Solving 1-Laplacians in Nearly Linear Time: Collapsing and Expanding a Topological Ball *

Michael B. Cohen[†], Brittany Terese Fasy[‡], Gary L. Miller[§], Amir Nayyeri[§], Richard Peng[†], Noel Walkington[§]

October 14, 2013

Abstract

We present an efficient algorithm for solving a linear system arising from the 1-Laplacian of a collapsible simplicial complex with a known collapsing sequence. When combined with a result of Chillingworth, our algorithm is applicable to convex simplicial complexes embedded in $\mathbb{R}^3$. The running time of our algorithm is nearly-linear in the size of the complex and is logarithmic on its numerical properties.

Our algorithm is based on projection operators and combinatorial steps for transferring between them. The former relies on decomposing flows into circulations and potential flows using fast solvers for graph Laplacians, and the latter relates Gaussian elimination to topological properties of simplicial complexes.

1 Introduction

Over the past two decades, substantial progress has been made in designing very fast linear system solvers for the case of symmetric diagonally dominate systems. These solvers have been shown to substantially speedup the worst case times for many problems with applications to image processing, numerical analysis, machine learning, and maximum flows in graphs. These problems reduce to approximation algorithms for solutions to a graph Laplacian. Progress in finding fast solvers for general symmetric systems has been more elusive; see related work below. In this paper, we consider solving a natural generalization of the graph Laplacian: the 1-Laplacian.

Recall that an undirected graph $G = (V, E)$ with n vertices and m edges can be viewed as a one-dimensional complex; that is, it consists of zero-simplices (vertices) and one-simplices (edges). There is a natural map from each oriented edge to its boundary (two vertices). This gives an n by m boundary matrix, which we denote by ∂_1, where each row corresponds to a vertex and each column corresponds to an edge; see Figure 5 in the Appendix. The (i, j) entry is $1, -1$, 0 if the i^{th} vertex is a head, tail, or neither of the j^{th} edge, respectively. The weighted graph Laplacian is defined as $\partial_1 W_1 \partial_1^T$. We use $\Delta_0 = \partial_1 \partial_1^T$ to denote the unweighted graph Laplacian, also known as the 0-Laplacian.

Suppose the K_2 is a two-dimensional complex, i.e., a collection of oriented triangles, edges, and vertices. The (signed) weighted sum of k-simplices is a *k-chain*. For edges, one interpretation of the weights is electric flow, where positive flow is in the direction of the edge and negative flow is in the opposite direction. We then compute the boundary of the k-chain, which is a $(k-1)$-chain. For example, the weights on triangles induce weights on edges via the boundary operator ∂_2, and weights on edges induce weights on triangles via the co-boundary operator ∂_2^T; see Figure 1. Weights on edges induced from weights on triangles can be interpreted as a flow along the boundary of the triangle. Weights on tetrahedra induced from weights on triangles can be interpreted as a flux in the perpendicular direction to the triangle; see Figure 2.

This paper focuses on solving the 1-Laplacian corresponding to a restricted class of two-complexes. Letting K_2 be a two-complex, and d_1 be a one-chain in K_2, we define the central problem addressed in this paper:

Problem 1.1. *Approximate the solution f_1 to the following system of equations:*

$$\Delta_1 f_1 = d_1,$$

where $\Delta_1 = \partial_2 \partial_2^T + \partial_1^T \partial_1$ is the 1-Laplacian.

For convenience, we define the operators $\Delta_1^{\uparrow} = \partial_2 \partial_2^T$ and $\Delta_1^{\downarrow} = \partial_1^T \partial_1$.

*This work was partially supported by the National Science Foundation under grant number CCF-1018463 and CCF-1065106.

[†]Massachusetts Institute of Technology
[‡]Tulane University
[§]Carnegie Mellon University

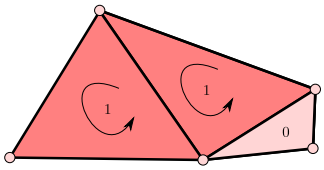


Figure 1: We add the face circulations in order to obtain the cumulative circulation around the boundary of the region shown in dark pink; here, the flow on the internal edge cancels. Algebraically, this is equivalent to applying the boundary operator to a two-chain.

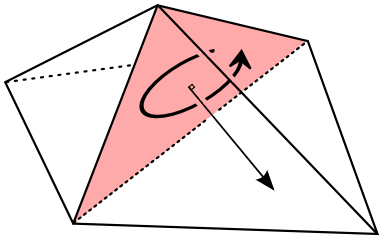


Figure 2: A unit weight on the oriented triangle induces a flux in the direction perpendicular to the triangle, as indicated above.

A key observation used in this paper is that the spaces $\mathrm{im}(\Delta_1^{\uparrow})$ and $\mathrm{im}(\Delta_1^{\downarrow})$ are orthogonal. Further, if K_2 has trivial homology, these two subspaces span all one-chains (i.e., all flows). We use the term *bounding cycle* to refer to the elements of $\mathrm{im}\,\partial_2$, which in our case is equal to $\mathrm{im}\,\Delta_1^{\uparrow}$. The space of bounding cycles coincide with the space of *circulations* if the homology is trivial. We will also use the terms *co-bounding chain* and *potential flow* for $\mathrm{im}\,\partial_1^T$, which is the same subspace as for $\mathrm{im}\,\Delta_1^{\downarrow}$; see Section 3. As we will see later, this decomposition is part of the *Hodge decomposition*.

In order to solve 1-Laplacians, we split the problem into two parts:

$$\Delta_1^{\uparrow} f_1 = \partial_2 \partial_2^T f_1 = d_1^{(c)} \tag{1.1}$$

and

$$\Delta_1^{\downarrow} f_1 = \partial_1^T \partial_1 f_1 = d_1^{(p)}, \tag{1.2}$$

where $d_1^{(c)}$ and $d_1^{(p)}$ the projections of d_1 onto the boundary (circulation) subspace and co-boundary (potential flow) subspace, respectively. We reduce (1.2) to the graph Laplacian, and focus the majority of this paper on handling (1.1). Note that we do not find the exact decomposition of d_1 into $d_1^{(c)}$ and $d_1^{(p)}$, due to the fact that known fast graph Laplacian solvers use iterative methods. Thus, the error analysis is an important contribution of this paper.

Our approach to solving (1.1) is to decompose the problem into two steps:

1. Find any solution f_2 to $\partial_2 f_2 = d_1^{(c)}$.
2. Solve the system $\partial_2^T f_1 = f_2$ for f_1.

Step one is extremely easy in one lower dimension, i.e., when solving $\partial_1 f_1 = d_0$ were d_0 is orthogonal to the all-ones vector. Given a spanning tree, the initial value at an internal node n is uniquely determined by the initial values of the leaf nodes or the subtree rooted at n. The process of determining these values is precisely back substitution in linear algebra.

On the other hand, step one seems to be much more intricate when we look for a two-chain to generate the given one-chain $d_1^{(c)}$. Restricting the input complex allows us to apply results from simple homotopy theory [Coh73]; see Section 5. In higher dimensions, collapsible complexes seem to be the analog of the tree that we used in graph Laplacians.

Finally, step two can be solved for the case of a convex three-complex using duality to reduce the problem to a graph Laplacian.

Paper Outline. In the rest of this section, we briefly survey related results to solving discrete Laplace equations. We continue by presenting some basic background material in Section 2. In Section 3, we present orthogonal decompositions of one-chains and describe the related fast projection operators. An algorithm, which exploits the known collapsibility of the input complex, is described in Sections 4 and 5. Finally, extensions of the current paper are briefly discussed in Section 6.

1.1 Motivation and Related Results.

In this subsection, we present an overview of some related work of solving linear systems and their applications.

Solvers. More than 20 years ago, Vaidya [Vai91] observed that spanning trees can be used as good preconditioners for graph Laplacians. This observation led to Alon et al. [AKPW95] to use a low stretch spanning tree as a preconditioner. The long history of solvers for the graph Laplacian matrix (and, more generally, for symmetric diagonally dominant matrices) culminated with the first nearly-linear time solver [Spi04]. More recently, significant progress has been made in making this approach practical [KMP10, KMP11, KOSZ13, LS13].

Strong empirical evidence over several decades shows that most linear systems can be solved in time much faster than the state of the art worst-case bound of $O(n^{\omega})$ for direct methods [Wil12] and

$O(nm)$ for iterative methods [HS52], where n and m are the dimension and the number of nonzero entries of the matrix, respectively. The important open question here is whether ideas from graph Laplacian solvers leads to fast solvers for a wider classes of linear systems. A glance at the literature, e.g., [Axe94, BCPT05, Dai08], shows that the current nearly-linear time solvers exist for only small class of matrices (namely, those with a small factor-width), while a vast number of systems of interest do not have nearly-linear solvers.

Applications. Using solvers for inear systems as subroutines for graph algorithms has led to state of the art algorithms for a variety of classical graph optimization problems in both the sequential and parallel setting [BGK+13, Dai08, KM09, CKM+11].

Further, they have motivated faster solvers for applications in scientific computing such as Poisson equations [BHV08] and two dimensional trusses [DS07]. We believe this work can be considered as a first step towards solving vector Poisson equations in a similar asymptotic running time.

Discrete Hodge decomposition of the chain spaces has found many applications in literature, including statistical ranking [JLYY11], electromagnetism and fluid mechanics [DKT08]. Friedman [Fri98] used the idea of Hodge decomposition in computing Betti numbers (the rank of homology groups) that, in general, reduces to linear algebraic questions such as computing the Smith normal form of boundary matrices [Mun30] and requires matrix multiplication time [EP14]. The special cases that can be solved faster are for embedded simplicial complexes in $\mathbb{R}^3$ [Del93, DG98, Epp03] and for an output-sensitive result [CK13]. In this paper, we only work with a special case of Hodge Decomposition,the discrete Helmholtz decomposition, where the underlying space has trivial homology.

2 Background

In this section, we review background from linear algebra and algebraic topology. For more details, we refer the interested reader to Strang [Str93] and Hatcher [Hat01].

2.1 Matrices.

An $n \times m$ matrix A can be thought of as a linear operator from $\mathbb{R}^n$ to $\mathbb{R}^m$. A square matrix A is *positive semidefinite* if for all vectors $x \in \mathbb{R}^n$ we have $x^T Ax \geq 0$. Let A be any matrix realizing a linear map $X \to Y$. With a slight abuse of notation, we let A denote both the linear map and the matrix. It is known that Y is decomposed into orthogonal subspaces $\mathbf{im}(A)$ and $\mathbf{null}(A^T)$; indeed, this fact is widely known as the *Fundamental Theorem of Linear Algebra*. The identity of $\mathbf{im}(AA^T)$ and $\mathbf{im}(A)$ as well as the following projection lemma are significant implications of this decomposition that we use repeatedly in this paper.

Lemma 2.1. *Let $A : X \to Y$ and let $y \in Y$. Then, the projection of y onto $\mathbf{im}(A)$ is $A(A^TA)^+A^Ty$, where M^+ denotes the pseudo-inverse of M.*

In our analysis, we bound errors using the 2-norm and matrix norms. The *A-norm* of a vector $v \in \mathbb{R}^n$ is defined, using a positive semidefinite matrix A, as $||v||_A = \sqrt{v^T Av}$. The *two-norm* of a vector $v \in \mathbb{R}^n$ is defined as $||v||_2 = ||v||_I = \sqrt{v^T v}$, where I is the identity matrix. Also, the two-norm of a matrix A is $\max_{x \neq 0} \left(\left\| x^T A \right\|_2 / \|x\|_2 \right)$.

We use $\kappa(A)$ to denote the *condition number* of A, which is the ratio of the maximum to the minimum singular values of A. Much of our error analysis relies on a partial order between matrices. Specifically, we use $A \preceq B$ to denote that $B - A$ is positive semidefinite. We make repeated use of the following known fact about substituting the intermediate term in symmetric product of matrices.

Lemma 2.2 (Composition of Bounds). *Let $a \geq 1$. For any matrices V, A, B, with VAV^T and VBV^T defined, if $A \preceq B \preceq aA$, then $VAV^T \preceq VBV^T \preceq aVAV^T$.*

2.2 Simplicial Complexes.

A k-simplex σ can be viewed as an ordered set of $k+1$ vertices. For example, a simplex σ can be written as $\sigma = [v_0, \ldots, v_k]$; in this case, we write $\dim(\sigma) = k$. A *face* τ of σ is a simplex obtained by removing one or more of the vertices defining σ. A *simplicial complex* K is a collection of simplices such that any face of a simplex in K is also contained in K and that the intersection of any two simplices is a face of both. The dimension of a simplicial complex is the maximum dimension of its composing simplices. In this paper, we use the term *k-complex* to refer to a k-dimensional simplicial complex.

Our systems of equations are based on simplicial three-complexes that are piecewise linearly embedded in $\mathbb{R}^3$. Such an embedding maps a zero-simplex to a point, a one-simplex to a line segment, a two-simplex to a triangle and a three-simplex to a tetrahedron. An embedding of a simplicial complex is *convex* if the union of the images of its simplices $|K|$ is convex. We use the phrase *convex simplicial complex* to refer to a simplicial complex together with a convex embedding of it. If $|K|$ is homeomorphic to a topological space $\mathbb{X}$, then we say that K triangulates $\mathbb{X}$. In particular, we will often assume that K triangulates a three-ball;

that is $|K|$ is homeomorphic to the unit ball, given by $\{x : x \in \mathbb{R}^3, ||x||_2 \leq 1\}$.

2.3 Chains and Boundary Operators. We define a function $f\colon K_k \to \mathbb{R}$, which assigns a real number to each k-simplex of K; we can think of this as a labeling on the k-simplices. The set of all such functions forms a vector space over $\mathbb{R}$ that is known as the k-(co)chain group, and is denoted by $C_k = C_k(K)$. In this paper, we are interested in solving a linear system of the form $Ax_k = b_k$, where x_k and b_k are both k-chains and x_k is unknown.

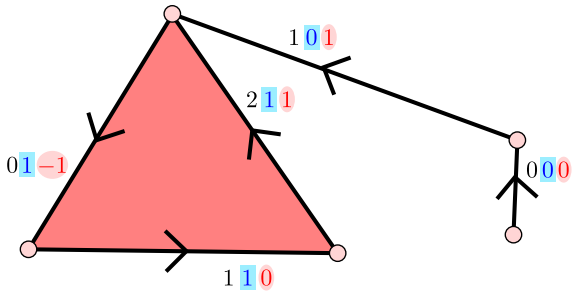


Figure 3: Theorem 3.1 states that every one-chain (e.g., the labeling given in black) can be decomposed into two parts: $\mathbf{im}(\partial_2)$ (the labeling with blue boxes) and $\mathbf{im}(\partial_1^T)$ (the labeling with pink ovals). Here, we see such a decomposition.

A linear boundary map $\partial_k : C_k \to C_{k-1}$ can be defined based on a global permutation of the vertices. The columns and rows of ∂_k correspond to k-simplices and $(k-1)$-simplices, respectively. Given a k-simplex $\sigma = [v_0, v_1, \ldots, v_k]$, the column of ∂_k contains $k+1$ non-zero entries. The i^{th} of these corresponds to the face $[v_0, \ldots, \hat{v}_i, \ldots, v_k]$ obtained by removing the v_i from σ. In particular, we write:

$$\partial_k([v_0, \ldots, v_k]) = \sum_{i=0}^{k} (-1)^i [v_0, \ldots, \hat{v}_i, \ldots, v_k]. \tag{2.3}$$

See Figure 5b for an example. When $k = 1$, each row of the corresponding matrix represents a vertex and each column represents an edge. There are exactly two nonzero entries in each column, since an edge is incident to exactly two vertices. Applying the boundary operator twice results in the trivial operator:

$$\partial_{k-1} \circ \partial_k(x_k) = 0; \tag{2.4}$$

that is, zero is obtained if the boundary operator is applied twice to a k-simplex x_k. The images of ∂_k and ∂_k^T have special names; we call them the *boundary cycles* and *cobounding chains*, respectively. Furthermore, in our setting, the kernel of ∂_k is called the *cycle* group.

2.4 Combinatorial Laplacian. The k-Laplacian of a simplicial complex $\Delta_k\colon C_k \to C_k$ is defined as:

$$\Delta_k = \partial_{k+1}\partial_{k+1}^T + \partial_k^T \partial_k. \tag{2.5}$$

As discussed in the introduction, the special case of $\Delta_0 = \partial_1 \partial_1^T$ is commonly referred to as the graph Laplacian. This paper focuses on Δ_1, which has two parts by (2.5): $\Delta_1^{\uparrow} = \partial_2 \partial_2^T$ and $\Delta_1^{\downarrow} = \partial_1^T \partial_1$, which we refer to as the *up* and *down* operators, respectively.

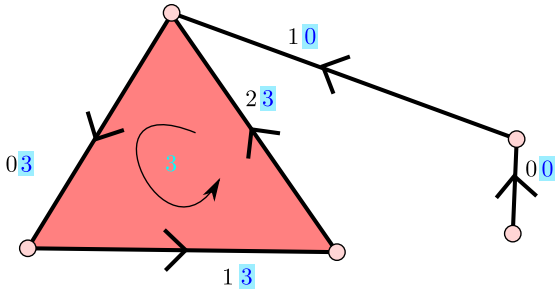


Figure 4: A one-chain (a flow) x_1 is labeled in black, its coboundary (the flux) $\partial_2^T x_1$ is labeled inside the triangle, and, finally, the boundary of $\partial_2^T x_1$ gives the residual flow $\Delta_2^{\uparrow} x_1$ labeled in blue.

The operators ∂_k and Δ_k have a physical interpretation as electric flow if k is small. For $k = 0$ (the graph Laplacian), consider the system of equations $\Delta_0 x_0 = b_0$. If x_0 is interpreted as voltages on the vertices, then $\partial_0^T(x_0)$ is the current on the edges induced by the voltages, and $\Delta_0 x_0$ is the residual voltage on the vertices. When we solve the system of equations, b_k is known and we solve for x_k. For this reason, we refer to the k-chain b_k as demands and the k-chain x_k as potentials.

There are efficient algorithms to solve linear systems envolving graph Laplacians (Δ_0). The following Lemma describe a slightly generalized class of solvers.

Lemma 2.3 (Approximate Solver). *[Spi04, KMP10, KMP11, KOSZ13, LS13] Given a n-by-n symmetrically diagonally dominant linear system A with m non-zero entries and an error parameter ε, there is a linear operator* SolveZeroLap(A, ε) *such that for any vector b,* SolveZeroLap(A, ε) *can be evaluated in time $\widetilde{O}(m \log(1/\varepsilon))$ and:*

$$(1-\varepsilon)A^{+} \preceq \textsc{SolveZeroLap}(A, \varepsilon) \preceq A^{+}.$$

More specifically, [KMP11] runs in time $O(m \log m \log\log m \log(1/\varepsilon))$ in the exact arithmetic model.

The solver algorithm can be viewed as producing in $O(m \log m \log\log m \log(1/\varepsilon))$ time a sequence of $O(m \log m \log\log m \log(1/\varepsilon))$ addition and mutiplication operations without branches. This sequence of operations gives the procedure SolveZeroLap(A, ε), which can also be viewed as an arithmetic circuit of similar size. In the absence of round-off errors, running this procedure leads to the bound in the lemma above. The time to generate SolveZeroLap is dominated by that of finding a good spanning tree [AN12], and is a lower order term that we can omit.

The exact runtime of the solver depends on the model of round-off errors. The result established in [KMP11] assumes exact arithmetic. Recently, the numerical stabilities of solver procedures were analyzed in settings close to fixed-point arithmetic [KOSZ13, LS13, Pen13]. This is not considered here as this paper also works in the exact arithmetic model.

2.5 Collapsibility. Collapsibility was first introduced by Whitehead [Whi39]. Later, Cohen [Coh73] build the concept of simple homotopy equivalence as a refinement of homotopy equivalence based on the collapsing and expansion operations.

Let K be a simplicial complex. A k-simplex $\sigma \in K$ is *free* if it is properly contained in exactly one simplex τ. In this case, an *elementary collapse* of K at σ gives the simplicial complex $K' = K - \tau - \sigma$. An elementary collapse at σ is a $\dim(\sigma)$-collapse.

A simplicial complex K collapses to a simplicial complex L if there is a sequence of simplicial complexes $K = K_0 \supset K_2 \supset \cdots \supset K_k = L$ such that K_i is obtained from K_{i-1} by a collapse at σ_i. In this case, we call $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ a (K, L) *collapsing sequence* and write $L = K \searrow \Sigma$. A simplicial complex is called *collapsible* if it collapses to a single vertex. The following theorem of Chillingworth [Chi67, Chi80] relates collapsible and convex simplicial complexes.

Theorem 2.4 (Collapsing the Three-Ball). *If K is a convex simplicial complex that triangulates the three-ball, then K is collapsible. Furthermore, a collapsing sequence of K can be computed in linear time.*

A pair (σ_p, σ_q), with $1 \le p < q \le k$, is *flipped* if $\dim(\sigma_p) < \dim(\sigma_q)$. A (K, L) collapsing sequence is *monotone* if it has no flipped pair. In this case, we say that K monotonically collapses to L. The following lemma allows us to assume that Σ is monotone.

Lemma 2.5 (Monotone Collapse). *If K collapses to L, then K monotonically collapses to L. Furthermore, a monotone collapsing sequence can be computed from any collapsing sequence in linear time.*

Proof: Let $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ be a (K, L) collapsing sequence. We proceed by induction.

Suppose there is exactly one flipped pair (σ_i, σ_{i+1}). Necessarily, this flipped pair appears consecutively in Σ. Let $K' = K \searrow (\sigma_1, \ldots, \sigma_{i-1})$. Then, we know that σ_i is free in K'. Also, σ_{i+1} is free in $K' \searrow \sigma_i$. Since $\dim(\sigma_i) < \dim(\sigma_{i+1})$, we have that σ_{i+1} is also free in K'. It follows that $K \searrow (\sigma_1, \ldots, \sigma_{i+1}, \sigma_i) = K \searrow (\sigma_1, \ldots, \sigma_i, \sigma_{i+1}) = K'$ Thus, the sequence $\Sigma_1 = (\sigma_1, \sigma_2, \ldots, \sigma_{i+1}, \sigma_i, \ldots, \sigma_k)$ is a proper (K, L)-collapsing sequence.

We now assume that any (K, L)-collapsing sequence with less than n flipped pairs, can be modified to a monotone (K, L)-collapsing sequence by transposing exactly n pairs. If there exists n flipped pairs, let (σ_i, σ_{i+1}) be the first flipped pair, and find a new sequence Σ_1 as before. Since Σ_1 has exactly $n - 1$ flipped pairs, we can obtain a monotone collapsing from Σ sequence by transposing $n - 1$ pairs by our induction hypothesis. □

3 Decomposition via Projection

Recall from (2.4), which gives us that applying the boundary operator twice is the trivial operator. As a consequence, the images $\mathbf{im}(\partial_{k+1})$ and $\mathbf{im}(\partial_k^T)$ are orthogonal. In the present paper, we assume that K is homeomorphic to a three-ball; thus, we can assume $\mathbf{im}(\partial_{k+1}) = \mathbf{null}(\partial_k)$ and by the Fundamental Theorem of Linear Algebra, $\mathbf{im}(\partial_{k+1})$ and $\mathbf{im}(\partial_k^T)$ span the space of k-chains; that is:

Theorem 3.1 (Decomposing the Laplacian). *Any k-chain x_k, with $0 \le k \le 2$, of a simplicial complex with trivial k-dimensional homology (for example, the triangulation of a three-ball) can be uniquely decomposed into two parts: $x_k = x_k^{\uparrow} + x_k^{\downarrow}$, where $x_k^{\uparrow} \in \boldsymbol{im}(\partial_{k+1})$ and $x_k^{\downarrow} \in \boldsymbol{im}(\partial_k^T)$.*

Let $\Pi_k^{\uparrow}$ and $\Pi_k^{\downarrow}$ be the corresponding projection operators so that for any k-chain x_k, we have $x_k^{\uparrow} = \Pi_k^{\uparrow} x_k$ and $x_k^{\downarrow} = \Pi_k^{\downarrow} x_k$, where $x_k^{\uparrow}$ and $x_k^{\downarrow}$ are given as in Theorem 3.1. Lemma 2.1 implies $\Pi_k^{\uparrow} = \partial_{k+1}(\partial_{k+1}^T \partial_{k+1})^+ \partial_{k+1}^T$ and $\Pi_k^{\downarrow} = \partial_k^T (\partial_k \partial_k^T)^+ \partial_k$.

In particular, Theorem 3.1 leads to the discrete Helmholtz decomposition, which decomposes a one-chain x_1 into a cobounding chain (potential flow)

$$x_1^{\downarrow} = \Pi_1^{\downarrow} x_1 = \partial_1^T x_0$$

and a bounding cycle (circulation)

$$x_1^{\uparrow} = \Pi_1^{\uparrow} x_1 = \partial_2 x_2;$$

see Figure 3 for an example. A further generalization of this decomposition, where a third null space exists, is known as the Hodge decomposition [Hod41].

Here, we show that the operators $\Pi_1^{\uparrow}$ and $\Pi_1^{\downarrow}$ can be approximated in nearly-linear time. Throughout the rest of the paper, we denote the the approximated operators up to ε accuracy by $\widetilde{\Pi}_1^{\uparrow}(\varepsilon)$ and $\widetilde{\Pi}_1^{\downarrow}(\varepsilon)$.

Lemma 3.2 (Projections of One-Chains). *Give a graph Laplacian with m edges and any $\varepsilon > 0$, there exist operators $\widetilde{\Pi}_1^{\uparrow}(\varepsilon)$ and $\widetilde{\Pi}_1^{\downarrow}(\varepsilon)$, each computable in $O(m \log m \log\log m \log m/\varepsilon)$ time such that:*

$$(1-\varepsilon)\Pi_1^{\uparrow} \preceq \widetilde{\Pi}_1^{\uparrow}(\varepsilon) \preceq \Pi_1^{\uparrow}, \tag{3.6}$$

$$(1-\varepsilon)\Pi_1^{\downarrow} \preceq \widetilde{\Pi}_1^{\downarrow}(\varepsilon) \preceq \Pi_1^{\downarrow}. \tag{3.7}$$

Proof: Define

$$\widetilde{\Pi}_1^{\downarrow}(\varepsilon) = \partial_1^T \textsc{SolveZeroLap}(\partial_1 \partial_1^T, \varepsilon)\partial_1,$$

where SolveZeroLap is the solver given in Lemma 2.3. Hence, also by Lemma 2.3, we have:

$$\begin{aligned}(1-\varepsilon)(\partial_1\partial_1^T)^+ &\preceq \textsc{SolveZeroLap}(\partial_1\partial_1^T, \varepsilon) \\ &\preceq (\partial_1\partial_1^T)^+.\end{aligned}$$

Applying Lemma 2.2 allows us to compose this bound with the ∂_1^T and ∂_1 on the left and right, respectively, obtaining:

$$\begin{aligned}(1-\varepsilon)\,\partial_1^T(\partial_1\partial_1^T)^+\partial_1 \preceq \widetilde{\Pi}_1^{\downarrow}(\varepsilon) \preceq \partial_1^T(\partial_1\partial_1^T)^+\partial_1 \\ (1-\varepsilon)\,\Pi_1^{\downarrow} \preceq \widetilde{\Pi}_1^{\downarrow}(\varepsilon) \preceq \Pi_1^{\downarrow}.\end{aligned}$$

Proving (3.7) is more intricate. For a spanning tree $\mathcal{T}$ of the one-skeleton of K, we define a non-orthogonal projection operator $\Pi_{\mathcal{T}}$ that takes a flow and returns the unique flow using only edges from $\mathcal{T}$ satisfying the same demands. We note here that both $\Pi_{\mathcal{T}}$ and $\Pi_{\mathcal{T}}^T$ can be computed in linear time. Since $\Pi_1^{\uparrow}$ returns a cycle (circulation), we have $\Pi_{\mathcal{T}}\Pi_1^{\uparrow} = 0$, which implies:

$$(I - \Pi_{\mathcal{T}})\,\Pi_1^{\uparrow}\,(I - \Pi_{\mathcal{T}})^T = \Pi_1^{\uparrow}.$$

We define the approximate projection operator, based on this equivalent representation of $\Pi_1^{\uparrow}$ and the fact that $\Pi_1^{\uparrow} = I - \Pi_1^{\downarrow}$:

$$\widetilde{\Pi}_1^{\uparrow}(\varepsilon) = (1-\varepsilon)\,(I - \Pi_{\mathcal{T}})\left(I - \widetilde{\Pi}_1^{\downarrow}(\varepsilon/m^2)\right)(I - \Pi_{\mathcal{T}})^T.$$

Note that a smaller value of ε could add a logarithmic factor to the runtime; however, as we already need to set ε to $\frac{1}{\Omega(m^c)}$, it only multiplies the total runtime by a constant. Manipulating the bounds in (3.6) proven above gives:

$$\begin{aligned}(1-\varepsilon/m^2)\Pi_1^{\downarrow} \preceq \quad \widetilde{\Pi}_1^{\downarrow}(\varepsilon/m^2) \quad &\preceq \Pi_1^{\downarrow} \\ \Pi_1^{\uparrow} \preceq I - \widetilde{\Pi}_1^{\downarrow}(\varepsilon/m^2) \ &\preceq \Pi_1^{\uparrow} + \varepsilon/m^2 I.\end{aligned}$$

Then applying Lemma 2.2 to all inequalities in this bound gives:

$$\begin{aligned}&(I - \Pi_{\mathcal{T}})\Pi_1^{\uparrow}(I - \Pi_{\mathcal{T}})^T \\ &\preceq\ (I - \Pi_{\mathcal{T}})\left(I - \widetilde{\Pi}_1^{\downarrow}(\varepsilon/m^2)\right)(I - \Pi_{\mathcal{T}})^T \\ &\preceq\ (I - \Pi_{\mathcal{T}})\Pi_1^{\uparrow}(I - \Pi_{\mathcal{T}})^T + \varepsilon/m^2(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T\end{aligned}$$

Multiplying both sides by $(1-\varepsilon)$ and applying the definition of $\widetilde{\Pi}_1^{\uparrow}$, we obtain:

$$\begin{aligned}&(1-\varepsilon)\Pi_1^{\uparrow} \preceq \widetilde{\Pi}_1^{\uparrow}(\varepsilon) \\ &\quad\preceq\ (1-\varepsilon)(\Pi_1^{\uparrow} + \varepsilon/m^2(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T).\end{aligned}$$

Therefore it remains to upper bound $\varepsilon/m^2(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T$ spectrally by $\Pi_1^{\uparrow}$. Note $\Pi_{\mathcal{T}}$ maps each off-tree edge to all tree edges generated by its fundamental cycle, and diagonal entries of $\Pi_{\mathcal{T}}$ are non-zero when the edge is on the tree. Therefore, each matrix element of $I - \Pi_{\mathcal{T}}$ has absolute value at most one. This allows us to bound the spectral norm of $(I - \Pi_{\mathcal{T}})^T(I - \Pi_{\mathcal{T}})$:

$$(I - \Pi_{\mathcal{T}})^T(I - \Pi_{\mathcal{T}}) \preceq m^2 I.$$

Applying Lemma 2.2 again, with $\Pi_1^{\uparrow}$ as the outer matrix, gives:

$$\begin{aligned}\Pi_1^{\uparrow}(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T\Pi_1^{\uparrow} &\preceq \Pi_1^{\uparrow}\left(m^2 I\right)\Pi_1^{\uparrow} \\ &\preceq m^2\Pi_1^{\uparrow}.\end{aligned}$$

The last relation follows from I commuting with $\Pi_1^{\uparrow}$ and $\Pi_1^{\uparrow}$ being a projection matrix.

Since $I - \Pi_{\mathcal{T}}$ returns a cycle (circulation), we have:

$$\Pi_1^{\uparrow}(I - \Pi_{\mathcal{T}}) = I - \Pi_{\mathcal{T}}.$$

We multiply each side by its transpose to obtain:

$$\Pi_1^{\uparrow}(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T\Pi_1^{\uparrow} = (I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T.$$

Hence we have $(I - \Pi_{\mathcal{T}})(I - \Pi_{\mathcal{T}})^T \preceq m^2\Pi_1^{\uparrow}$. Putting everything together gives the desired bounds on $\widetilde{\Pi}_1^{\uparrow}$:

$$\begin{aligned}(1-\varepsilon)\Pi_1^{\uparrow} \preceq \quad \widetilde{\Pi}_1^{\uparrow}(\varepsilon) &\preceq \quad (1-\varepsilon)(\Pi_1^{\uparrow} + \varepsilon/m^2 \cdot m^2\Pi_1^{\uparrow}) \\ &\preceq \quad (1-\varepsilon)(1+\varepsilon)\Pi_1^{\uparrow} \\ &\preceq \quad \Pi_1^{\uparrow}.\end{aligned}$$

□

As a result of the above theorem, $\Pi_1^{\uparrow}$ and $\Pi_1^{\downarrow}$ are approximations of $\Pi_1^{\uparrow}$ and $\Pi_1^{\downarrow}$, respectively. We can obtain similar approximating operators for $\Pi_2^{\uparrow}$ and $\Pi_2^{\downarrow}$ by observing a duality in the complex. The dual graph that we use here is the one-skeleton of the dual cell structure [Hat01, Ch. 3], initially introduced by Poincaré.

Duality. Let K be a three-complex homeomorphic to a three manifold. We define the dual graph of K, denoted K^*, as follows: for each tetrahedron $t \in K$, we define a dual vertex $t^* \in K^*$. There is an edge between two vertices $t_1^*, t_2^* \in K^*$ if the corresponding tetrahedra t_1 and t_2 share a triangle in K. We extend this definition to a three manifold with boundry by adding a special vertex σ^* in K^* called the *infinite vertex* and connecting σ^* to every $t^* \in K^*$ that is dual to a tetrahedron containing a boundary triangle; a *boundary triangle* is a triangle which is incident on at most one tetrahedra. The vertices and edges of K^* correspond to the tetrahedra and triangles of K, and σ^* corresponds to $\mathbb{S}^3 \backslash K$. We note here that this is the same duality that exists between Delaunay triangulations and Voronoi Diagrams.

The duality defined above and the fact that K represents a three manifold imply the following correspondence. Three-chains of K correspond to zero-chains (vertex potentials) in K^*, where zero is assigned to σ. Two-chains of K correspond to one-chains (flows) in K^*. Thus, we obtain:

Lemma 3.3 (Projection of Two-Chains). *Let K be a triangulation of a three-ball and let $\Pi_2^{\uparrow}$ and $\Pi_2^{\downarrow}$ be as defined above. Then, for any $\varepsilon > 0$, the operators $\widetilde{\Pi}_2^{\uparrow}(\varepsilon)$ and $\widetilde{\Pi}_2^{\downarrow}(\varepsilon)$ can be computed in $\tilde{O}(m \log m \log\log m \log m/\varepsilon)$ time such that*

$$(1-\varepsilon)\Pi_2^{\uparrow} \preceq \widetilde{\Pi}_2^{\uparrow}(\varepsilon) \preceq \Pi_2^{\uparrow},$$
$$(1-\varepsilon)\Pi_2^{\downarrow} \preceq \widetilde{\Pi}_2^{\downarrow}(\varepsilon) \preceq \Pi_2^{\downarrow}.$$

4 Algorithm for Solving the 1-Laplacian

In this section, we sketch our algorithm to solve the linear system $\Delta_1 x_1 = b_1$ for a simplicial complex K of a collapsible three-ball with a known collapsing sequence.

4.1 Flow Decomposition.

Recall from the discussion following (2.5), that the Laplacian can be decomposed into two parts: $\Delta_1 = \Delta_1^{\uparrow} + \Delta_1^{\downarrow}$. We are interested in solving the following problem: given a one-chain b_1, find another one-chain x_1 such that $\Delta_1 x_1 = b_1$. The following lemma enables us to decompose the equation into two different equations through a set of projections; Lemma 3.2 provides efficiently computable operators for such projections.

Lemma 4.1 (Splitting the Flow). *Let K be a triangulation of a three-ball and let b_1 be a one-chain. Consider the systems of equations $\Delta_1 x_1 = b_1$, $\Delta_1^{\uparrow} y_1 = b_1^{\uparrow}$, and $\Delta_1^{\downarrow} z_1 = b_1^{\downarrow}$. Then, the following holds: $x_1 = y_1^{\uparrow} + z_1^{\downarrow}$.*

Proof: Recall that $\mathbf{im}(\Delta_1^{\uparrow})$ and $\mathbf{im}(\Delta_1^{\downarrow})$ orthogonaly decompose $\mathbf{im}(\Delta)$. Thus, we have $y_1^{\uparrow}, z_1^{\uparrow} \in \mathbf{null}(\Delta_1^{\downarrow})$ and $y_1^{\downarrow}, z_1^{\downarrow} \in \mathbf{null}(\Delta_1^{\uparrow})$, which gives us:

$$\begin{aligned}
\Delta_1(y_1^{\uparrow} + z_1^{\downarrow}) &= (\Delta_1^{\uparrow} + \Delta_1^{\downarrow})(y_1^{\uparrow} + z_1^{\downarrow}) \\
&= \left(\Delta_1^{\uparrow} y_1^{\uparrow} + \Delta_1^{\uparrow} z_1^{\downarrow}\right) + \left(\Delta_1^{\downarrow} y_1^{\uparrow} + \Delta_1^{\downarrow} z_1^{\downarrow}\right) \\
&= \Delta_1^{\uparrow} y_1 + \Delta_1^{\downarrow} z_1 \\
&= b_1^{\uparrow} + b_1^{\downarrow}.
\end{aligned}$$

The third equality holds since $\Delta_1^{\uparrow} y_1^{\downarrow}$, $\Delta_1^{\uparrow} z_1^{\downarrow}$, $\Delta_1^{\downarrow} y_1^{\uparrow}$ and $\Delta_1^{\downarrow} z_1^{\uparrow}$ are trivial. □

4.2 Down Operator.

Now that we have divided the problem into two parts, $\Delta_1^{\uparrow} y_1 = b_1^{\uparrow}$ and $\Delta_1^{\downarrow} z_1 = b_1^{\downarrow}$, we begin with $\Delta_1^{\downarrow}$. As $\Delta_1^{\downarrow}$ is defined as $\partial_1^T \partial_1$, it is helpful to remind the reader the combinatorial interpretation of ∂_1. Given one-chain (a flow) f_1, $\partial_1 f_1$ is the residue of the flow at all the vertices. On the flip side, given zero-chain (vertex potentials) p_0, $\partial_1^T p_0$ is a potential flow obtained by setting the flow along each edge to the potential difference between its two endpoints. This interpretation also plays a crucial role in the electrical flow based Laplacian solver [KOSZ13]. However, as we will see, both recovering potentials from a flow, and finding a flow meeting a set of residuals are fairly simple combinatorial operations. Solving $\Delta_1^{\downarrow} x_1 = b_1^{\downarrow}$ is a simpler operation than solving a graph Laplacian. The following lemma provides a linear operator $(\Delta_1^{\downarrow})^+$ to solve $\Delta_1^{\downarrow} x_1 = b_1^{\downarrow}$; we highlight the assumption that $b_1^{\downarrow} \in \mathbf{im}(\Delta_1^{\downarrow})$,

Lemma 4.2 (Down Solver). *Let K be any graph and suppose $\Delta_1^{\downarrow} x_1 = b_1^{\downarrow}$ has at least one solution. Then, a linear operator $(\Delta_1^{\downarrow})^+$ exists such that $\Delta_1^{\downarrow}(\Delta_1^{\downarrow})^+ b_1^{\downarrow} = b_1^{\downarrow}$. Furthermore, $(\Delta_1^{\downarrow})^+$ and $(\Delta_1^{\downarrow})^+ b_1^{\downarrow}$ can be computed in linear time.*

Proof: Recall $\Delta_1^{\downarrow} = \partial_1^T \partial_1$. We first find $z_0 = \sum_i c_i v_i$ such that $\partial_1^T z_0 = b_1^{\downarrow}$, then we will find z_1 that satisfies $\partial_1 z_1 = z_0$.

Without loss of generality, we assume that K is connected; otherwise, we can solve the problem for each connected component separately. Pick an

arbitrary spanning tree $\mathcal{T}$ of edges in K. The one-chain z_0 can be written as the weighted sum of vertices: $\sum_{i=0}^{n} c_i v_i$, where $c_i \in \mathbb{R}$ and n is the number of vertices in K. or any edge (v_j, v_k) of $\mathcal{T}$, knowing c_j implies a unique value for c_k. Letting v_0 be a root of $\mathcal{T}$, we set $c_0 = 0$. Then, we can uniquely determine all values c_i by traversing the edges of $\mathcal{T}$.

By Theorem 3.1, we know $z_0^{\uparrow} = z_0 - z_0^{\downarrow}$. Since K is connected, we have $\partial_0 = \mathbb{1}^T$. Consequently, $\mathbb{1}^T z_0^{\uparrow} = 0$ and $z_0^{\downarrow} = c\mathbb{1}$ for some constant $c \in \mathbb{R}$. Overall,

$$0 = \mathbb{1}^T z_0^{\uparrow} = \mathbb{1}^T(z_0 - c\mathbb{1}) = \mathbb{1}^T z_0 - c\mathbb{1}^T\mathbb{1}$$

Thus, we can compute $z_0^{\downarrow}$ (and $z_0^{\uparrow}$) by computing c by finding the unique c such that $\mathbb{1}^T z_0 - c\mathbb{1}^T\mathbb{1} = 0$.

It remains to find z_1 such that $\partial_1 z_1 = z_0^{\uparrow}$. This is equivalent to finding a flow that meets the set of demands given by $z_0^{\uparrow}$ at each vertex. Again, we pick a spanning tree $\mathcal{T}$ of the one-skeleton of K. Knowing the demand on any leaf of $\mathcal{T}$ uniquely determines the value of z_1 on its only incident edge. Hence, we can compute z_1 recursively in linear time.

It is straight forward to put the used operations together to get the linear operator $(\Delta_1^{\downarrow})^+$. In fact, the whole process can be seen as collapsing forward (and expanding backward) the spanning tree $\mathcal{T}$. The process of finding a sequence of Gaussian elimination steps that corresponds to this collapse is very similar to the argument presented in Section 5. □

4.3 Up Operator. Our algorithm for solving $\Delta_1^{\uparrow} y_1 = b_1^{\uparrow}$ proceeds similarly. Ideally, we want a two-chain b_2 such that a solution y_1 exists satisfying $\partial_2 b_2 = b_1^{\uparrow}$ and $\partial_2^T y_1 = b_2$. Having such a b_2, we can solve the equation $\partial_2^T y_1 = b_2$ to obtain y_1. Given any two-chain y_2 such that $\partial_2 y_2 = b_1^{\uparrow}$, we observe in Lemma 4.3 that $b_2 = y_2^{\uparrow}$ has the desired properties.

Surprisingly (compared to the lower dimensional case), solving $\partial_2 y_2 = b_1^{\uparrow}$ to find any solution y_2 is not straight forward. In Section 5 we describe an algorithm for the special case where K is collapsible and the collapsing sequence is known. Lets call the operator to solve the equation under this condition ∂_2^+ (see Theorem 5.2), and recall the projection operator $\Pi_2^{\downarrow}$ of Theorem 5.2. The following lemma describes a linear operator to solve $\Delta_1^{\uparrow} y_1 = b_1^{\uparrow}$.

Lemma 4.3 (Up Solver). *Let K be a triangulation of a three-ball and let $(\Delta_1^{\uparrow})^+ = (\partial_2^+)^T \Pi_2^{\downarrow} \partial_2^+$. If $\Delta_1^{\uparrow} x_1 = b_1^{\uparrow}$ has at least one solution, then $(\Delta_1^{\uparrow})^+ b_1^{\uparrow}$ is a solution.*

Proof: By Theorem 5.2, the two-chain $y_2 = \partial_2^+ b_1^{\uparrow}$ is a solution to $\partial_2 y_2 = b_1^{\uparrow}$. Consider the decomposition $y_2 = y_2^{\uparrow} + y_2^{\downarrow}$. We have:

$$b_1^{\uparrow} = \partial_2 y_2 = \partial_2(y_2^{\uparrow} + y_2^{\downarrow}) = \partial_2 y_2^{\downarrow}.$$

The last equality follows from the facts $y_2^{\uparrow} \in \mathbf{im}(\partial_3)$ and $\partial_2\partial_3 = 0$.

Since $y_2^{\downarrow} \in \mathbf{im}(\partial_2^T)$, there exists a y_1 such that $\partial_2^T y_1 = y_2^{\downarrow}$. Applying Theorem 5.2 again, we obtain $y_1 = (\partial_2^T)^+ y_2^{\downarrow}$. Since $y_2^{\downarrow} = \Pi_2^{\downarrow} y_2$ and $y_2 = \partial_2^+ b_1$, the statement of the lemma follows from Theorem 3.1. □

The first and last steps are solving $\partial_2 y_2 = b_1^{\uparrow}$ and $\partial_2^T y_1 = y_2^{\downarrow}$ (equivalently, computing ∂_2^+). Recall that to solve a similar set of equations in a lower dimension, we exploited the structure of a spanning tree; see the proof of Lemma 4.2. Spanning trees are especially nice because they form a basis of the column space of the boundary matrix, and, more importantly, they are collapsible. On the other hand, it is not necessarily true that a set of independent faces in higher dimensions is collapsible. Our algorithm, described in Section 5, assumes that a collapsing sequence of the simplicial complex is known in order to compute a cheap sequence of Gaussian eliminations.

Lemma 3.3 and Theorem 5.2 enable us to compute (within an approximation factor of ε) the parts of $(\Delta_1^{\uparrow})^+$ as in Lemma 4.3. Then, the following lemma is immediate using Lemma 2.2.

Lemma 4.4 (Pseudoinverse of Up Operator). *Let K be a collapsible simplicial complex that triangulates a three-ball with m simplices and a known collapsing sequence. For any $0 < \varepsilon < 1$, the operator* SolveUpLap$((\Delta_1^{\uparrow})^+, K, \varepsilon)$ *with the following property can be computed in $\widetilde{O}(m \log 1/\varepsilon))$ time.*

$$(1-\varepsilon)(\Delta_1^{\uparrow})^+ \preceq \textsc{SolveUpLap}((\Delta_1^{\uparrow})^+, K, \varepsilon) \preceq (\Delta_1^{\uparrow})^+.$$

4.4 Summing Up. Lemma 4.1, Lemma 4.2 and Lemma 4.3 imply that $\Delta_1^+ = \Pi_1^{\downarrow}(\Delta_1^{\downarrow})^+\Pi_1^{\downarrow} + \Pi_1^{\uparrow}(\Delta_1^{\uparrow})^+\Pi_1^{\uparrow}$ is an operator to solve one-Laplacians. The following lemma finds application in approximating this operator:

Lemma 4.5. *Let $A : C_1 \to C_1$ be a symmetric linear operator, Π be an orthogonal projection and $\widetilde{\Pi}$ be a linear operator that satisfies $-\varepsilon\Pi \preceq \Pi - \widetilde{\Pi} \preceq \varepsilon\Pi$. Then, we have:*

$$(1 - 3\varepsilon\kappa^{\Pi}(A))\Pi A\Pi \preceq \widetilde{\Pi}A\widetilde{\Pi} \preceq (1 + 3\varepsilon\kappa^{\Pi}(A))\Pi A\Pi$$

where κ_A^{Π} is the condition number of A restricted to the subspace of the image of Π.

Proof: The proof first establishes a matrix norm bound. This follows from the triangle inequality and

the fact that $\|\Pi\|_2 \leq 1$ (as Π is a projection matrix). In particular, we have:

$$\begin{aligned}
&\left\|\widetilde{\Pi}A\widetilde{\Pi} - \Pi A\Pi\right\|_2 \\
&= \left\|\widetilde{\Pi}A\widetilde{\Pi} - \widetilde{\Pi}A\Pi + \widetilde{\Pi}A\Pi - \Pi A\Pi\right\|_2 \\
&\leq \left\|\widetilde{\Pi}A\widetilde{\Pi} - \widetilde{\Pi}A\Pi\right\|_2 + \left\|\widetilde{\Pi}A\Pi - \Pi A\Pi\right\|_2 \\
&\leq \left\|\widetilde{\Pi}\right\|_2 \|A\|_2 \left\|\widetilde{\Pi} - \Pi\right\|_2 + \left\|\widetilde{\Pi} - \Pi\right\|_2 \|A\|_2 \|\Pi\|_2 \\
&\leq 3\varepsilon\lambda^{\Pi}_{\max}(A)
\end{aligned}$$

The spectral bound property of $\widetilde{\Pi}$ implies that $\widetilde{\Pi}$ is 0 for vectors in the nullspace of Π, and always outputs vectors in the image of Π. The same then must hold for $\widetilde{\Pi}A\widetilde{\Pi}$. This, combined with the matrix norm just proved, means that

$$-3\varepsilon\lambda^{\Pi}_{\max}(A)\Pi \preceq \widetilde{\Pi}A\widetilde{\Pi} - \Pi A\Pi \preceq 3\varepsilon\lambda^{\Pi}_{\max}(A)\Pi$$

But

$$\begin{aligned}
3\varepsilon\lambda^{\Pi}_{\max}(A)\Pi &= 3\varepsilon\lambda^{\Pi}_{\max}(A)\Pi I\Pi \\
&\preceq 3\varepsilon\frac{\lambda^{\Pi}_{\max}(A)}{\lambda^{\Pi}_{\min}(A)}\Pi A\Pi \\
&= 3\varepsilon\kappa^{\Pi}(A)\Pi A\Pi
\end{aligned}$$

This gives

$$-3\varepsilon\kappa^{\Pi}(A)\Pi A\Pi \preceq \widetilde{\Pi}A\widetilde{\Pi} - \Pi A\Pi \preceq 3\varepsilon\kappa^{\Pi}(A)\Pi A\Pi$$

$$(1 - 3\varepsilon\kappa^{\Pi}(A))\Pi A\Pi \preceq \widetilde{\Pi}A\widetilde{\Pi} \preceq (1 + 3\varepsilon\kappa^{\Pi}(A))\Pi A\Pi$$

as desired. □

Now, we are ready to state and prove the main theorem of this section.

Theorem 4.6 (Collapsible Complex Solver). *Let K be a collapsible simplicial complex that triangulates a three-ball with m simplices and a known collapsing sequence. For any $0 < \varepsilon < 1$, the linear operator* SolveOneLap$(\Delta_1, K, \varepsilon)$ *and the vector* SolveOneLap$(\Delta_1, K, \varepsilon) \cdot b_1$ *for any one-chain b_1 can be computed in $O(m \log m \log\log m \log \kappa/\varepsilon))$ time, where κ is the maximum of the condition numbers of the up and down Laplacians, such that*

$$(1 - \varepsilon)\Delta_1^+ \preceq \textsc{SolveOneLap}(\Delta_1, K, \varepsilon) \preceq \Delta_1^+$$

Proof: In this proof we write the projection operators of the form $\widetilde{\Pi}_1^{\downarrow}(\varepsilon')$ more concisely as $\widetilde{\Pi}_1^{\downarrow}$ by dropping the parameter ε'.

Consider the operator

$$\begin{aligned}
&\textsc{SolveOneLap}(\Delta_1, K, \varepsilon) = \\
&\quad (1 - \varepsilon/2)\widetilde{\Pi}_1^{\downarrow}(\varepsilon/6\kappa)(\Delta_1^{\downarrow})^+\widetilde{\Pi}_1^{\downarrow}(\varepsilon/6\kappa) + \\
&\quad (1 - \varepsilon/3)\widetilde{\Pi}_1^{\uparrow}(\varepsilon/9\kappa)\textsc{SolveUpLap}(\Delta_1^{\uparrow}, \varepsilon/3)\widetilde{\Pi}_1^{\uparrow}(\varepsilon/9\kappa)
\end{aligned}$$

The error between this operator and the exact inverse can be measured separately for each summand. For the first one Lemma 3.2 and Lemma 4.5 imply:

$$(1 - \varepsilon/2)\Pi_1^{\downarrow}(\Delta_1^{\downarrow})^+\Pi_1^{\downarrow} \preceq \widetilde{\Pi}_1^{\downarrow}(\Delta_1^{\downarrow})^+\widetilde{\Pi}_1^{\downarrow} \preceq (1 + \varepsilon/2)\Pi_1^{\downarrow}(\Delta_1^{\downarrow})^+\Pi_1^{\downarrow}$$

$$(1 - \varepsilon)\Pi_1^{\downarrow}(\Delta_1^{\downarrow})^+\Pi_1^{\downarrow} \preceq (1 - \varepsilon/2)\widetilde{\Pi}_1^{\downarrow}(\Delta_1^{\downarrow})^+\widetilde{\Pi}_1^{\downarrow} \preceq \Pi_1^{\downarrow}(\Delta_1^{\downarrow})^+\Pi_1^{\downarrow}$$

The up Laplacian can be bounded similarly, with additional error from the difference between SolveUpLap$(\Delta_1^{\uparrow}, \varepsilon)$ and $(\Delta_1^{\uparrow})^+$.

$$\begin{aligned}
&(1 - \varepsilon/3)\widetilde{\Pi}_1^{\uparrow}(\Delta_1^{\uparrow})^+\widetilde{\Pi}_1^{\uparrow} \\
&\quad \preceq \widetilde{\Pi}_1^{\uparrow}\textsc{SolveUpLap}(\Delta_1^{\uparrow}, \varepsilon/3)\widetilde{\Pi}_1^{\uparrow} \\
&\quad \preceq \widetilde{\Pi}_1^{\uparrow}(\Delta_1^{\uparrow})^+\widetilde{\Pi}_1^{\uparrow} \\
&(1 - 2\varepsilon/3)\Pi_1^{\uparrow}(\Delta_1^{\uparrow})^+\Pi_1^{\uparrow} \\
&\quad \preceq \widetilde{\Pi}_1^{\uparrow}\textsc{SolveUpLap}(\Delta_1^{\uparrow}, \varepsilon/3)\widetilde{\Pi}_1^{\uparrow} \\
&\quad \preceq (1 + \varepsilon/3)\Pi_1^{\uparrow}(\Delta_1^{\uparrow})^+\Pi_1^{\uparrow} \\
&(1 - \varepsilon)\Pi_1^{\uparrow}(\Delta_1^{\uparrow})^+\Pi_1^{\uparrow} \\
&\quad \preceq (1 - \varepsilon/3)\widetilde{\Pi}_1^{\uparrow}\textsc{SolveUpLap}(\Delta_1^{\uparrow}, \varepsilon/3)\widetilde{\Pi}_1^{\uparrow} \\
&\quad \preceq \Pi_1^{\uparrow}(\Delta_1^{\uparrow})^+\Pi_1^{\uparrow}
\end{aligned}$$

The first chain of inequalities follows from Lemma 4.4 and the second from Lemma 4.5. □

In general, finding a collapsing sequence for a simplicial complex efficiently is difficult. Recently, Tancer [Tan12] has shown that even testing whether a simplicial complex of dimension three is collapsible is NP-hard. It is not known whether the collapsibility problem is tractable for special cases of embeddable simplicial complexes in $\mathbb{R}^3$, or even for topological three-balls. However, Theorem 2.4 provides a linear time algorithm to compute a collapsing sequence of a convex simplicial complex, which in turn implies the final result of this section.

Corollary 4.7 (1-Laplacian Pseudoinverse). *Let K be a convex simplicial complex that triangulates a three-ball with m simplices. For any $0 < \varepsilon < 1$, the linear operator* SolveOneLap$(\Delta_1, K, \varepsilon)$ *(and the vector* SolveOneLap$(\Delta_1, K, \varepsilon) \cdot b_1$ *for any one-chain b_1) can be computed in $O(m \log m \log\log m\kappa/\varepsilon)$ time, where κ is the maximum of the condition numbers of the up and down Laplacians, such that*

$$(1 - \varepsilon)\Delta_1^+ \preceq \textsc{SolveOneLap}(\Delta_1, K, \varepsilon) \preceq \Delta_1^+$$

5 Collapsibility and Gaussian Elimination

In this section, we solve the linear system $\partial_2 x_2 = b_1$ for a collapsible simplicial complex K given a collapsing sequence Σ of K. The key insight is to view the Gaussian elimination operators as simplicial collapses in K.

By Lemma 2.5, we can assume without loss of generality that Σ is monotone. Therefore, Σ may be expressed as $\Sigma_2 \cdot \Sigma_1 \cdot \Sigma_0$, where Σ_i is the ordered sequence of i-collapses. We draw a parallel between the collapses of the leaf nodes in the proof of Lemma 4.2 and collapsing the triangles incident to exactly one tetrahedron. In this light, the removal of such triangles does not decrease the rank of ∂_2, and therefore does not affect the solution space. In terms of linear algebra, collapsing triangles is equivalent to setting the corresponding coordinate of x_2 to zero. The remaining triangles in Σ_1 are removed with edge collapses. Given a one-chain, each edge collapse uniquely determines the value associated the triangle that it collapses. Furthermore, collapsing edges is equivalent to Gaussian eliminations of rows with exactly one nonzero member in ∂_2. This means that the triangles and edges in the collapsing order describes the operations needed to solve the linear system $\partial_2 x_2 = b_1$. The collapsing order allows us to compute the inverse of ∂_2 via an upper-triangular matrix.

We clarify some notations before proceeding into the formal statements. For a subsequence Σ' of Σ, we denote by $E(\Sigma')$ to be the ordered set of edges that are collapsed in Σ'; note that these edges may collpase as a result of either vertex-edge collapses or edge-triangle collapses. Similarly, we denote by $V(\Sigma')$ and $F(\Sigma')$ the ordered sets of vertices and triangles, respectively, that are collapsed in Σ'.

Lemma 5.1 (Collapsing Sequence). *Let $\Sigma = \Sigma_2 \cdot \Sigma_1 \cdot \Sigma_0$ be a monotone collapsing sequence for K. Let $\overline{\Sigma}$ denote the reverse of the sequence Σ. The submatrix of ∂_2 induced by the rows $E(\overline{\Sigma}_1)$ and the columns $F(\overline{\Sigma}_1)$ is upper triangular.*

Proof: The effect of elementary collapses on $E(\overline{\Sigma}_1)$ can be viewed as removing the rows in a bottom up order in the submatrix of $E(\overline{\Sigma}_1)$ and $F(\overline{\Sigma}_1)$, while the fact that each collapse is elementary guarantees that when a row is removed, no triangles incident to it remains. Thus, all the non-zero entries in each row are to the right of the diagonal, as the columns are arranged in order of the triangles removed. □

So, we write ∂_2 as follows:

$$\begin{array}{c} \\ E(\overline{\Sigma}_0) \\ E(\overline{\Sigma}_1) \end{array} \begin{array}{c} \begin{array}{cccc} & F(\overline{\Sigma}_1) & & F(\overline{\Sigma}_2) \end{array} \\ \begin{pmatrix} \dots & \dots & \dots & \dots \\ \pm 1 & \dots & \dots & \dots \\ \mathbf{0} & \pm 1 & \dots & \dots \\ \mathbf{0} & \mathbf{0} & \pm 1 & \dots \end{pmatrix} \end{array}. \tag{5.8}$$

This upper-triangular permutation allows us to obtain a solution to $\partial_2 x_2 = b_1$ using back substitution after setting some of the coordinates of x_2 to zero.

Theorem 5.2 (Recovering ∂_2^+). *Let K be a collapsible simplicial complex of size m and $\partial_2 x_2 = b_1$ (respectively, $\partial_2^T x_1 = b_2$) be a system of equations with at least one solution. Given a collapsing sequence of K, we can find, in $O(m)$ time, a linear operator ∂_2^+ (respectively, ∂_2^{+T}) such that $\partial_2 \partial_2^+ b_1 = b_1$ (respectively, $\partial_2^T \partial_2^{+T} b_2 = b_2$). Furthermore, the solution $\partial_2^+ b_1$ (respectively, $\partial_2^{+T} b_2$) can be evaluated in $O(m)$ time.*

Proof: Lemma 2.5 implies that a monotone collapsing sequence Σ of K can be computed in linear time. Lemma 5.1 allows us to rearrange the rows and columns of ∂_2 as in Equation 5.8 so that the edges and triangles involved in Σ_1 are in upper triangular form.

The elementary collapses involving columns corresponding to $F(\overline{\Sigma}_2)$ does not decrease the rank of ∂_2, so the submatrix induced by the columns in $F(\overline{\Sigma}_1)$ still has the same rank. Therefore it suffices to invert the submatrix of ∂_2 involving $E(\overline{\Sigma}_1)$ and $F(\overline{\Sigma}_1)$. Let this invertible matrix be $\mathbf{Q}$, then ∂_2^+ can be written as:

$$\partial_2^+ = \begin{array}{c} \\ F(\overline{\Sigma}_1) \\ F(\overline{\Sigma}_2) \end{array} \begin{array}{c} \begin{array}{cc} E(\overline{\Sigma}_0) & E(\overline{\Sigma}_1) \end{array} \\ \begin{pmatrix} 0 & Q^{-1} \\ 0 & 0 \end{pmatrix} \end{array}.$$

Although Q^{-1} can be dense even when Q is sparse, evaluating $Q^{-1}\mathbf{b}$ be done by back substitution in reverse order of rows in linear time; see e.g., [Str93] for more details. Also, Q^T is a lower triangular matrix, so $Q^{-T}\mathbf{b}$ can also be evaluated in linear time.

Thus, any $\partial_2^+ b_1$ satisfying $b_1 = \partial_2 \overline{x}_2$, is a solution for $\partial_2 x_2 = b_1$. In other words, for any vector $\overline{x}_2$, we have:

$$\begin{aligned} \partial_2(\partial_2^+ b_1) &= b_1 \\ \partial_2 \partial_2^+ \partial_2 \overline{x}_1 &= \partial_2 \overline{x}_1 \end{aligned}$$

As $\overline{x}_1$ can be any vector, $\partial_2 \partial_2^+ \partial_2$ and ∂_2 are the same matrix. Similarly, it can be shown that $\partial_2^T \partial_2^{+T} \partial_2^T$ and ∂_2^T are identical. □

6 Discussion

In this paper, we have presented a nearly linear time algorithm for computing the 1-Laplacian arising from a particular class of two-complexes with trivial homology. This is the first paper attempting to solve the 1-Laplacian optimally.

Weighted Laplacian. There is a natural generalization from the Combinatorial (1-Laplacian) to the

weighted Combinatorial Laplacian. Let K be a two-complex and ∂_1 and ∂_2 the corresponding boundary matrices. Given weight matrix W_2 and W_0 (on faces and edges, respectively), the weighted Combinatorial Laplacian is the following operator:

$$\Delta_1^W := \partial_2 W_2 \partial_2^T + \partial_1^T W_0 \partial_1.$$

The techniques presented in this paper can be generalized to incorporate unit diagonal weights. However, the case of a general weight matrix is an open question. Perhaps handling more general weight matrices will look like the methods used in [BHV08].

Extending Input. The input complexes that we handle in this paper are convex three-complexes. A natural next step is to find fast solvers for Laplacians arising from more general complexes. As we mentioned in the introduction, collapsible complexes seem to be the analog of the tree that we used in graph Laplacians. An interesting open question surrounds the idea of generalizing the notion of tree to higher dimensions. In other words, can we define a class of complexes, which we call *frames*, so that we can always find a subcomplex which is a frame, find the solution on the frame, then extend the result to the entire complex?

Acknowledgments The authors would like to thank Tamal Dey, Nathan Dunfield, Anil Hirani, Don Sheehy, and the anonymous reviewers for helpful discussions and suggestions.

References

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995. Referenced in 1.1.

[AN12] Ittai Abraham and Ofer Neiman. Using petal-decompositions to build a low stretch spanning tree. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 395–406, New York, NY, USA, 2012. ACM. Referenced in 2.4.

[Axe94] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, NY, 1994. Referenced in 1.1.

[BCPT05] Erik G. Boman, Doron Chen, Ojas Parekh, and Sivan Toledo. On factor width and symmetric h-matrices. *Linear Algebra and Its Applications*, 405:239–248, 2005. Referenced in 1.1.

[BGK+13] Guy E. Blelloch, Anupam Gupta, Ioannis Koutis, Richard Miller, Gary L. and Peng, and Kanat Tangwongsan. Nearly-linear work parallel SDD solvers, low-diameter decomposition, and low-stretch subgraphs. *Theory of Computing Systems*, pages 1–34, March 2013. Referenced in 1.1.

[BHV08] Erik G. Boman, Bruce Hendrickson, and Stephen A. Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *SIAM J. Numerical Analysis*, 46(6):3264–3284, 2008. Referenced in 1.1, 6.

[Chi67] David R. J. Chillingworth. Collapsing three-dimensional convex polyhedra. *Mathematical Proceedings of the Cambridge Philosophical Society*, 63(02), 1967. Referenced in 2.5.

[Chi80] David R. J. Chillingworth. Collapsing three-dimensional convex polyhedra: correction. *Mathematical Proceedings of the Cambridge Philosophical Society*, 88(02), 1980. Referenced in 2.5.

[CK13] Chao Chen and Michael Kerber. An output-sensitive algorithm for persistent homology. *Computational Geometry: Theory and Applications*, 46(4):435 –447, 2013. Referenced in 1.1.

[CKM+11] Paul Christiano, Jonathan A. Kelner, Aleksander Mądry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 273–282, New York, NY, USA, 2011. ACM. Referenced in 1.1.

[Coh73] Marshall M. Cohen. *A Course in Simple Homotopy Theory*. Graduate texts in mathematics. Springer, New York, 1973. Referenced in 1, 2.5.

[Dai08] Daniel A. Daitch, Samuel I. and Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 451–460, New York, NY, USA, 2008. ACM. Referenced in 1.1, 1.1.

[Del93] Herbert Delfinado, Cecil Jose A. and Edelsbrunner. An incremental algorithm for Betti numbers of simplicial complexes. In *Proceedings of the 9th Annual Symposium on Computational Geometry*, pages 232–239, New York, NY, USA, 1993. ACM. Referenced in 1.1.

[DG98] Tamal K. Dey and Sumanta Guha. Computing homology groups of simplicial complexes in $\mathbb{R}^3$. *J. ACM*, 45(2):266–287, March 1998. Referenced in 1.1.

[DKT08] Mathieu Desbrun, Eva Kanso, and Yiying Tong. Discrete differential forms for computational modeling. In *Discrete Differential Geometry*, pages 287–324. Springer, 2008. Referenced in 1.1.

[DS07] Samuel I. Daitch and Daniel A. Spielman. Support-graph preconditioners for 2-dimensional trusses. *CoRR*, abs/cs/0703119, 2007. Referenced in 1.1.

[EP14] Herbert Edelsbrunner and Salman Parsa. On computational complexity of Betti numbers: Reductions from matrix rank. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2014. Referenced in 1.1.

[Epp03] David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms*, pages 599–608, Philadelphia, PA, USA, 2003. Society

for Industrial and Applied Mathematics. Referenced in 1.1.

[Fri98] Joel Friedman. Computing Betti numbers via combinatorial laplacians. *Algorithmica*, 21(4):331–346, 1998. Referenced in 1.1.

[Hat01] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2001. Referenced in 2, 3.

[Hod41] William V. D. Hodge. *The Theory and Applications of Harmonic Integrals*. Cambridge University Press, 1941. Referenced in 3.

[HS52] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, December 1952. Referenced in 1.1.

[JLYY11] Xiaoye Jiang, Lek-Heng Lim, Yuan Yao, and Yinyu Ye. Statistical ranking and combinatorial Hodge theory. *Mathematical Programming*, 127(1):203–244, March 2011. Referenced in 1.1.

[KM09] Jonathan A. Kelner and Aleksander Mądry. Faster generation of random spanning trees. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 13–21, Washington, DC, USA, 2009. IEEE Computer Society. Referenced in 1.1.

[KMP10] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD linear systems. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 235–244, Washington, DC, USA, 2010. IEEE Computer Society. Referenced in 1.1, 2.3.

[KMP11] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-$m \log n$ time solver for SDD linear systems. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 590–598, Washington, DC, USA, 2011. IEEE Computer Society. Referenced in 1.1, 2.3, 2.4.

[KOSZ13] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 911–920, New York, NY, USA, 2013. ACM. Referenced in 1.1, 2.3, 2.4, 4.2.

[LS13] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. *CoRR*, abs/1305.1922, 2013. Referenced in 1.1, 2.3, 2.4.

[Mun30] James R. Munkres. *Elements of Algebraic Topology*. Perseus Books Publishing, 1930. Reprint 1984. Referenced in 1.1.

[Pen13] Richard Peng. *Algorithm Design Using Spectral Graph Theory*. PhD thesis, Carnegie Mellon University, September 2013. Referenced in 2.4.

[Spi04] Shang-Hua Spielman, Daniel A. and Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004. Referenced in 1.1, 2.3.

[Str93] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993. Referenced in 2, 5.

[Tan12] Martin Tancer. Recognition of collapsible complexes is NP-complete. *CoRR*, abs/1211.6254, 2012. Referenced in 4.4.

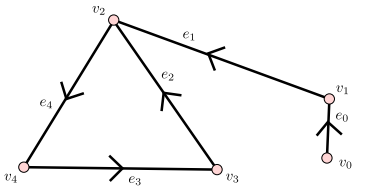
[Vai91] Pravin M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. Workshop Talk at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991. Minneapolis, MN. Referenced in 1.1.

[Whi39] John Whitehead. Simplicial spaces, nuclei and m-groups. *Proceedings of the London Mathematical Society*, 1(s2-45), 1939. Referenced in 2.5.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 887–898, New York, NY, USA, 2012. ACM. Referenced in 1.1.

A An Example

In Figure 5, we step through the construction of the graph Laplacian matrix. Consider the graph G given in Figure 5a. The boundary matrix for G, denoted ∂_1 and given in Figure 5b, is the matrix corresponding to a linear operator that maps one-chains to zero-chains. The corresponding graph Laplacian Δ_0, given in Figure 5c is given by the formula: $\Delta_0 = \partial_0 \partial_0^T$, mapping zero-chains to zero-chains.



(a) A graph from which a boundary matrix ∂_0 and Laplacian Δ_0 can be defined. Notice that this graph has a cycle: $e_1 + e_2 + e_3$.

∂_1	e_0	e_1	e_2	e_3	e_4
v_0	-1	0	0	0	0
v_1	1	-1	0	0	0
v_2	0	1	1	0	-1
v_3	0	0	-1	1	0
v_4	0	0	0	-1	1

(b) The corresponding boundary matrix which, as an operator, takes one-chains and maps them to zero-chains.

Δ_0	v_0	v_1	v_2	v_3	v_4
v_0	1	-1	0	0	0
v_1	-1	2	-1	0	0
v_2	0	-1	3	-1	-1
v_3	0	0	-1	2	-1
v_4	0	0	-1	-1	2

(c) The corresponding graph Laplacian matrix which, as an operator, takes zero-chains and maps them to zero-chains.

Figure 5: Given the graph on the left, we give both the corresponding boundary matrix ∂_1 and the corresponding graph Laplacian Δ_0.