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 $\partial_1$, 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 $i^{\text{th}}$ vertex is a head, tail, or neither of the $j^{\text{th}}$ 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 $\partial_2$, and weights on edges induce weights on triangles via the co-boundary operator $\partial_2^T$; 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 $\Delta_1^\dagger = \partial_2 \partial_2^T$ and $\Delta_1^\ddagger = \partial_1^T \partial_1$. 

---

*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 \( \text{im}(\Delta_1^T) \) and \( \text{im}(\Delta_2^T) \) 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 \( \text{im}(\partial_2) \), which in our case is equal to \( \text{im} \Delta_1^T \). 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 \( \text{im}(\partial_1^T) \), which is the same subspace as for \( \text{im} \Delta_2^T \); 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:

\[
\Delta_1^T f_1 = \partial_2 \partial_2^T f_1 = d_1^{(c)} \tag{1.1}
\]

and

\[
\Delta_2^T f_1 = \partial_1^T \partial_1^T f_1 = d_1^{(p)}, \tag{1.2}
\]

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 [KM10, KM11, 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
We believe this work can be considered as a first step while a vast number of systems of interest do not have classes of linear systems. A glance at the literature, open question here is whether ideas from graph entries of the matrix, respectively. The important faster are for embedded simplicial complexes in realizing a linear map matrix. It is known that orthogonal subspaces applications in scientific computing such as Poisson equations this fact is widely known as the Fundamental Theorem of Linear Algebra. The identity of im(AA^T) and 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 → Y and let y ∈ Y. Then, the projection of y onto im(A) is A(A^TA)^+A^T y, 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 ∈ R^n is defined, using a positive semidefinite matrix A, as ∥v∥_A = \sqrt{v^T A v}. The two-norm of a vector v ∈ 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_{v≠0} (∥x^T A∥_2 / ∥x∥_2).

We use κ(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 ≤ 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 ≥ 1. For any matrices V, A, B, with VAV^T and VBV^T defined, if A ≤ B ≤ aA, then VAV^T ≤ VBV^T ≤ 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 σ = [v_0, ..., v_k]; in this case, we write dim(σ) = 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 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 X, then we say that K triangulates 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 \( f : K_k \rightarrow \mathbb{R} \), 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 \( Ax = b \), where \( x \) and \( b \) are both \( k \)-chains and \( x \) 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: \( \text{im}(\partial_k) \) (the labeling with blue boxes) and \( \text{im}(\partial_k^T) \) (the labeling with pink ovals). Here, we see such a decomposition.

A linear boundary map \( \partial_k : C_k \rightarrow C_{k-1} \) can be defined based on a global permutation of the vertices. The columns and rows of \( \partial_k \) correspond to \( k \)-simplices and \( (k-1) \)-simplices, respectively. Given a \( k \)-simplex \( \sigma = [v_0,v_1,\ldots,v_k] \), the column of \( \partial_k \) 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 \( \sigma \). In particular, we write:

\[
\partial_k([v_0,\ldots,v_k]) = \sum_{i=0}^{k} (-1)^i [v_0,\ldots,\hat{v_i},\ldots,v_k].
\]

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:

\[
\partial_{k-1} \circ \partial_k(x_k) = 0;
\]

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

### 2.4 Combinatorial Laplacian

The \( k \)-Laplacian of a simplicial complex \( \Delta_k : C_k \rightarrow C_k \) is defined as:

\[
\Delta_k = \partial_{k+1} \partial_{k+1}^T + \partial_k^T \partial_k.
\]

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 \( \Delta_1 \), which has two parts by (2.5): \( \Delta_1^+ = \partial_2 \partial_2^T \) and \( \Delta_1^- = \partial_2^T \partial_1 \), 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 \( \Delta_1^+ x_1 \) labeled in blue.

The operators \( \partial_k \) and \( \Delta_k \) 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, \( b_0 \) is known and we solve for \( x_0 \). 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 involving graph Laplacians (\( \Delta_0 \)). The following Lemma describe a slightly generalized class of solvers.

**Lemma 2.3 (Approximate Solver).**

\([\text{Spi04, KM10, KM11, KOSZ13, LS13}]\) Given a \( n \)-by-\( n \) symmetrically diagonally dominant linear system \( A \) with \( m \) non-zero entries and an error parameter \( \varepsilon \), there is a linear operator \( \text{SolveZeroLap}(A, \varepsilon) \) such that for any vector \( b \), \( \text{SolveZeroLap}(A, \varepsilon) \) can be evaluated in time \( \tilde{O}(m \log(1/\varepsilon)) \) and:

\[
(1 - \varepsilon) A^+ \preceq \text{SolveZeroLap}(A, \varepsilon) \preceq A^+.
\]
More specifically, [KM11] 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 multiplication operations without branches. This sequence of operations gives the procedure $\text{SOLVEZERO LAP}(A, \varepsilon)$, 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 $\text{SOLVEZERO LAP}$ 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 [KM11] 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] built 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 $\tau$. In this case, an elementary collapse of $K$ at $\sigma$ gives the simplicial complex $K' = K - \tau - \sigma$. An elementary collapse at $\sigma$ is a $\dim(\sigma)$-collapse.

A simplicial complex $K$ collapses to a simplicial complex $L$ if there is a sequence of simplicial complexes $K = K_0 \supset K_1 \supset \cdots \supset K_k = L$ such that $K_i$ is obtained from $K_{i-1}$ by a collapse at $\sigma_i$. In this case, we call $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ a $(K, L)$ collapsing sequence and write $L = K \leadsto \Sigma$. 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 $(\sigma_p, \sigma_q)$, with $1 \leq p < q \leq k$, 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 $\Sigma$ 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 $(\sigma_i, \sigma_{i+1})$. Necessarily, this flipped pair appears consecutively in $\Sigma$. Let $K' = K \setminus (\sigma_1, \ldots, \sigma_{i+1})$. Then, we know that $\sigma_i$ is free in $K'$. Also, $\sigma_{i+1}$ is free in $K' \setminus \sigma_i$. Since $\dim(\sigma_i) < \dim(\sigma_{i+1})$, we have that $\sigma_{i+1}$ is also free in $K'$. It follows that $K \setminus (\sigma_1, \ldots, \sigma_{i+1}, \sigma_i) = K' \setminus (\sigma_1, \ldots, \sigma_i, \sigma_{i+1}) = K'$. 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 $(\sigma_i, \sigma_{i+1})$ be the first flipped pair, and find a new sequence $\Sigma_1$ as before. Since $\Sigma_1$ has exactly $n - 1$ flipped pairs, we can obtain a monotone collapsing from $\Sigma_1$ sequence by transposing $n - 1$ pairs by our induction hypothesis. \hfill \Box

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 $\text{im}(\partial_{k+1})$ and $\text{im}(\partial_k^T)$ are orthogonal. In the present paper, we assume that $K$ is homeomorphic to a three-ball; thus, we can assume $\text{im}(\partial_{k+1}) = \text{null}(\partial_k)$ and by the Fundamental Theorem of Linear Algebra, $\text{im}(\partial_{k+1})$ and $\text{im}(\partial_k^T)$ span the space of $k$-chains; that is:

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

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

In particular, Theorem 3.1 leads to the discrete Helmholtz decomposition, which decomposes a one-chain $x_1$ into a cobounding chain (potential flow) $x_1^T = \Pi_1^T x_1 = \partial_1^T x_0$.
and a bounding cycle (circulation) 

\[ x_i^1 = \Pi^1_i x_1 = \partial_2 x_2; \]

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 \( \Pi^1_i \) and \( \Pi^2_i \) can be approximated in nearly-linear time. Throughout the rest of the paper, we denote the the approximated operators up to \( \varepsilon \) accuracy by \( \widetilde{\Pi}^1_i(\varepsilon) \) and \( \widetilde{\Pi}^2_i(\varepsilon) \).

**Lemma 3.2 (Projections of One-Chains).** Give a graph Laplacian with \( m \) edges and any \( \varepsilon > 0 \), there exist operators \( \widetilde{\Pi}^1_i(\varepsilon) \) and \( \widetilde{\Pi}^2_i(\varepsilon) \), each computable in \( O(m \log m \log \log m / \varepsilon) \) time such that:

\[
\begin{align*}
(1 - \varepsilon)\Pi^1_i & \leq \widetilde{\Pi}^1_i(\varepsilon) \leq \Pi^1_i, \\
(1 - \varepsilon)\Pi^2_i & \leq \widetilde{\Pi}^2_i(\varepsilon) \leq \Pi^2_i.
\end{align*}
\]

**Proof:** Define 

\[ \widetilde{\Pi}^1_i(\varepsilon) = \partial_i^T \text{SOLVEZERO}\text{\textsc{lap}}(\partial_i \partial_i^T, \varepsilon) \partial_i, \]

where \( \text{SOLVEZERO}\text{\textsc{lap}} \) is the solver given in Lemma 2.3. Hence, also by Lemma 2.3, we have:

\[
(1 - \varepsilon)(\partial_i \partial_i^T)^+ \preceq \text{SOLVEZERO}\text{\textsc{lap}}(\partial_i \partial_i^T, \varepsilon) \preceq (\partial_i \partial_i^T)^+.
\]

Applying Lemma 2.2 allows us to compose this bound with the \( \partial_i^T \) and \( \partial_i \) on the left and right, respectively, obtaining:

\[
(1 - \varepsilon)\partial_i^T (\partial_i \partial_i^T)^+ \partial_i \preceq \widetilde{\Pi}^1_i(\varepsilon) \preceq \partial_i^T (\partial_i \partial_i^T)^+ \partial_i.
\]

Proving (3.7) is more intricate. For a spanning tree \( T \) of the one-skeleton of \( K \), we define a non-orthogonal projection operator \( \Pi_T \) that takes a flow and returns the unique flow using only edges from \( T \) satisfying the same demands. We note here that both \( \Pi_T \) and \( \Pi^1_T \) can be computed in linear time. Since \( \Pi^1_i \) returns a cycle (circulation), we have \( \Pi_T \Pi^1_i = 0 \), which implies:

\[ (I - \Pi_T) \Pi^1_i (I - \Pi_T)^T = \Pi^1_i. \]

We define the approximate projection operator, based on this equivalent representation of \( \Pi^1_i \) and the fact that \( \Pi^1_i = I - \Pi^1_T \):

\[ \widetilde{\Pi}^1_i(\varepsilon) = (1 - \varepsilon)(I - \Pi_T) \left( I - \Pi^1_i(\varepsilon/m^2) \right) (I - \Pi_T)^T. \]

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

\[
(1 - \varepsilon/m^2)\Pi^1_i \preceq \widetilde{\Pi}^1_i(\varepsilon/m^2) \preceq \Pi^1_i,
\]

\[
\Pi^1_i \preceq I - \widetilde{\Pi}^1_i(\varepsilon/m^2) \preceq \Pi^1_i + \varepsilon/m^2 I.
\]

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

\[
(I - \Pi_T)\Pi^1_i(I - \Pi_T)^T
\]

\[ \preceq (I - \Pi_T)(I - \Pi^1_i(\varepsilon/m^2))(I - \Pi_T)^T \]

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

\[
(1 - \varepsilon)\Pi^1_i \leq \widetilde{\Pi}^1_i(\varepsilon) \preceq (1 - \varepsilon)(\Pi^1_i + \varepsilon/m^2(I - \Pi_T)(I - \Pi_T)^T).
\]

Therefore it remains to upper bound \( \varepsilon/m^2(I - \Pi_T)(I - \Pi_T)^T \) spectrally by \( \Pi^1_i \). Note \( \Pi_T \) maps each off-tree edge to all tree edges generated by its fundamental cycle, and diagonal entries of \( \Pi_T \) are non-zero when the edge is on the tree. Therefore, each matrix element of \( I - \Pi_T \) has absolute value at most one. This allows us to bound the spectral norm of \( (I - \Pi_T)^T(I - \Pi_T) \):

\[
(I - \Pi_T)^T(I - \Pi_T) \preceq m^2 I.
\]

Applying Lemma 2.2 again, with \( \Pi^1_i \) as the outer matrix, gives:

\[
\Pi^1_i(I - \Pi_T)(I - \Pi_T)^T \Pi^1_i \preceq \Pi^1_i(m^2I) \Pi^1_i \preceq m^2 \Pi^1_i.
\]

The last relation follows from \( I \) commuting with \( \Pi^1_i \) and \( \Pi^1_i \) being a projection matrix.

Since \( I - \Pi_T \) returns a cycle (circulation), we have:

\[ \Pi^1_i(I - \Pi_T) = I - \Pi_T. \]

We multiply each side by its transpose to obtain:

\[ \Pi^1_i(I - \Pi_T)(I - \Pi_T)^T \Pi^1_i = (I - \Pi_T)(I - \Pi_T)^T. \]

Hence we have \((I - \Pi_T)(I - \Pi_T)^T \preceq m^2 \Pi^1_i \). Putting everything together gives the desired bounds on \( \widetilde{\Pi}^1_i \):

\[
(1 - \varepsilon)\Pi^1_i \preceq \widetilde{\Pi}^1_i(\varepsilon) \preceq (1 - \varepsilon)(\Pi^1_i + \varepsilon/m^2 \cdot m^2 \Pi^1_i) \preceq (1 - \varepsilon)(1 + \varepsilon)\Pi^1_i \preceq \Pi^1_i.
\]
As a result of the above theorem, \( \Pi^t_2 \) and \( \Pi^z_2 \) are approximations of \( \Pi^t_1 \) and \( \Pi^z_1 \), respectively. We can obtain similar approximating operators for \( \Pi^t_2 \) and \( \Pi^z_2 \) 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 boundary by adding a special vertex \( \sigma^* \) in \( K^* \) called the infinite vertex and connecting \( \sigma^* \) 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 \( \sigma^* \) corresponds to \( S^3 \setminus K \). 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 \( \sigma \). 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 \( \Pi^t_2 \) and \( \Pi^z_2 \) be as defined above. Then, for any \( \varepsilon > 0 \), the operators \( \Pi^t_2(\varepsilon) \) and \( \Pi^z_2(\varepsilon) \) can be computed in \( O(m \log m \log \log m / \varepsilon) \) time such that

\[
\begin{align*}
(1 - \varepsilon) \Pi^t_2 & \leq \Pi^t_2(\varepsilon) \leq \Pi^t_2, \\
(1 - \varepsilon) \Pi^z_2 & \leq \Pi^z_2(\varepsilon) \leq \Pi^z_2.
\end{align*}
\]

### 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: \( \Delta_1 = \Delta^t_1 + \Delta^z_1 \). 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 \( \Delta_1 x_1 = b_1, \Delta^t_1 y_1 = b^t_1 \), and \( \Delta^z_1 z_1 = b^z_1 \). Then, the following holds:

\[
x_1 = y^t_1 + z^z_1.
\]

**Proof:** Recall that \( \text{im}(\Delta^t_1) \) and \( \text{im}(\Delta^z_1) \) orthonogally decompose \( \text{im}(\Delta_1) \). Thus, we have \( y^t_1, z^z_1 \in \text{null}(\Delta^t_1) \) and \( y^t_1, z^z_1 \in \text{null}(\Delta^z_1) \), which gives us:

\[
\Delta_1 (y^t_1 + z^z_1) = (\Delta^t_1 + \Delta^z_1)(y^t_1 + z^z_1) = (\Delta^t_1 y^t_1 + \Delta^z_1 z^z_1) + (\Delta^t_1 z^z_1 + \Delta^z_1 y^t_1) = \Delta^t_1 y^t_1 + \Delta^z_1 z^z_1 = b^t_1 + b^z_1.
\]

The third equality holds since \( \Delta^t_1 y^t_1, \Delta^z_1 z^z_1, \Delta^t_1 z^z_1 \) and \( \Delta^z_1 y^t_1 \) are trivial. \( \square \)

#### 4.2 Down Operator

Now that we have divided the problem into two parts, \( \Delta^t_1 y_1 = b^t_1 \) and \( \Delta^z_1 z_1 = b^z_1 \), we begin with \( \Delta^t_1 \). As \( \Delta^t_1 \) is defined as \( \partial_T^T \partial_T \), it is helpful to remind the reader the combinatorial interpretation of \( \partial_T \). Given one-chain (a flow) \( f_1, \partial_T f_1 \) is the residue of the flow at all the vertices. On the flip side, given zero-chain (vertex potentials) \( p_0, \partial_T^T p_0 \) 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 \( \Delta^t_1 x_1 = b^t_1 \) is a simpler operation than solving a graph Laplacian. The following lemma provides a linear operator \( (\Delta^t_1)^+ \) to solve \( \Delta^t_1 x_1 = b^t_1 \); we highlight the assumption that \( b^t_1 \in \text{im}(\Delta^t_1) \).

**Lemma 4.2 (Down Solver).** Let \( K \) be any graph and suppose \( \Delta^t_1 x_1 = b^t_1 \) has at least one solution. Then, a linear operator \( (\Delta^t_1)^+ \) exists such that \( (\Delta^t_1)^+ b^t_1 = b^t_1 \). Furthermore, \( (\Delta^t_1)^+ \) and \( (\Delta^t_1)^+ b^t_1 \) can be computed in linear time.

**Proof:** Recall \( \Delta^t_1 = \partial_T^T \partial_T \). We first find \( z_0 = \sum_i c_i v_i \) such that \( \partial_T^T z_0 = b^t_1 \); then we will find \( z_1 \) that satisfies \( \partial_T z_1 = z_0 \).

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 $T$ 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 $T$, knowing $c_i$ implies a unique value for $c_k$. Letting $v_0$ be a root of $T$, we set $c_0 = 0$. Then, we can uniquely determine all values $c_i$ by traversing the edges of $T$.

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

\[
0 = 1^T z_0^+ = 1^T (z_0 - c 1) = 1^T z_0 - c 1^T 1
\]

Thus, we can compute $z_0^+$ (and $z_0^-$) by computing $c$ by finding the unique $c$ such that $1^T z_0 - c 1^T 1 = 0$.

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

It is straightforward to put the used operations together to get the linear operator $(\Delta_1^+)^+$. In fact, the whole process can be seen as collapsing forward (and expanding backward) the spanning tree $T$. 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 $\Delta_1^+ y_1 = b_1^+$ proceeds similarly. Ideally, we want a two-chain $b_2$ such that a solution $y_1$ exists satisfying $\partial_2 b_2 = b_1^+$ and $\partial_2^+ y_1 = b_2$. Having such a $b_2$, we can solve the equation $\partial_2^+ y_1 = b_2$ to obtain $y_1$. Given any two-chain $y_2$ such that $\partial_2 y_2 = b_1^+$, we observe in Lemma 4.3 that $b_2 = y_2^+$ has the desired properties.

Surprisingly (compared to the lower dimensional case), solving $\partial_2 y_2 = b_1^+$ to find any solution $y_2$ is not straightforward. In Section 5 we describe an algorithm for the special case where $K$ is collapsible and the collapsing sequence is known. Let's call the operator to solve the equation under this condition $\partial_2^+$ (see Theorem 5.2), and recall the projection operator $\Pi^2_+$ of Theorem 5.2. The following lemma describes a linear operator to solve $\Delta_1^+ y_1 = b_1^+$.

Lemma 4.3 (Up Solver). Let $K$ be a triangulation of a three-ball and let $(\Delta_1^+)^+ = \partial_2^+ (\Delta_1^+)^+$. If $\Delta_1^+ x_1 = b_1^+$ has at least one solution, then $(\Delta_1^+)^+ y_2$ is a solution.

**Proof:** By Theorem 5.2, the two-chain $y_2 = \partial_2^+ b_1^+$ is a solution to $\partial_2 y_2 = b_1^+$. Consider the decomposition $y_2 = y_2^+ + y_2^-$.

We have:

\[
b_1^+ = \partial_2 y_2 = \partial_2 (y_2^+ + y_2^-) = \partial_2 y_2^+.
\]

The last equality follows from the facts $y_2^+ \in \text{im}(\partial_1)$ and $\partial_2 \partial_3 = 0$.

Since $y_2^+ \in \text{im}(\partial_2^+)$, there exists a $y_1$ such that $\partial_2^+ y_1 = y_2^+$. Applying Theorem 5.2 again, we obtain $y_1 = (\partial_2^+)^+ y_2^+$. Since $y_2^+ = \Pi^2_+ y_2$ and $y_2 = \partial_2^+ b_1^+$, the statement of the lemma follows from Theorem 3.1.

The first and last steps are solving $\partial_2 y_2 = b_1^+$ and $\partial_2^+ y_1 = y_2^+$ (equivalently, computing $\partial_2^+$). 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 $\varepsilon$) the parts of $(\Delta_1^+)^+$ 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 $\text{SolveUpLap}((\Delta_1^+)^+, K, \varepsilon)$ with the following property can be computed in $O(m \log 1/\varepsilon)$ time:

\[
(1 - \varepsilon)(\Delta_1^+)^+ \preceq \text{SolveUpLap}((\Delta_1^+)^+, K, \varepsilon) \preceq (\Delta_1^+)^+.
\]

4.4 Summing Up. Lemma 4.1, Lemma 4.2 and Lemma 4.3 imply that $\Delta_1^+ = \Pi_1^2 (\Delta_1^+)^+ \Pi_1^2 + \Pi_1^2 (\Delta_1^+)^+ \Pi_1^2$ 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, $\Pi$ be an orthogonal projection and $\tilde{\Pi}$ be a linear operator that satisfies $-\varepsilon \Pi \leq \Pi - \tilde{\Pi} \leq \varepsilon \Pi$. Then, we have:

\[
(1 - 3\mu^2(A))\Pi A \Pi \preceq \Pi \tilde{\Pi} \Pi \preceq (1 + 3\mu^2(A))\Pi A \Pi
\]

where $\mu^2(A)$ is the condition number of $A$ restricted to the subspace of the image of $\Pi$.

**Proof:** The proof first establishes a matrix norm bound. This follows from the triangle inequality and...
the fact that \( \|\Pi\|_2 \leq 1 \) (as \( \Pi \) is a projection matrix). In particular, we have:

\[
\begin{align*}
\|\Pi A\Pi - \Pi A\Pi\|_2 \\
= \|\Pi A\Pi - \Pi A\Pi + \Pi A\Pi - \Pi A\Pi\|_2 \\
\leq \|\Pi A\Pi - \Pi A\Pi\|_2 + \|\Pi A\Pi - \Pi A\Pi\|_2 \\
\leq \|\Pi\|_2 \|A\|_2 \|\Pi - \Pi\|_2 + \|\Pi - \Pi\|_2 \|A\|_2 \|\Pi\|_2 \\
\leq 3\varepsilon \lambda_{\max}^\Pi(A)
\end{align*}
\]

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

\[-3\varepsilon \lambda_{\max}^\Pi(A) \Pi \leq \Pi A\Pi - \Pi A\Pi \leq 3\varepsilon \lambda_{\max}^\Pi(A) \Pi\]

But

\[3\varepsilon \lambda_{\max}^\Pi(A) \Pi = 3\varepsilon \lambda_{\max}^\Pi(A) \Pi\Pi/\Pi \\
\leq 3\varepsilon \lambda_{\min}^\Pi(A) \Pi A\Pi \\
= 3\varepsilon \lambda_{\max}^\Pi(A) \Pi A\Pi
\]

This gives

\[-3\varepsilon \lambda_{\max}^\Pi(A) \Pi A\Pi \leq \Pi A\Pi - \Pi A\Pi \leq 3\varepsilon \lambda_{\max}^\Pi(A) \Pi A\Pi \\
(1 - 3\varepsilon \lambda_{\max}^\Pi(A)) \Pi A\Pi \leq \Pi A\Pi \leq (1 + 3\varepsilon \lambda_{\max}^\Pi(A)) \Pi A\Pi
\]
as desired. \( \square \)

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 \( \text{SolveOneLap}(\Delta_1, K, \varepsilon) \) and the vector \( \text{SolveOneLap}(\Delta_1, K, \varepsilon) \cdot b_1 \) for any one-chain \( b_1 \) can be computed in \( O(m \log m \log \log m \log \kappa/\varepsilon) \) time, where \( \kappa \) is the maximum of the condition numbers of the up and down Laplacians, such that

\[(1 - \varepsilon)\Delta_1^+ \preceq \text{SolveOneLap}(\Delta_1, K, \varepsilon) \preceq \Delta_1^+
\]

**Proof:** In this proof we write the projection operators of the form \( \Pi_{\varepsilon'}(c') \) more concisely as \( \Pi_{\varepsilon'}^+ \) by dropping the parameter \( \varepsilon' \).

Consider the operator

\[
\text{SolveOneLap}(\Delta_1, K, \varepsilon) = \\
(1 - \varepsilon/2)\Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+(\varepsilon/6\lambda) + \\
(1 - \varepsilon/3)\Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+(\varepsilon/9\lambda) \Pi_{\varepsilon/9\lambda}^+ in K.
\]

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:

\[
(1 - \varepsilon/2)\Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+ \Pi_{\varepsilon/6\lambda}^+(\varepsilon/6\lambda) + \\
(1 - \varepsilon/3)\Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+ \Pi_{\varepsilon/9\lambda}^+(\varepsilon/9\lambda) \Pi_{\varepsilon/9\lambda}^+ in K.
\]

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

\[
(1 - \varepsilon/3)\Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+(\varepsilon/3\lambda) + \\
(1 - \varepsilon/3)\Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+ \Pi_{\varepsilon/3\lambda}^+(\varepsilon/3\lambda) \Pi_{\varepsilon/3\lambda}^+ in K.
\]

The first chain of inequalities follows from Lemma 4.4 and the second from Lemma 4.5. \( \square \)

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 \( \text{SolveOneLap}(\Delta_1, K, \varepsilon) \) and the vector \( \text{SolveOneLap}(\Delta_1, K, \varepsilon) \cdot b_1 \) for any one-chain \( b_1 \) can be computed in \( O(m \log m \log \log m \log \kappa/\varepsilon) \) time, where \( \kappa \) is the maximum of the condition numbers of the up and down Laplacians, such that

\[(1 - \varepsilon)\Delta_1^+ \preceq \text{SolveOneLap}(\Delta_1, K, \varepsilon) \preceq \Delta_1^+
\]

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 \( \Sigma \) of \( K \). The key insight is to view the Gaussian elimination operators as simplicial collapses.
By Lemma 2.5, we can assume without loss of generality that $\Sigma$ is monotone. Therefore, $\Sigma$ may be expressed as $\Sigma_2 \cdot \Sigma_1 \cdot \Sigma_0$, where $\Sigma_i$ 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 $\partial_2$, 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 $\Sigma_1$ are removed with edge collapses. Given a one-chain, each edge collapse uniquely determines the value associated to the triangle that it collapses. Furthermore, collapsing edges is equivalent to Gaussian eliminations of rows with exactly one nonzero member in $\Sigma_2$; note that these edges may collapse 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 $\Sigma'$.

**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 $\Sigma$. The submatrix of $\partial_2$ 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 $E(\Sigma_1)$ can be viewed as removing the rows in a bottom up order in the submatrix of $E(\Sigma_1)$ and $F(\Sigma_1)$, 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 $\partial_2$ as follows:

\[
\begin{pmatrix}
E(\Sigma_0) & F(\Sigma_1) & F(\Sigma_2) \\
E(\Sigma_1) & \pm 1 & \ldots & \ldots & \ldots \\
E(\Sigma_2) & 0 & \pm 1 & \ldots & \ldots \\
\end{pmatrix}
\]

(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 $\partial_2^+$).** Let $K$ be a collapsible simplicial complex of size $m$ and $\partial_2 x_2 = b_1$ (respectively, $\partial_2^+ x_1 = b_2$) 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 $\partial_2^+$ (respectively, $\partial_2^{+T}$) such that $\partial_2 \partial_2^+ b_1 = b_1$ (respectively, $\partial_2^T \partial_2^+ b_2 = b_2$). 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 $\Sigma$ of $K$ can be computed in linear time. Lemma 5.1 allows us to rearrange the rows and columns of $\partial_2$ as in Equation 5.8 so that the edges and triangles involved in $\Sigma_1$ are in upper triangular form.

The elementary collapses involving columns corresponding to $F(\Sigma_2)$ do not decrease the rank of $\partial_2$, so the submatrix induced by the columns in $F(\Sigma_1)$ still has the same rank. Therefore it suffices to invert the submatrix of $\partial_2$ involving $E(\Sigma_1)$ and $F(\Sigma_1)$. Let this invertible matrix be $Q$, then $\partial_2^+$ can be written as:

\[
\partial_2^+ = \begin{pmatrix}
E(\Sigma_0) & E(\Sigma_1) \\
F(\Sigma_2) & \begin{pmatrix}
0 & Q^{-1} \\
0 & 0
\end{pmatrix}
\end{pmatrix}
\]

Although $Q^{-1}$ can be dense even when $Q$ is sparse, evaluating $Q^{-1} b$ 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 $Q^{-T} b$ can also be evaluated in linear time.

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

\[
\begin{align*}
\partial_2 (\partial_2^+ b_1) &= b_1 \\
\partial_2 \partial_2^+ \partial_2 \overline{x}_2 &= \partial_2 \overline{x}_2
\end{align*}
\]

As $\overline{x}_2$ can be any vector, $\partial_2 \partial_2^+ \partial_2$ and $\partial_2$ are the same matrix. Similarly, it can be shown that $\partial_2^T \partial_2^+ \partial_2^T$ and $\partial_2^T$ 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
A natural next step is to find fast solvers for Laplacian matrices. Let $K$ be a two-complex and $\partial_1$ and $\partial_2$ 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:

$$
\Delta^W_1 := \partial_2 W_2 \partial_2^T + \partial_1^T W_0 \partial_1.
$$

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 the case of a general weight matrix is an open question. Extending Input.

**References**


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 $\partial_1$ and given in Figure 5b, is the matrix corresponding to a linear operator that maps one-chains to zero-chains. The corresponding graph Laplacian $\Delta_0$, 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 $\partial_0$ and Laplacian $\Delta_0$ can be defined. Notice that this graph has a cycle: $e_1 + e_2 + e_3$.

\[
\begin{array}{c|cccc}
\partial_1 & e_0 & e_1 & e_2 & e_3 & e_4 \\
\hline
v_0 & -1 & 0 & 0 & 0 & 0 \\
v_1 & 1 & -1 & 0 & 0 & 0 \\
v_2 & 0 & 1 & 1 & 0 & -1 \\
v_3 & 0 & 0 & -1 & 1 & 0 \\
v_4 & 0 & 0 & 0 & -1 & 1 \\
\end{array}
\]

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

\[
\begin{array}{c|cccc}
\Delta_0 & v_0 & v_1 & v_2 & v_3 & v_4 \\
\hline
v_0 & 1 & -1 & 0 & 0 & 0 \\
v_1 & -1 & 2 & -1 & 0 & 0 \\
v_2 & 0 & -1 & 3 & -1 & -1 \\
v_3 & 0 & 0 & -1 & 2 & -1 \\
v_4 & 0 & 0 & -1 & -1 & 2 \\
\end{array}
\]

(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 $\partial_1$ and the corresponding graph Laplacian $\Delta_0$. 