i,j)}) = \bigcup_{p=1}^{k(i,j)} \{(u,v) \in E_r^{(i,j)} : u \in V_p^{(i,j)} \wedge v \in V_p^{(i,j)}\}$
    $E(\underrightarrow{G}^{(i,j)}) = \bigcup_{p=1}^{k(i,j)} \{(u,v) \in \underrightarrow{E}_r^{(i,j)} : u \in V_p^{(i,j)} \wedge v \in V_p^{(i,j)}\}$
    $H^{(i,j)} = (V(H^{(i)}), E(H^{(i,j)}))$; $\underrightarrow{G}^{(i,j)} = (V(\underrightarrow{G}^{(i)}), E(\underrightarrow{G}^{(i,j)}), \omega_{\underrightarrow{G}^{(i)}})$
    $\underrightarrow{\tilde{G}}^{(i,j)} = \sum_{p=1}^{k(i,j)} \textsc{Patch}(\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}])$
  return $\underrightarrow{\tilde{G}}^{(i)} = \sum_{j=1}^{10 \log n} \underrightarrow{\tilde{G}}^{(i,j)}$
Procedure PATCH($\underrightarrow{H}$)
  Let $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{R}^n_{\geq 0}$ so that $\boldsymbol{a}(v) = \deg^{+}_{\underrightarrow{H}}(v)$ and $\boldsymbol{b}(v) = \deg^{-}_{\underrightarrow{H}}(v)$. // Note $\|\boldsymbol{a}\|_1 = \|\boldsymbol{b}\|_1$.
  $E(\underrightarrow{\tilde{H}}) = \emptyset$; $\omega_{\underrightarrow{\tilde{H}}}(e) = 0$ for all e.
  while $\|\boldsymbol{a}\| \neq 0$ **do**
    Let i and j be arbitrary such that $\boldsymbol{a}(i) > 0$ and $\boldsymbol{b}(j) > 0$.
    $w = \min\{\boldsymbol{a}(i), \boldsymbol{b}(j)\}$; $\boldsymbol{a}(i) = \boldsymbol{a}(i) - w$; $\boldsymbol{b}(j) = \boldsymbol{b}(j) - w$
    $E(\underrightarrow{\tilde{H}}) = E(\underrightarrow{\tilde{H}}) \cup \{(i,j)\}$; $\omega(i,j) = w$
  return $\underrightarrow{\tilde{H}}$

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

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

$$\begin{aligned}&\left\|L^{+/2}_{H^{(i,j)}}(L_{\underrightarrow{G}^{(i,j)}} - L_{\underrightarrow{\tilde{G}}^{(i,j)}})L^{+/2}_{H^{(i,j)}}\right\|_2 \\ &\leq \max_p \left\|L^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}(L_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]} - L_{\underrightarrow{\tilde{G}}^{(i,j)}[V_p^{(i,j)}]})L^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\right\|_2 \end{aligned} \tag{3}$$

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

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

for $\boldsymbol{d}_p^{(i,j)}$ being the degree vector of $H^{(i,j)}[V_p^{(i,j)}]$, where i) is by Lemma III.11 and ii) is by the expansion of $H^{(i,j)}[V_p^{(i,j)}]$ and Lemma III.5. Let $\boldsymbol{x}, \boldsymbol{y} \perp \boldsymbol{1}$ be maximizing the right hand side of the previous inequality. Then, also $\boldsymbol{x}' = \boldsymbol{x} - \frac{\boldsymbol{x}^T\boldsymbol{d}_p^{(i,j)}}{\left\|\boldsymbol{d}_p^{(i,j)}\right\|_2}\boldsymbol{1}$ and $\boldsymbol{y}' = \boldsymbol{y} - \frac{\boldsymbol{y}^T\boldsymbol{d}_p^{(i,j)}}{\left\|\boldsymbol{d}_p^{(i,j)}\right\|_2}\boldsymbol{1}$ are maximizing. We obtain

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

where the last equality is by Lemma III.11. Next we upper bound

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

and

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

By Lemma III.12 we have

$$\begin{aligned}&\left\|D^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}L_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}D^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\right\| \\ &\leq \max\left\{\left\|L_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}D^{+}_{H^{(i,j)}[V_p^{(i,j)}]}\right\|_1, \left\|L^T_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}D^{+}_{H^{(i,j)}[V_p^{(i,j)}]}\right\|_1\right\}.\end{aligned}$$

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

$$\begin{aligned}&\deg_{\omega^{\min}_{\underrightarrow{G}}\cdot 2^{i+1}\cdot H^{(i,j)}[V_p^{(i,j)}]}(v) \\ &\quad\geq \max\left\{\deg^+_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}(v), \deg^-_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}(v)\right\}\end{aligned}$$

we have

$$\left\|\boldsymbol{D}^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\boldsymbol{L}_{\underrightarrow{G}^{(i,j)}[V_p^{(i,j)}]}\boldsymbol{D}^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\right\| \leq \omega^{\min}_{\underrightarrow{G}}\cdot 2^{i+2}. \tag{6}$$

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

$$\left\|\boldsymbol{D}^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\boldsymbol{L}_{\underrightarrow{\tilde{G}}^{(i,j)}[V_p^{(i,j)}]}\boldsymbol{D}^{+/2}_{H^{(i,j)}[V_p^{(i,j)}]}\right\| \leq \omega^{\min}_{\underrightarrow{G}}\cdot 2^{i+2}. \tag{7}$$

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

$$\begin{aligned}&\left\|(\boldsymbol{U}_{L_{\underrightarrow{G}}})^{+/2}(\underrightarrow{G}^{(i,j)} - \underrightarrow{\tilde{G}}^{(i,j)})(\boldsymbol{U}_{L_{\underrightarrow{G}}})^{+/2}\right\|_2 \\ &\quad\leq 128\cdot\mathrm{Exp}(2\cdot(\log n)^\gamma)\end{aligned}$$

which concludes our proof. □

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

Lemma IV.9. $|E(\underrightarrow{\tilde{G}}^{(i,j)})| = O(n)$.

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

Proof of Lemma IV.7. Notice that each expander decomposition peels of half of the edges, and thus every edge is part of an unique expander part by the end of the procedure SPARSIFY() in Algorithm 3. Thus our algorithm exactly preserves the in- and out-degrees of $\underrightarrow{G}_1 = \beta\cdot\mathcal{U}(\underrightarrow{G}) + \underrightarrow{G}$, since each individual patching exactly preserves degrees. Since the graph $\underrightarrow{G}_2 = \beta\cdot\mathcal{U}(\underrightarrow{G}) + \underrightarrow{R}$ remains connected the null-spaces are unaltered. Further, since $\underrightarrow{R}$ is the sum of $O(\tilde{1})$ graphs with $O(n)$ edges the total amount of edges of $\underrightarrow{R}$ is bounded by $\tilde{O}(n)$.

Finally, we show the approximation bound. By Lemma III.10 we have

$$\begin{aligned}&\left\|\boldsymbol{I}_{\mathrm{im}(\mathcal{U}^{(\beta)}(\underrightarrow{G}))} - \boldsymbol{L}^+_{\underrightarrow{G}_2}\boldsymbol{L}_{\underrightarrow{G}_1}\right\|_{\boldsymbol{U}_{L_{\underrightarrow{G}_2}}\to\boldsymbol{U}_{L_{\underrightarrow{G}_2}}} \\ &\leq \left\|\boldsymbol{U}^{+/2}_{L_{\underrightarrow{G}_2}}(\boldsymbol{L}_{\underrightarrow{R}} - \boldsymbol{L}_{\underrightarrow{G}})\boldsymbol{U}^{+/2}_{L_{\underrightarrow{G}_2}}\right\|.\end{aligned}$$

We use Lemma III.11 to obtain

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

Using Lemma III.11 again we have

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

where i) follows from Lemma IV.8 and ii) is by $\beta \geq 128 \cdot 20 \cdot P \cdot \log n \cdot \operatorname{Exp}(2 \cdot (\log n)^\gamma)$. Chaining the inequalities shows the approximation statement and concludes our proof. □

C. Sparsifying the Undirected Part

The final task we have left is to sparsify the undirected graph $\mathcal{U}(\underrightarrow{G})$. This can be more or less directly achieved by employing a sparsification theorem presented in [24]. Mainly for notational convenience in our algorithm, we adapt this sparsifcation technique to be degree preserving in the full paper and state the resulting lemma here.

Lemma IV.10. *(Degree Preserving Sparsification) 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* $P := \frac{\max_{e\in E}\omega(e)}{\min_{e\in E}\omega(e)} = poly(n)$ *computes* $\tilde{G}$ *satsifying*

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

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 IV.11. *There exists a routine* $\tilde{G} =$ SPECTRALSPARSIFYDEG$(\mathcal{U}(\underrightarrow{G}), \gamma)$ *that given the undirected graph* $\mathcal{U}(\underrightarrow{G})$ *computes an undirected graph* $\tilde{G}$ *with* $\tilde{O}(n)$ *edges so that* $\boldsymbol{L}^+_{\underrightarrow{G}_3} = (\frac{\beta}{\eta}\boldsymbol{L}_{\tilde{G}} + \boldsymbol{L}_{\underrightarrow{R}})^+$ *is an* $\left(1 - \frac{1}{2\cdot\operatorname{Exp}(4\cdot(\log n)^\gamma)}\right)$*-approximate pseudoinverse of* $\boldsymbol{L}_{\underrightarrow{G}_2} = \beta \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}}} + \boldsymbol{L}_{\underrightarrow{R}}$ *with respect to* $\boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_3}}$*. The degrees of* $\tilde{G}$ *and* $\mathcal{U}(\underrightarrow{G})$ *are the same.*

Proof. The sparsity and degree preservation follows directly from Lemma IV.10. To show the approximation property, we use Lemma III.10 and obtain

$$\begin{aligned}
&\left\| \boldsymbol{I}_{\operatorname{im}(\boldsymbol{L}_{\underrightarrow{G}})} - \boldsymbol{L}^+_{\underrightarrow{G}_3}\boldsymbol{L}_{\underrightarrow{G}_2} \right\|_{\boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_3}} \to \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{G}_3}}} \\
&\leq \left\| \boldsymbol{U}^{+/2}_{\boldsymbol{L}_{\underrightarrow{G}_3}} \left(\frac{\beta}{\eta}\boldsymbol{L}_{\tilde{G}} - \beta \boldsymbol{L}_{\mathcal{U}(\underrightarrow{G})}\right) \boldsymbol{U}^{+/2}_{\boldsymbol{L}_{\underrightarrow{G}_3}} \right\|_2. \qquad (8)
\end{aligned}$$

Then we use Lemma III.11 twice with $\frac{\beta}{\eta}\boldsymbol{L}_{\tilde{G}} \preceq \frac{\beta}{\eta}\boldsymbol{L}_{\tilde{G}} + \boldsymbol{U}_{\boldsymbol{L}_{\underrightarrow{R}}}$ to obtain

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

Next we compute

$$\begin{aligned}
&\left\| \boldsymbol{L}^{+/2}_{\tilde{G}}(\boldsymbol{L}_{\tilde{G}} - \eta \boldsymbol{L}_{\mathcal{U}(\underrightarrow{G})})\boldsymbol{L}^{+/2}_{\tilde{G}} \right\| \qquad (10) \\
&= \left\| \boldsymbol{I}_{\operatorname{im}(\boldsymbol{L}_{\tilde{G}})} - \eta \boldsymbol{L}^{+/2}_{\tilde{G}} \boldsymbol{L}_{\mathcal{U}(\underrightarrow{G})} \boldsymbol{L}^{+/2}_{\tilde{G}} \right\|_2. \qquad (11)
\end{aligned}$$

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

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

where $\lambda_*(\boldsymbol{M})$ denotes the smallest non-zero eigenvalue of $\boldsymbol{M}$. We first lower bound $\lambda_*(\boldsymbol{M})$. By Lemma IV.10 we have

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

and thus $\operatorname{Exp}(-(\log n)^\gamma) \leq \lambda_*(\boldsymbol{M})$. We obtain $\|\boldsymbol{M}\|_2 \leq \operatorname{Exp}(2 \cdot (\log n)^\gamma)$ in an analogous way from Lemma IV.10. Then, we use $\eta = \operatorname{Exp}(-3(\log n)^\gamma) \leq \frac{\lambda_*(\boldsymbol{M})}{\|M\|_2^2}$ to conclude.

$$\left\| \boldsymbol{I}_{\operatorname{im}(\boldsymbol{L}_{\tilde{G}})} - \eta \boldsymbol{M} \right\|_2 \leq 1 - \frac{1}{2 \cdot \operatorname{Exp}(4 \cdot (\log n)^\gamma)} \qquad (12)$$

where we use $\sqrt{1-\epsilon} \leq 1 - \epsilon/2$ for $\epsilon \in (0, 1)$. Chaining inequalities (8), (9), (11) and (12) shows the desired approximate pseudoinverse property and finishes the proof. □

D. Proof of Lemma IV.4

Now that we have assembled the pieces we are ready to prove Lemma IV.4, the main lemma of this section.

Proof of Lemma IV.4. We show each point in the enumeration separately.

1) The statement for $i = 1$ is by Lemma IV.6 and $\beta = \tilde{O}(1) \cdot \mathrm{Exp}(2(\log n)^{\gamma})$, for $i = 2$ it is by Lemma IV.7 and for $i = 3$ it is by Lemma IV.11.
2) The analysis of the condition number is deferred to the full version of our paper.
3) Clearly, $\underrightarrow{G}_1$ has at most twice as many edges as $\underrightarrow{G} = \underrightarrow{G}_0$. Further, $|E(\underrightarrow{G}_2)| \leq |E(\underrightarrow{G}_1)| + |E(\underrightarrow{R})|$ and $|E(\underrightarrow{R})| = \tilde{O}(n)$ by Lemma IV.7. Finally $|E(\underrightarrow{G}_3)| \leq |E(\underrightarrow{\tilde{G}})| + |E(\underrightarrow{R})| = \tilde{O}(n)$ by Lemma IV.11 and Lemma IV.7. .
4) Adding β times the undirectifaction 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 IV.7 and Lemma IV.11 and the extra scaling by $\frac{1}{\eta}$ of the undirected part of $\underrightarrow{G}_3$

Finally, the runtime follows from Theorem III.7. □

References

[1] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, "Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, (New York, NY, USA), p. 410–419, Association for Computing Machinery, 2017. 1, 2, 3

[2] R. Peng and Z. Song, "Sparsified block elimination for directed laplacians," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, (New York, NY, USA), p. 557–567, Association for Computing Machinery, 2022. 1, 2, 3, 5

[3] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, (New York, NY, USA), p. 81–90, Association for Computing Machinery, 2004. 1

[4] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 1–26, 2013. 1, 2

[5] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011. 1, 2

[6] D. A. Spielman and S.-H. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014. 1, 2

[7] M. B. Cohen, J. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 583–592, 2016. 1, 2

[8] R. Peng and D. 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, (New York, NY, USA), p. 333–342, Association for Computing Machinery, 2014. 1, 2, 5

[9] M. B. Cohen, J. Kelner, R. Kyng, J. Peebles, R. Peng, A. B. Rao, and A. 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)*, pp. 898–909, 2018. 1, 2, 5

[10] R. Kyng and S. Sachdeva, "Approximate gaussian elimination for laplacians - fast, sparse, and simple," in *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 573–582, 2016. 1, 2, 5

[11] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman, "Sparsified cholesky and multigrid solvers for connection laplacians," in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16, (New York, NY, USA), p. 842–850, Association for Computing Machinery, 2016. 1, 2, 4, 5

[12] I. Koutis, G. L. Miller, and R. Peng, "Approaching optimality for solving sdd linear systems," *SIAM Journal on Computing*, vol. 43, no. 1, pp. 337–354, 2014. 2

[13] I. Koutis, G. L. Miller, and R. Peng, "A nearly-m log n time solver for sdd linear systems," in *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 590–598, 2011. 2

[14] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu, "A simple, combinatorial algorithm for solving sdd systems in nearly-linear time," in *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, (New York, NY, USA), p. 911–920, Association for Computing Machinery, 2013. 2

[15] M. B. Cohen, R. Kyng, J. W. Pachocki, R. Peng, and A. B. Rao, "Preconditioning in expectation," *CoRR*, vol. abs/1401.6236, 2014. 2

[16] A. Jambulapati and A. Sidford, "Ultrasparse ultrasparsifiers and faster laplacian system solvers," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 540–559, 2021. 2

[17] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011. 2

[18] J. Batson, D. A. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012. 2

[19] J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak, "A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond," in *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020* (S. Irani, ed.), pp. 1158–1167, IEEE, 2020. 2, 4

[20] A. Ahmadinejad, J. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. Vadhan, "High-precision estimation of random walks in small space," in *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 1295–1306, 2020. 2, 5

[21] E. Rozenman and S. Vadhan, "Derandomized squaring of graphs," in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques* (C. Chekuri, K. Jansen, J. D. P. Rolim, and L. Trevisan, eds.), (Berlin, Heidelberg), pp. 436–447, Springer Berlin Heidelberg, 2005. 2, 5

[22] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. Sidford, and A. Vladu, "Faster algorithms for computing the stationary distribution, simulating random walks, and more," *CoRR*, vol. abs/1608.03270, 2016. 2, 6

[23] R. Kyng, S. Meierhans, and M. P. Gutenberg, "Derandomizing directed random walks in almost-linear time," *CoRR*, vol. abs/2208.10959, 2022. 3

[24] J. Chuzhoy, Y. Gao, J. Li, D. Nanongkai, R. Peng, and T. Saranurak, "A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond," *CoRR*, vol. abs/1910.08025, 2020. 4, 7, 8, 11

[25] M. B. Cohen, J. A. Kelner, J. Peebles, R. Peng, A. B. Rao, A. Sidford, and A. Vladu, "Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs," *CoRR*, vol. abs/1611.00755, 2016. 5, 6, 7

[26] A. Ahmadinejad, A. Jambulapati, A. Saberi, and A. Sidford, *Perron-Frobenius Theory in Nearly Linear Time: Positive Eigenvectors, M-matrices, Graph Kernels, and Other Applications*, pp. 1387–1404. 2019. 6

[27] J. Peebles, "Fast spectral primitives for directed graphs," vol. PhD thesis, Massachusetts Institute of Technology, 2019. 6

[28] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," *J. ACM*, vol. 51, p. 497–515, may 2004. 7

[29] O. Goldreich and D. Ron, "A sublinear bipartiteness tester for bounded degree graphs," *Combinatorica*, vol. 19, no. 3, pp. 335–373, 1999. 7