Computational Topology in a Collapsing Universe: Laplacians, Homology, Cohomology*

Mitchell Black† William Maxwell† Amir Nayyeri† Eli Winkelman†

Abstract

We consider a variety of topology problems on a d-dimensional simplicial complex K given that $K \subset X$ for X a collapsible simplicial complex embedded in $\mathbb{R}^{d+1}$ with known collapsing sequence.

Our first result is a solver for the linear system $L_1 x = b$, where L_1 is the 1-Laplacian of a simplicial complex K with $\dim H_1(K) = 0$ and $K \subset X$ for X a collapsible simplicial complex embedded in $\mathbb{R}^3$ with a known collapsing sequence. Our algorithm runs in $\tilde{O}(n \log^2(n\kappa/\varepsilon))$ time, where n is the total number of vertices, edges, and triangles in X, κ is the largest condition number of the two parts of the Laplacian, and ε quantifies the approximation quality. This result is a generalization of Cohen et al. [SODA 2014]. The new technical piece of our Laplacian solver, in addition to the machinery described by Cohen et al., is an algorithm to compute a bounding chain of a 1-cycle within K.

In addition, we describe faster algorithms for testing null-homology of $(d-1)$-cycles and null-cohomology of d-cocycles. Our algorithm runs in $O(n_d)$ time, where n_d is the number of d-simplices in X.

Finally, we describe an algorithm to compute a $(d-1)$-cohomology basis from a given $(d-1)$-homology basis for a d-simplicial complex K in $O(\beta_{d-1} n_d)$ time; β_{d-1} is the rank of the $(d-1)$st homology group of K. In particular, we can obtain a cohomology basis for subcomplexes of a collapsible complex X embedded in $\mathbb{R}^3$ in $O(n_d \log n_d + \beta_{d-1} n)$ time using a homology basis computed by the algorithm of Dey [SODA 2019].

For all of the problems above, if $K \subset \mathbb{R}^3$ and the collapsible supercomplex X is not provided, we can expand K into a convex ball of possibly quadratic complexity, which is known to be collapsible, resulting in nearly quadratic time algorithms.

*This research was supported in part by NSF grants CCF-1941086, CCF-1816442, and CCF-1617951.

†Oregon State University.

1 Introduction

Efficiently solving a linear system, $Ax = b$, is the central problem in numerical linear algebra. On one hand, we have direct solvers built on Gaussian elimination. These solvers are exact, yet with their super-quadratic running time, they are not sufficiently fast for many applications. On the other hand, we have iterative solvers that produce approximate solutions but are much faster, especially for sparse matrices. These solvers work through several iterations, each composed of a constant number of matrix-vector multiplications. Hence, their running time is bounded by $O(m \cdot t)$, where m is the number of nonzero elements in A, and t is the number of iterations required to achieve the desired accuracy. Ideally, we want iterative algorithms that converge in polylog(m) iterations to obtain nearly linear time solvers.

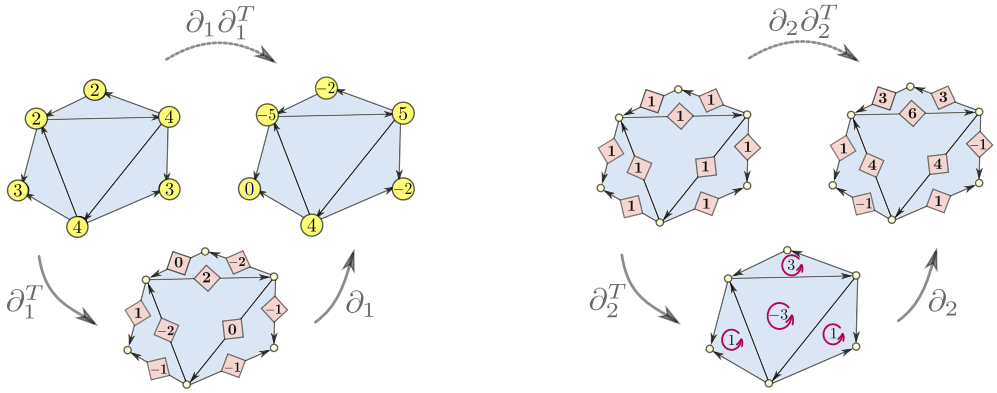
As a result of decades of research [29, 3, 26, 27, 2, 4, 25, 13, 23], we have nearly linear time solvers when A is the Laplacian of a graph. Based on these Laplacian solvers, we obtain nearly linear time solvers for the broader class of symmetric diagonally dominant matrices: matrices A where $A = A^T$ and the value of every diagonal entry of A is larger than the sum of the absolute values of all other entries on the same row. Fast graph Laplacian solvers are applied to solve elliptic partial differential equations (PDEs) [5].

The success of graph Laplacian solvers motivates research towards faster solvers for a similar, yet more general, classes of matrices: combinatorial Laplacians. Such solvers would pave the way for obtaining more efficient solvers for more general PDEs like those involving the vector Laplacian. Moreover, they provide insights for solving classes of matrices that are not diagonally dominant.

Let K be a simplicial complex composed of vertices, edges, triangles, tetrahedra, and higher dimensional simplices. There is a sequence of matrices associated with K called combinatorial Laplacians. The dth matrix in this sequence acts on vectors that assign values to the d-dimensional simplices and is defined as

$$L_d = \partial_{d+1}\partial_{d+1}^T + \partial_d^T \partial_d,$$

where ∂_d is the dth boundary matrix of K. The first matrix in this sequence, L_0, is composed of a constant map $\partial_0^T \partial_0$ and the known graph Laplacian $\partial_1 \partial_1^T$, with ∂_1^T being the incidence matrix of an arbitrary orientation of the graph. The 1-Laplacian, L_1 is the sum of $\partial_1^T \partial_1$ and $\partial_2 \partial_2^T$. The former is composed of the same matrices of the graph Laplacian but in different order, and techniques from graph Laplacian solvers can be used to solve systems involving it. The latter, however, includes a higher dimensional boundary matrix, ∂_2, a matrix that captures the incidence relation between triangles and edges of the complex. Solving a system with this matrix requires more novel techniques. The Figure below shows two examples of applying $\partial_1 \partial_1^T$ (the graph Laplacian), and $\partial_2 \partial_2^T$, on the left and right side, respectively.



Laplacian solvers. In this paper, among other problems, we study solvers for 1-Laplacians, the second matrix in the sequence of combinatorial Laplacians.

Cohen et al. [12] initiated the study of 1-Laplacian solvers by describing a nearly linear time solver provided that the input complex K is collapsible and embedded in $\mathbb{R}^3$. In particular, their solver works for a piecewise linear triangulation of a convex ball in $\mathbb{R}^3$, which is known to be collapsible. Being the first work on this subject, the algorithm is restrictive as it requires the collapsing sequence of K, which is NP-hard to compute in general. The algorithm cannot produce any solution if a collapsing sequence of K is not known, even if K is very similar to a collapsible complex, for example where K is an almost convex ball.

In this paper, we study the case that K is embedded, not necessarily collapsible, but can be extended into a collapsible complex X. Our solvers work in nearly linear time with respect to the complexity of X.

THEOREM 1.1. *Let X be a collapsible simplicial complex with a known collapsing sequence embedded in $\mathbb{R}^3$, and let $K \subset X$ such that $H_1(K) = 0$. For any $\varepsilon > 0$, there is an operator $LaplacianSolver(X, K, \varepsilon)$ such that*

$$(1-\varepsilon)(L_1[K])^+ \preceq LaplacianSolver(X, K, \varepsilon) \preceq (L_1[K])^+.$$

where $(L_1[K])^+$ is the pseudoinverse of the 1-Laplacian $L_1[K]$. Further, for any $x \in C_1$, $LaplacianSolver(X, K, \varepsilon) \cdot x$ can be computed in $\tilde{O}(n \log^2(n\kappa/\varepsilon))$ time, where $n = n_0 + n_1 + n_2$ is the total number of vertices, edges and triangles in X, and κ is the condition number of $L_1^{up}[K]$ within the space of boundary cycles.

In comparison with the work of Cohen et al. [12], the new technical contribution of this paper is a nearly linear time algorithm for computing a bounding 2-chain of a boundary 1-chain in K (Lemma 3.9). All other pieces work as described by Cohen et al. We include a slightly modified presentation of some of their results in the Section 4 to be self sufficient.

In principle, K can always be extended to a piecewise linear convex ball, which is known to be collapsible and whose collapsing sequence can be computed in linear time. The complexity of this ball can be quadratic in worst case resulting in the following corollary. One can view this corollary as an algorithm whose running time depends on how non-convex K is; our algorithm is faster if K extends to convex ball by adding fewer simplices.

COROLLARY 1.1. *Let K be a simplicial complex embedded in $\mathbb{R}^3$ with $H_1(K) = 0$. For any $\varepsilon > 0$, there is an operator $LaplacianSolver(K, \varepsilon)$ such that*

$$(1-\varepsilon)(L_1[K])^+ \preceq LaplacianSolver(K, \varepsilon) \preceq (L_1[K])^+.$$

Further, for any $x \in C_1$, $LaplacianSolver(K, \varepsilon) \cdot x$ can be computed in $\tilde{O}(n^2 \log^2(n\kappa/\varepsilon))$ time, where $n = n_0 + n_1 + n_2$ is the total number of vertices, edges and triangles in K, and κ is the maximum of the condition number of $L_1^{up}[K]$ within the space of boundary cycles and the condition number of $L_1^{down}[K]$ within the space of coboundary cycles.

Testing homology/cohomology. Laplacian matrices are defined using the boundary matrices, ∂_d. Computation with boundary and coboundary matrices is key in any computational problem about homology. Perhaps the most basic problem of this kind is to decide whether a cycle is null-homologous or whether a cocycle is null-cohomologous. To test the null-homology for a cycle γ, one needs to solve $\partial_d x = \gamma$, or more accurately decide if this system of equations has a solution. For testing null-cohomology, one needs to solve a similar linear system. The general algorithm for solving these problems have running times of $O(n^\omega)$. Faster algorithms exists only for very restrictive families of complexes like surfaces. In this paper, we study testing homology in d-simplicial complexes K assuming that K is contained in a collapsible $(d+1)$-complex X in $\mathbb{R}^{d+1}$. We obtain linear time testers with respect to the size of X for testing homology of $(d-1)$-cycles and d-cocycles.

COROLLARY 1.2. *Let X be a collapsible $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$, and let $K \subset X$. There is an $O(n_d)$ time algorithm for deciding if a $(d-1)$-cycle of K is null-homologous (in K), where n_d is the number of d-simplices in X.*

COROLLARY 1.3. *Let X be a collapsible $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$, and let $K \subset X$. There is an $O(n_d)$ time algorithm for deciding if a d-cocycle of K is null-cohomologous (in K), where n_d is the number of d-simplices in X.*

For $d = 2$, we obtain a quadratic time testers with respect to the size of K, even if X is not provided, by extending K into a super complex that is convex, thus collapsible.

Computing Cohomology Basis. Computing bases for homology and cohomology is a fundamental problem in computational topology. In contrast to computing homology ranks, not much is known about computing homology bases beyond the general linear algebraic algorithm that involves matrix multiplication, so requires $\Omega(n^\omega)$ time [6]. For complexes embedded in $\mathbb{R}^3$, Dey [15] describes an algorithm for computing a $\mathbb{Z}_2$-homology

basis in $O(n \log n + k)$ time, where k is the size of the output. Dey's algorithm implies that an $\mathbb{R}$-homology basis can be computed in the same running time. Dey proposes computing a cohomology basis efficiently as an open problem. Within the setting of this paper, we show that a cohomology basis can be computed within the same asymptotic time bound as Dey's algorithm. In fact, we show for any d and within our setting, there is an algorithm for computing a cohomology basis from a given homology basis. This result holds for both $\mathbb{Z}_2$-homology and $\mathbb{R}$-homology.

LEMMA 1.1. *Let X be a collapsible $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$, and let $K \subset X$. Also, let $\Gamma = \{\gamma_1, \ldots, \gamma_{\beta_{d-1}}\}$ be a homology basis for $C_{d-1}(K)$. There is a linear operator C such that $C_\Gamma = \{C\gamma_1, \ldots, C\gamma_{\beta_{d-1}}\}$ is a cohomology basis for $C_{d-1}(K)$. Further, C_Γ can be computed in $O(\beta_{d-1} \cdot n_d)$ time, where n_d is the number of d-simplices in X and β_{d-1} is the rank of the $(d-1)$st homology group of K.*

Therefore, by Dey's result, we obtain the following corollary for the case $d = 2$.

COROLLARY 1.4. *Let K be a simplicial complex embedded in $\mathbb{R}^3$, there is an $O(n^2 \log n)$ time algorithm to compute a cohomology basis for K.*

Paper Organization. We give a brief overview of the related work in the rest of this section. In Section 2, we define basic concepts and notations used throughout the paper. In Section 3, we describe our algorithm for finding a bounding d-chain of a $(d-1)$-boundary cycle. In Section 6, we describe our algorithm for computing a cohomolgy basis, provided a homology basis. Finally, in Section 4, we describe our 1-Laplacian solver.

1.1 Related Work

Laplacian Solvers. The conjugate gradient method can be used to solve a Laplacian linear system in $O(mn)$ time, where m is the number of nonzero entries of a matrix and n is its size. Vaidya [31] showed that minimum stretch spanning trees are good preconditioners for graph Laplacians. Following his observation, a long sequence of significant results resulted in nearly linear time solvers for graph Laplacians [29, 3, 26, 27, 2, 4, 25, 13, 23]. These solvers have been applied to a series of other problems, for example problems in algorithm design [32], and solving elliptic PDEs [5].

Cohen et al. [12] describe the first nearly linear time solvers for 1-Laplacians of collapsible complexes. In this paper, we show a similar result for a subcomplex of a collapsible complex.

Homology. Testing whether a cycle is null-homologous, or equivalently whether two cycles are homologous, is a fundamental problem in computational topology. The problem reduces to solving a linear system, thus can be solved in $O(n^\omega)$ time in general, where ω is the matrix multiplication constant. For surfaces of constant genus, testing homology can be done in nearly linear time, for example by constructing a tree cotree structure [18, 19]. Testing homology for a collapsible complexes is a trivial problem, as all cycles are null-homologous. In this paper, we show linear time algorithms provided a collapsible super complex (with respect to the size of the super complex). Also, we show nearly quadratic time algorithms for embedded 2-complexes in $\mathbb{R}^3$.

From the technical point of view, Delfinalo and Edelsbrunner's incremental algorithm for computing Betti numbers [14] is similar to our homology testing algorithm as it iteratively removes edges from the dual graph; however, the two algorithms have different objectives and use different properties of the dual graph. Dey and Guha [16] show a topological algorithm for computing Betti numbers. Friedman [20] uses power methods to find out the multiplicity of the zero eigenvalue of the Laplacian matrices, which are equal to the Betti numbers.

2 Preliminaries

In this section, we review standard concepts from linear algebra and algebraic topology used in this paper. For further background we refer the reader to references [7, 22, 30, 21].

Linear Maps. A linear map $A : \mathbb{R}^m \to \mathbb{R}^n$ can be represented by an $n \times m$ matrix; we use A to refer to the linear map and the matrix. The ***kernel*** of A, denoted $\ker(A)$, is the subset of $\mathbb{R}^m$ that A maps to zero. The ***image*** of A, denoted $\mathrm{im}(A)$, is the subset of $\mathbb{R}^n$ composed of all vectors Ax for $x \in \mathbb{R}^m$. Further, $\ker^\perp(A)$ is composed of all vectors in $\mathbb{R}^m$ that are perpendicular to $\ker(A)$. Similarly, $\mathrm{im}^\perp(A)$ is the set of all vectors in $\mathbb{R}^n$ that are perpendicular to $\mathrm{im}(A)$. The Fundamental Theorem of Linear Algebra says that $\mathrm{im}^\perp(A) = \ker(A^T)$ and $\ker^\perp(A) = \mathrm{im}(A^T)$ where A^T is the transpose of A.

Let W be a vector subspace of $\mathbb{R}^n$, and let $W^\perp$ be the subspace of all vectors perpendicular to W. Any vector $x \in \mathbb{R}^n$ can be uniquely decomposed as $x = x_W + x_{W^\perp}$, where $x_W \in W$ and $x_{W^\perp} \in W^\perp$. In this case, we write

$\mathbb{R}^n = W \oplus W^\perp$. In particular, for any $n \times m$ matrix A, we have $\mathbb{R}^m = \ker(A) \oplus \ker^\perp(A)$, and $\mathbb{R}^n = \mathrm{im}(A) \oplus \mathrm{im}^\perp(A)$. The Fundamental Theorem of Linear Algebra implies that the restriction of A to the $\ker^\perp(A)$ is a one-to-one map between $\ker^\perp(A)$ and $\mathrm{im}(A)$. We use this basic property many times in this paper.

A ***projection matrix*** Π is a matrix such that $\Pi^2 = \Pi$. A projection matrix is ***orthogonal*** if and only it it is symmetric, that is $\Pi^T = \Pi$. For a vector subspace W, the ***projection onto W*** is the orthogonal projection Π_W with $\mathrm{im}(\Pi_W) = W$. For any $x = x_W + x_{W^\perp}$, $\Pi_W x = x_W$.

For fixed m and n, any vector norm $||\cdot||$ in $\mathbb{R}^n$ implies a ***matrix norm*** on $n \times m$ matrices A defined as $||A|| = \sup\{||Ax||/||x|| : x \in \mathbb{R}^m\}$. In this paper, we only use 2-norm of matrices, arising from the 2-norm of vectors.

A symmetric matrix A is ***positive semidefinite*** if for all $x \in \mathbb{R}^n$, we have $x^T A x \geq 0$. The ***Loewner order*** is a partial order on $n \times n$ symmetric matrices. For two such matrices A and B, we say $A \preceq B$ if and only if $B - A$ is positive semidefinite.

The ***pseudoinverse*** of a matrix A, denoted A^+, is the unique matrix such that (i) $AA^+A = A$, (ii) $A^+AA^+ = A^+$, (iii) $(AA^+)^T = AA^+$, and (iv) $(A^+A)^T = A^+A$. One inconvenience with the algebra of the pseudoinverse matrices is that the relation $(AB)^+ = B^+A^+$ does not generally hold. However, it holds provided certain conditions; in particular, $(AA^T)^+ = (A^T)^+A^+$. Finally, we find the following standard lemma (see for example Campbell [7], Theorem 3.1.1) useful in computing the pseudo-inverse of the 1-Laplacian.

LEMMA 2.1. *For symmetric matrices A, B, if $A^TB = B^TA = 0$ then $(A+B)^+ = A^+ + B^+$.*

Simplicial Complexes A ***simplicial complex*** is a set K such that (1) each element of K is a finite set and (2) for each $\tau \in K$, if $\sigma \subset \tau$, then $\sigma \in K$. An element of K is a ***simplex***. For simplices $\sigma \subset \tau$, we say that σ is a ***face*** of τ, and τ is a ***coface*** of σ. The simplices σ and τ are ***incident***. Two d-simplices τ_1 and τ_2 are ***adjacent*** if $\tau_1 \cap \tau_2$ is a $(d-1)$-simplex. The ***vertices*** of K are the set $\cup_{\sigma \in K} \sigma$. We assume there is a fixed but arbitrary order $(v_1, \ldots, v_n)$ on the vertices of K.

A simplex $\sigma \in K$ of size $|\sigma| = d+1$ is a ***d-simplex***. A 0-simplex is a ***vertex***, a 1-simplex an ***edge***, a 2-simplex a ***triangle***, and a 3-simplex a ***tetrahedron***. The set of all d-simplices of K is denoted K_d. The ***d-skeleton*** of K is $K^d = \cup_{i=0}^d K_i$. The ***dimension*** of K is the largest d such that K contains a d-simplex; a 1-dimensional simplicial complex is a ***graph***.

A simplicial complex defines an ***underlying topological space***. We omit the exact construction for space, but it can be found in most books on algebraic topology, e.g. [17, 21]. The idea is each simplex is a topological space, and the entire space is all the simplices "glued together." Let X be a topological space. A ***triangulation*** of X is a simplicial complex T such that the underlying topological space of T is homeomorphic to X.

Homology. The ***d^{th} chain group*** $C_d(K)$ is the vector space over $\mathbb{R}$ with orthonormal basis K_d. An element of $C_d(K)$ is a ***d-chain***. The ***support*** of a d-chain c is $\mathrm{supp}(c) = \{\sigma \in K_d \mid c[\sigma] \neq 0\}$. Let $\sigma = \{v_{i_0}, \ldots, v_{i_d}\}$ be a d-simplex in K with $v_{i_j} \leq v_{i_k}$ whenever $j \leq k$. The ***boundary*** of σ is the $(d-1)$-chain $\partial\sigma = \sum_{j=0}^d (-1)^j \sigma \setminus \{v_{i_j}\}$. The ***$d^{th}$ boundary map*** is the linear map $\partial_d[K] : C_d(K) \to C_{d-1}(K)$ defined $\partial_d[K] f = \sum_{\sigma \in K_d} f(\sigma) \partial\sigma$. An element in $\ker(\partial_d[K])$ is a ***cycle***, and an element in $\mathrm{im}(\partial_d[K])$ is a ***boundary*** or a ***null-homologous cycle***. The ***d^{th} coboundary map*** is $\partial_{d+1}^T[K] : C_d(K) \to C_{d+1}(K)$. To simplify the notation, sometimes we denote the coboundary map by δ_d; so, $\delta_d = \partial_{d+1}^T$. An element of $\ker(\partial_{d+1}^T[K])$ is a ***cocycle***, and an element in $\mathrm{im}(\partial_d^T[K])$ is a ***coboundary*** or a ***null-cohomologous cocycle***.

The boundary maps have the property that $\partial_d[K] \circ \partial_{d+1}[K] = 0$, so $\mathrm{im}(\partial_{d+1}[K]) \subset \ker(\partial_d[K])$. The ***$d^{th}$ homology group*** is the quotient group $H_d(K) = \ker(\partial_d[K]) / \mathrm{im}(\partial_{d+1}[K])$. The ***$d^{th}$ Betti number*** β_d is the dimension of $H_d(K)$. The ***d^{th} cohomology group*** is the quotient group $H^d(K) = \ker(\partial_{d+1}^T[K]) / \mathrm{im}(\partial_d^T[K])$.

The ***d^{th} Laplacian*** is $L_d[K] = \partial_{d+1}[K]\partial_{d+1}^T[K] + \partial_d^T[K]\partial_d[K]$. The ***$d^{th}$ up Laplacian*** is $L_d^{up} = \partial_{d+1}[K]\partial_{d+1}^T[K]$, and the ***$d^{th}$ down Laplacian*** is $L_d^{down} = \partial_d^T[K]\partial_d[K]$.

Collapsibility A simplex $\sigma \in K$ is ***free*** if σ is the face of exactly one other simplex τ. A ***collapse*** at a free simplex σ is the removal of σ and τ, resulting in the complex $K' = K \setminus \{\sigma, \tau\}$. The complexes K and K' are homotopy equivalent. A simplicial complex K ***collapses to a subcomplex*** L if there is a nested sequence of subcomplexes $L = K(m) \subset K(m-1) \subset \cdots \subset K(0) = K$ such that each $K(i)$ is obtained from $K(i-1)$ by collapsing some free simplex σ_i; the sequence $L = K(m) \subset K(m-1) \subset \cdots \subset K(0) = K$ is a ***collapsing sequence***. A complex K is ***collapsible*** if K collapses to a single vertex. As homology is preserved by homotopy equivalence, then the homology of a collapsible complex K is the same as a single vertex; namely, $H_0(K) = \mathbb{R}$ and $H_i(K) = 0$ for $i > 0$.

Embedded Complexes and Dual Graphs Let K be a d-dimensional simplicial complex. The complex K is ***embedded*** if $K \subset R$ for a triangulation R of $\mathbb{R}^{d+1}$. Equivalently, K is embedded if $K \subset S$ for a triangulation S of $\mathbb{S}^{d+1}$, the $(d+1)$-sphere. A key property of embedded complexes is the dual graph. We first give an informal geometric definition of the dual graph to provide intuition. We then give a formal algebraic definition of the dual graph: the Lefschetz set.

The ***dual graph*** of K is defined as follows. The vertices of the dual graph are the connected components of $S \setminus K$ in the underlying space. The edges of dual graph are in one-to-one correspondence with the d-simplices of K. The endpoints of the edge corresponding to the d-simplex σ are the two (possibly the same) connected components incident to σ. In particular, the $(d+1)$-sphere $\mathbb{S}^{d+1}$ has a dual graph with vertices corresponding to its $(d+1)$-simplices.

A corollary of the Alexander Duality Theorem [21, Corollary 3.45] is that $H_d(K) \oplus \mathbb{R} \cong H_0(S \setminus K)$. The dimension of the 0th homology group counts the number of connected components of a space, so $S \setminus K$ has $\beta_d + 1$ connected components. The fact that the number of vertices of the dual graph is the number of d-cycles of K plus one suggests a connection between the dual graph and the d-cycles of K. In fact, we can alternatively define the dual graph with vertices corresponding to d-cycles of K, rather than connected components of $S^{d+1} \setminus K$. A ***Lefschetz set*** for K is a set of $\beta_d + 1$ d-cycles V such that (1) V generates $H_d(K)$, (2) each d-simplex is in the support of zero or two elements of V, and (3) if a d-simplex σ is in the support of two elements $v_1, v_2 \in V$, then $v_1(\sigma) = -v_2(\sigma) = \pm 1$. [1]

A Lefschetz set for an embedded complex K can be found from a dual graph of K by taking V to be the coboundaries of the dual vertices. Conversely, a Lefschetz set defines a graph. It is easy to verify that the matrix M_V with columns that are a Lefschetz set V is actually the 0-coboundary matrix of a graph. We call this graph a dual graph, even if the Lefschetz set didn't arise from an embedding. A dual graph arising from a Lefschetz set is not necessarily a dual graph that is defined by an embedding of K; yet, the conditions placed on the dual graph by the definition of a Lefschetz set are sufficient for our purposes. We will overload notation and use elements of the Lefschetz set to refer to dual vertices and their coboundaries.

3 Solver for boundary cycles

Let X be a $(d+1)$-dimensional collapsible simplicial complex embedded in $\mathbb{R}^{d+1}$ with a known collapsing sequence, and let K be a d-dimensional subcomplex of X. In this section, we describe an algorithm for computing an operator $U : C_{d-1}[K] \to C_d[K]$ such that for any boundary $x \in \text{im}(\partial_d[K])$, $\partial_d[K]Ux = x$. That is, U maps a $(d-1)$-dimensional boundary x in K to a d-chain in K whose boundary is x. In particular, this result implies a linear time algorithm for testing homology provided the collapsing sequence of X. For the case $d = 2$ we can compute the operator for any simplicial complex K by extending it to a collapsible simplicial complex in nearly quadratic time.

Our operator U is the composition of three operators, $U = S(X, K) \circ F(X) \circ N(K, X)$:

(i) The Include operator $N(K, X) : C_{d-1}[K] \to C_{d-1}[X]$: For a $(d-1)$-chain $x \in C_{d-1}(K)$ and for any $\sigma \in X$, $(N(K, X)x)[\sigma] = x[\sigma]$ if $\sigma \in K$ and $(N(K, X)x)[\sigma] = 0$ otherwise.

(ii) The Fill operator $F(X) : C_{d-1}[X] \to C_d[X]$: For any $(d-1)$-cycle in X, $\partial_d[X]F(X)y = y$. This is basically a boundary solver within the collapsible complex X.

(iii) The Squeeze operator $S(X, K) : C_d[X] \to C_d[K]$: Given a d-chain in X whose boundary is in K, this operator returns a d-chain in K with the same boundary if such a chain exists.

The Include operator is trivial to compute. The Squeeze operator is described in Section 3.1. The Fill operator was found by Cohen et al. [12] and is described in the following Lemma. This is the only operator of the three that requires collapsibility of X. The running time of the algorithm of Cohen et al. is linear with respect to the length of the collapsing sequence. Since, we consider embeddable X, we can bound the running time by $O(n_d)$.

[1] Lefschetz originally defined these sets in graphs; the existence of a Lefschetz set is a necessary and sufficient criterion for planarity in graphs [28]. In higher dimensions, the existence of a Lefschetz set is a necessary but not sufficient condition for embeddability. The real projective plane $\mathbb{R}P^2$ is a counter example: it can be represented by a 2-dimensional simplicial complex and is not embeddable in $\mathbb{R}^3$. The 2^{nd} homology of $\mathbb{R}P^2$ is $H_2(\mathbb{R}P^2) = 0$, so $\mathbb{R}P^2$ has a trivial Lefschetz set containing only the zero vector.

LEMMA 3.1. (COHEN ET AL. [12], THEOREM 5.2) *Let X be a $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$ that collapses into its $(d-1)$-skeleton via a known collapsing sequence. There is an $O(n_d)$ time algorithm to compute the operator $F(X) : C_{d-1}[X] \to C_d[X]$, such that for any $y \in \text{im}(\partial_d[X])$, $\partial_d[X] \cdot F(X) \cdot y = y$, where n_d is the number of d-simplices in X. Further, for any $y \in C_{d-1}[X]$, $F(X) \cdot y$ can be computed in $O(n_d)$ time.*

Proof. Cohen et al. show that the operator $F(X)$ exists and that the running time of the algorithm is linear with respect to the length of the collapsing sequence. We show that the length of the collapsing sequence is $O(n_d)$ for an embeddable complex X in $\mathbb{R}^{d+1}$.

The collapsing sequence is a sequence of nested subcomplex $X^{d-1} = X(m) \subset X(m-1) \subset \cdots \subset X(0) = X$ where each $X(i)$ is the result of an elementary collapse on $X(i-1)$. An elementary collapse is defined the removal of simplex $\sigma_i \in X(i-1) \setminus X(i)$ and one of σ_i's face. The simplices σ_i must always be d- or $(d+1)$-simplices, so the length of the collapsing sequence is $O(n_d + n_{d+1})$.

On the other hand, each $(d+1)$-simplex of X is the coface of the $(d+2)$ d-simplices in its boundary. Conversely, each d-simplex is the face of at most two $(d+1)$-simplices, as X is embeddable in $\mathbb{R}^{d+1}$. Hence, $(d+2)n_{d+1} \leq \frac{1}{2}n_d$, which implies $O(n_d + n_{d+1}) \in O(n_d)$. ∎

3.1 The Squeeze Operator Now we describe the Squeeze operator, which for a d-chain x in X with boundary in K returns a d-chain in K with the same boundary if possible. Hence, it "squeezes" x into K.

LEMMA 3.2. *Let X be a $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$ such that $H_d(X) = 0$, and let $K \subset X$. Let n_d be the total number of d-simplices of X. There is an operator $S(X, K) : C_d[X] \to C_d[K]$ such that for any $x \in C_d[X]$ with the following properties,*

(i) $\partial_d[X] \cdot x$ is zero on $X \setminus K$, and

(ii) $\partial_d[X] \cdot x$ is null-homologous in K,

we have $\partial_d[X]x = \partial_d[K]S(X, K)x$. Further, for any $y \in C_d[X]$, $S(X, K)y$ can be computed in $O(n_d)$ time.

Let $\chi = \partial_d[X] \cdot x$. In Section 3.1.1, we describe the Squeeze algorithm that tests whether χ is null-homologous in K: if it is, the algorithm computes a chain in $C_d[K]$ with boundary χ, otherwise, it rejects. In Section 3.1.2, we give bounds on the time complexity of this algorithm. In Section 3.1.3, we show that this algorithm (with one small modification) is the linear operator described in Lemma 3.2.

3.1.1 Algorithm The Squeeze algorithm uses a Lefschetz set for X^d. We prove such a set exists and is easily computable in Lemma 3.4. To prove this, we need a lemma about $(d+1)$-spheres.

LEMMA 3.3. *Let S be a simplicial complex homeomorphic to $(d+1)$-sphere. There is a $(d+1)$-cycle c on S such that $c[\tau] = \pm 1$ for each $\tau \in S_{d+1}$.*

Proof. We begin with two observations. First, the d-skeleton S^d is an embedded complex, and each d-simplex in S is the face of exactly two $(d+1)$-simplices. Second, because $H_{d+1}(S) = \mathbb{R}$, there is a non-zero $(d+1)$-cycle c' for S. We will show that $c' = kc$ for some $k \in \mathbb{R}$.

Consider a $(d+1)$-simplex $\tau \in S_{d+1}$ such that $c'[\tau] = k \neq 0$. For each face σ of τ, the value of $\partial c'[\sigma] = 0$. There is one other $(d+1)$-simplex τ' with σ as a face, so we conclude that $c[\tau'] = \pm k$ as $\partial c'[\sigma] = c'[\tau]\partial\tau[\sigma] + c[\tau']\partial\tau'[\sigma] = 0$; the sign is positive if $\partial\tau[\sigma] = -\partial\tau'[\sigma]$ and negative if $\partial\tau[\sigma] = \partial\tau'[\sigma]$

The simplices τ and τ' are neighbors in the dual graph. We can inductively show that $c(\tau'') = \pm k$ for every other $(d+1)$-simplex τ'' using a breadth-first search traversal of the dual graph. ∎

LEMMA 3.4. *Let X be an embedded complex in $\mathbb{S}^{d+1}$ with $H_d(X) = 0$. There is a Lefschetz set for the d-skeleton X^d. Moreover, this set can be computed in $O(n_d)$ time, where n_d is the number of d-simplices of X.*

Proof. Let S be a triangulation of $\mathbb{S}^{d+1}$ with $X \subset S$, and let $c \in C_{d+1}(S)$ be the $(d+1)$-cycle with the properties listed in the statement of Lemma 3.3. The set of cycles $V = \{c[\tau]\partial\tau \mid \tau \in X_{d+1}\}$ is a generating set for $H_d(X^d)$ as the homology of the entire complex is $H_d(X) = 0$, but this set has one too few cycles to be a Lefschetz set. To

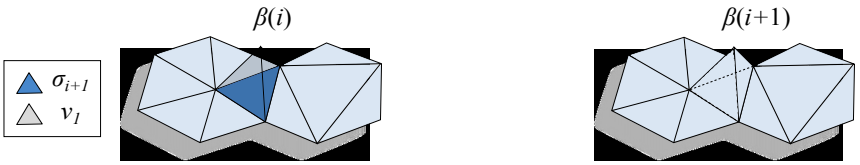


Figure 1: The Squeeze algorithm tries to replace a simplex σ_{i+1} in the chain $\beta(i)$ with another chain with boundary $\partial\sigma_{i+1}$. In this example, σ_{i+1} is in the support of a d-cycle v_1. The algorithm replaces σ_{i+1} with the rest of the cycle v_1 in $\beta(i+1)$.

remedy this, we add an additional cycle $v_\infty := \sum_{\tau \in X_{d+1}} -c[\tau]\partial\tau$. It is straightforward to verify this is a Lefschetz set.

To calculate $c[\tau]$, we first calculate $c[\tau]$ for each $(d+1)$-simplex τ using a BFS traversal of the dual graph. Start with an arbitrary $(d+1)$-simplex τ. WLOG we can assume that $c[\tau] = 1$. A neighbor of τ in the dual graph is a $(d+1)$-simplex τ' that shares d-dimensional face σ with τ. We can determine the sign of $c[\tau']$ based on the signs of $\partial\tau[\sigma]$ and $\partial\tau'[\sigma]$; the sign is positive if $\partial\tau[\sigma] = -\partial\tau'[\sigma]$ and negative if $\partial\tau[\sigma] = \partial\tau'[\sigma]$. This traversal takes $O(n_d)$, as there are $O(n_d)$ edges in the dual graph. We can then use the entries $c[\tau]$ to compute V in $O(dn_{d+1}) \in O(n_d)$ time by iterating through the boundaries of the d-simplices. ∎

Recall that we are given a d-chain x whose boundary χ is in K, and we are after a d-chain in K with the same boundary. The Squeeze algorithm iteratively removes the d-simplices $\{\sigma_1, \ldots, \sigma_m\} = X_d \setminus K_d$ in any order while maintaining a chain with boundary χ. We say $X(i) = X^d \setminus \{\sigma_1, \ldots, \sigma_i\}$ and $\beta(i)$ is a chain with boundary χ in $X(i)$.[2] Initially, we set $\beta(0) = x$. If there is no chain $\beta(i)$ in $X(i)$ that has boundary χ, our algorithm returns a failure. Otherwise, the final chain computed by our algorithm $\beta(m)$ has boundary χ in K. We also maintain a Lefschetz set of $X(i)$ called $\mathrm{V}(i)$. We can compute $\mathrm{V}(0)$ using the algorithm in the proof of Lemma 3.4.

We compute $\beta(i+1)$ from $\beta(i)$ as follows. If σ_{i+1} is contained in the support of two elements in $v_1, v_2 \in \mathrm{V}(i)$, we set $\beta(i+1) = \beta(i) - (\beta(i)[\sigma_{i+1}]v_1[\sigma_{i+1}])\, v_1$. If σ_{i+1} is not contained in the support of two elements in $\mathrm{V}(i)$ and $\sigma_{i+1} \in \operatorname{supp}\beta(i)$, we reject. If σ_{i+1} is not contained in the support of two elements in $\mathrm{V}(i)$ and $\sigma_{i+1} \notin \operatorname{supp}\beta(i)$, we set $\beta(i+1) = \beta(i)$. Intuitively, we want to replace σ_{i+1} in $\beta(i)$ with another chain with boundary $\partial\sigma_{i+1}$. If $\sigma_{i+1} \in \operatorname{supp} v_1 \in \mathrm{V}(i)$, as v_1 is a cycle, we can replace σ_{i+1} with "the rest" of v_1, namely $v_1 - v_1[\sigma_{i+1}]\sigma_{i+1}$. See Figure 1. Alternatively, if $\partial\sigma_{i+1}$ is not in the support of any elements of $\mathrm{V}(i)$, then we can show there is no chain in $X(i+1)$ with boundary $\partial\sigma_{i+1}$.

We compute $\mathrm{V}(i+1)$ from $\mathrm{V}(i)$ as follows. If there are distinct elements $v_1, v_2 \in \mathrm{V}(i)$ such that σ_{i+1} is in the support of v_1 and v_2, then we set $\mathrm{V}(i+1) = (\mathrm{V}(i) \setminus \{v_1, v_2\}) \cup \{v_1 + v_2\}$. Intuitively, the simplex σ_{i+1} is incident to two connected components of $\mathbb{R}^{d+1} \setminus X(i)$. If we remove σ_{i+1} from $X(i)$, then these two connected components are combined into one component in $\mathbb{R}^{d+1} \setminus X(i+1)$. If σ_i is in the support of no elements of $\mathrm{V}(i)$, we set $\mathrm{V}(i+1) = \mathrm{V}(i)$

To prove this algorithm is correct, we first prove the set $\mathrm{V}(i)$ is a Lefschetz set for $X(i)$.

LEMMA 3.5. *At the ith iteration of the algorithm, the set* $\mathrm{V}(i)$ *is a Lefschetz set for* $X(i)$.

Proof. It is simple to prove that each simplex $\sigma \in (X(i))_d$ is supported by zero or two elements of $\mathrm{V}(i)$, and if σ is supported by $v_1, v_2 \in \mathrm{V}(i)$, then $v_1(\sigma) = -v_2(\sigma) = \pm 1$. It is less obvious to prove that $\mathrm{V}(i)$ generates $H_d(X(i))$. We prove this by induction on the iteration.

This is true by assumption for $\mathrm{V}(0)$, so assume that $\mathrm{V}(k)$ generates $H_d(X(k))$. There are two cases: either σ_{k+1} is supported by two elements v_1, $v_2 \in \mathrm{V}(k)$, or σ_{k+1} is supported by no elements of $\mathrm{V}(k)$. We claim these two cases correspond to $H_d(X(k+1))$ having one fewer cycle than $H_d(X(k))$ and having the same number of cycles as $X(k)$ respectively.

We observe that $H_d(X(k+1)) \subset H_d(X(k))$ as $X(k+1)_d \subset X(k)_d$. As well, any cycle in $X(k)$ that is zero on σ_{k+1} is also a cycle in $X(k+1)$. We conclude that $H_d(X(k+1)) = \{\gamma \in H_d(X(k)) \mid \gamma[\sigma_{k+1}] = 0\}$.

If σ_{k+1} is not supported by any element of $\mathrm{V}(k)$, then the previous observation means $\mathrm{V}(k) \subset H_d(X(k+1))$. Thus, $H_d(X(k+1)) = H_d(X(k))$. Alternatively, suppose σ_{k+1} is supported by two elements v_1, $v_2 \in \mathrm{V}(k)$.

[2]Note that $X(0) = X^d$, but $X(m) \neq K$ in general as we do not require that the $(d-1)$-skeletons $X^{d-1} = K^{d-1}$. This is not an issue though, as a chain with boundary χ in $X(m)$ will have boundary χ in K.

All other elements of $\mathrm{V}(k) \setminus \{v_1, v_2\}$ are contained in $H_d(X(k+1))$. There is a unique 1-dimensional subspace of $\mathrm{span}\{v_1, v_2\}$ that is zero on σ_{k+1}, namely $\mathrm{span}\{v_1 + v_2\}$ as $v_1[\sigma_{k+1}] = -v_2[\sigma_{k+1}]$. Therefore, the maximal subspace of $\mathrm{V}(k)$ that is zero on σ_{k+1} is $\mathrm{span}\{(\mathrm{V}(k) \setminus \{v_1, v_2\}) \cup \{v_1 + v_2\}\} = \mathrm{span}\, \mathrm{V}(k+1)$, which means $\mathrm{V}(k+1)$ generates $H_d(X(k+1))$ ∎

We now prove that $\beta(i)$ is a chain with boundary χ.

LEMMA 3.6. *After the ith iteration of the algorithm where we don't reject, the chain $\beta(i)$ is a d-chain in $X(i)$ with boundary χ.*

Proof. We prove this by induction. This is true by assumption for $\beta(0) = x$ and $X(0) = X$, so assume this is true after iteration k. We now show it is true after iteration $k+1$ if we don't reject.

If $\sigma_{k+1} \notin \mathrm{supp}\, \beta(k)$, then we set $\beta(k+1) = \beta(k)$, so $\partial\beta(k+1) = \partial\beta(k) = \chi$ and the lemma is true. So assume that $\sigma_{k+1} \in \mathrm{supp}\, \beta(k)$. As we don't reject, then there are elements $v_1, v_2 \in \mathrm{V}(k)$ with $v_1[\sigma_{k+1}] = -v_2[\sigma_{k+1}] = \pm 1$. We set $\beta(k+1) = \beta(k) - (\beta(k)[\sigma_{k+1}]v_1[\sigma_{k+1}])v_1$. The chain $\beta(k+1)$ is a chain in $X(k+1)$ as $\beta(k+1)$ is 0 on σ_{k+1}, which we now verify:

$$\beta(k+1)[\sigma_{k+1}] = \beta(k)[\sigma_{k+1}] - (\beta(k)[\sigma_{k+1}]v_1[\sigma_{k+1}])v_1[\sigma_{k+1}] = \beta(k)[\sigma_{k+1}] - \beta(k)[\sigma_{k+1}]v_1[\sigma_{k+1}]^2 = 0$$

as $v_1[\sigma_{k+1}]^2 = (\pm 1)^2 = 1$. Moreover, $\beta(k+1)$ has boundary χ. As v_1 is a cycle by Lemma 3.5, then

$$\partial\beta(k+1) = \partial\beta(k) - (\beta(k)[\sigma_{k+1}]v_1[\sigma_{k+1}])\partial v_1 = \partial\beta(k) - 0 = \chi.$$

∎

COROLLARY 3.1. *If the algorithm doesn't reject at any iteration, the chain returned by the algorithm $\beta(m)$ has boundary χ in K.*

Corollary 3.1 tells us that if the algorithm doesn't reject, then χ is null-homologous in K. We now prove the inverse. If the algorithm does reject, then χ is not null-homologous in K.

LEMMA 3.7. *If the algorithm rejects, then χ is not null-homologous in K.*

Proof. Assume the algorithm rejects at the $(k+1)$st iteration. Because the algorithm rejects, we know that the simplex $\sigma_{k+1} \in \mathrm{supp}\, \beta(k)$. We conclude that χ is homologous to a multiple of $\partial\sigma_{k+1}$. If we can show that $\partial\sigma_{k+1}$ is not null-homologous in $X(k+1)$, this would imply that χ is not null-homologous in $X(k+1)$ either. In turn, this implies that χ is not null-homologous in K, as K has even fewer d-simplices than $X(k+1)$.

Because the algorithm rejects, we know that σ_{k+1} is not in the support of any elements of $\mathrm{V}(k)$. As $\mathrm{V}(k)$ generates $H_d(X(k))$, then we conclude that σ_{k+1} is not in the support of any d-cycle of $X(k)$.

Now suppose for the purposes of contradiction that $\partial\sigma_{k+1}$ is null-homologous in $X(k+1)$. Let $c \in C_d(X(k+1))$ be a chain with boundary $\partial c = \partial\sigma_{k+1}$. The chain $c - \sigma_{k+1}$ is a chain in $C_d(X(K))$. Moreover, $c - \sigma_{k+1}$ is a cycle in $C_d(X(k))$. Indeed,

$$\partial_d(c - \sigma_{k+1}) = \partial c - \partial\sigma_{k+1} = \partial\sigma_{k+1} - \partial\sigma_{k+1} = 0$$

The chain $c \in C_d(X(k+1))$, so $c[\sigma_{k+1}] = 0$. This implies that $\sigma_{k+1} \in \mathrm{supp}\, c - \sigma_{k+1}$, which contradicts our observation that σ_{k+1} is not in the support of any d-cycle of $X(k)$. ∎

Corollary 3.1 and Lemma 3.7 prove that the Squeeze algorithm actually works.

COROLLARY 3.2. *The Squeeze algorithm returns a chain $\beta(m)$ with boundary χ in K if such a chain exists. If no such chain exists, the Squeeze algorithm rejects.*

Dual Interpretation. The set $\mathrm{V}(i)$ is a Lefschetz set for $X(i)$, and as discussed in Section 2, a Lefschetz set is the set of coboundaries of the vertices of the dual graph. Interestingly, the updates to $\mathrm{V}(i)$ performed by our algorithm have a simple interpretation on the dual graph. Each iteration of the algorithm updates the set $\mathrm{V}(i)$ to be a Lefschetz set for $X(i)$ by adding the two elements of $v_1, v_2 \in \mathrm{V}(i)$ that are non-zero on σ_i. The dual interpretation of this update is that we are contracting the edge dual to σ_i. The new element $v_1 + v_2$ of $\mathrm{V}(i+1)$ is the coboundary of the new vertex. If a simplex σ_j other than σ_i is non-zero in both v_1 and v_2, then σ_j will be a loop on the new vertex. The boundary of a loop is 0, and accordingly, the coboundary $v_1 + v_2$ is zero on σ_j.

3.1.2 Time Complexity We can perform the Squeeze algorithm using any order of the simplices $X_d \setminus K_d$, but the running time depends on the order we choose. In particular, there is an order with the property that (almost) whenever a d-simplex σ_i is in the support of two elements of $v_1, v_2 \in \mathrm{V}(i)$, one of v_1 or v_2 will be the boundary of a $(d+1)$-simplex. In this case, updating $\mathrm{V}(i)$ and $\beta(i)$ will only take $O(d)$ time. Based on this observation, we obtain the following lemma.

LEMMA 3.8. *Let n_d be the number of d-simplices in X. The Squeeze algorithm can be performed in $O(n_d)$ time.*

Proof. Recall that the dual vertices of X are the $(d+1)$-simplices and the infinite face, and the edge of the dual graph are the d-simplices. In the dual graph of X, let F be a spanning forest of the edges dual to $X_d \setminus K_d$. We remove the d-simplices from X in the following order. Perform a depth first search from an arbitrary vertex v_r in each connected component of F. Each time we visit a new vertex v, remove the simplex σ connecting v to its parent. At the end of the traversal, if any simplex $\sigma \in (X_d \setminus K_d) \setminus F$ is still in the support of β, we reject.

The algorithm merges all of the vertices in a connected component into a single vertex. Consider a connected component T. Let $\{\sigma_1, \dots, \sigma_t\}$ be the primal d-simplices corresponding to dual edges of T in the order they are removed, and let $\{v_0, \dots, v_t\} \subset \mathrm{V}(0)$ be the vertices of T in the order they are traversed. During the first iteration of the algorithm, the vertices v_0 and v_1 are replaced in $\mathrm{V}(1)$ with $v_0 + v_1$. In general, the set of $\mathrm{V}(i) = \{\sum_{j=0}^{i} v_j, v_{i+1}, \dots, v_t\}$. At the $(i+1)$st iteration, we claim we replace $\sum_{j=0}^{i} v_j$ and V_{i+1} with $\sum_{j=0}^{i+1} v_j$. This is because the parent of v_{i+1} has already been visited, so σ_{i+1} is in the support of $\sum_{j=0}^{i} v_j$ and v_{i+1}.

After traversing T, the vertices of T will be merged into a single vertex $\sum_{j=0}^{t} v_j$. If $\sigma_{k+1} \in X_d \setminus K_d$ is a d-simplex whose dual edge is an off-tree edge in T, then σ_{k+1} will be not be in the support of any elements of $\mathrm{V}(k)$ as the sum $\sum_{i=0}^{t} v_i$ contains both endpoints of σ_k. Therefore, we need only check whether $\sigma_{k+1} \in \operatorname{supp} \beta(k)$ when we process σ_k.

We now analyze the time complexity. Note that an element of V is only changed when it is visited. Hence, when we visit a new vertex v_i, that vertex will be the same as it originally was in $\mathrm{V}(0)$. With the exception of v_∞, each v_i will be the boundary of a $(d+1)$-simplex. We update $\mathrm{V}(i)$ by adding the entries of v_1 to $\sum_{j=0}^{i} v_j$. This vector sum takes $O(d)$ time if v_i is the boundary of a $(d+1)$-simplex. If $v_i = v_\infty$, the updates to $\mathrm{V}(i)$ can take $O(n_d)$ time, but we only perform such an update once. As there are $O(n_{d+1})$ edges in our forest, we spend $O(dn_{d+1} + n_d) \in O(n_d)$ time in total updating $\mathrm{V}(i)$. We can update $\beta(i)$ in the same time.

We also need to check whether $\beta(i)$ is zero on each off-tree edge. Each check takes constant time. As there are $O(n_d)$ edges in the dual graph of X, so these checks take $O(n_d)$ time in total. Summing the time to perform updates to $\mathrm{V}(i)$ and $\beta(i)$ with the time to check $\beta(i)$, our algorithm takes $O(n_d)$ time in total. ∎

3.1.3 Linearity So far, we have presented the Squeeze algorithm as an algorithm takes a chain with boundary χ in X to a chain with boundary χ in K, if such a chain exists. Lemma 3.2 guarantees a linear operator $S(X, K) : C_d(X) \to C_d(K)$ defined on all d-chains of X. The difference between these two viewpoints is that the first algorithm will fail if χ is not null-homologous in K, while the second algorithm will return a chain in K with a boundary other than χ. We now prove that with a slight modification, the Squeeze algorithm can implement the operator $S[X, K]$, which is Lemma 3.2

Proof. [Proof of Lemma 3.2] Assume for now that the Squeeze algorithm succeeds. We first show the Squeeze algorithm is linear. We show that the $(i+1)$st iteration of the algorithm is a linear operator $S(i+1) \colon C_d(X(i)) \to C_d(X(i+1))$ that takes $S(i+1)\beta(i) = \beta(i+1)$. There are two cases for $S(i+1)$. The first case is that σ_{i+1} is in the support of two cycles $v_1, v_2 \in \mathrm{V}(i)$. In this case, we set $\beta(i+1) = \beta(i) - (\beta(i)[\sigma_{i+1}]v_1[\sigma_{i+1}])v_1$. We can represent this as the matrix

$$S(i+1) = \Pi_{C_d(X(i+1))}(I - v_1 v_1^T \Pi_{\sigma_i})$$

where Π_{σ_i} is the projection onto the σ_i and $\Pi_{C_d(X(i+1))}$ is the projection onto $C_d(X(i+1))$. The second case is that σ_{i+1} isn't in the support of any elements of $\mathrm{V}(i)$ or $\beta(i)$. In this case, we set $\beta(i+1) = \beta(i)$, so $S(i+1) = \Pi_{C_d(X(i+1))}$. We define $S(X, K) = S(m) \circ S(m-1) \circ \cdots \circ S(1)$.

While the Squeeze algorithm may fail on some inputs, the operator $S(X, K)$ is defined on all inputs as the individual operators $S(i+1)$ are only defined by σ_{i+1} and $\mathrm{V}(i)$. We can use the Squeeze algorithm to compute $S(X, K)$ in $O(n_d)$ time for any chain $x \in C_d(X)$. The only modification is that we don't fail if $\sigma_{i+1} \in \operatorname{supp} \beta(i)$ but not the support of any elements of $\mathrm{V}(i)$. Instead, we project $\beta(i)$ onto $C_d(X(i+1))$ and continue the algorithm. ∎

3.2 Summing up Now, we show that the operator $U = S(X,K)F(X)N(K,X)$ has the required properties.

LEMMA 3.9. *Let X be a $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$ that collapses into its $(d-1)$-skeleton via a known collapsing sequence. Let $K \subset X$. Let n_d be the number of d-simplices of X. There is an operator $U : C_{d-1}[K] \to C_d[K]$, such that for any $x \in \mathrm{im}(\partial_d[K])$, $\partial_d[K]Ux = x$. Further, for any $x \in C_{d-1}[K]$, Ux can be computed in $O(n_d)$ time.*

Proof. We show that $U = S(X,K)F(X)N(K,X)$ has the property of the algorithm. Let $x \in \mathrm{im}(\partial_d[K])$. Then $N(K,X) \cdot x$ is in $\mathrm{im}(\partial_d[X])$. So, by Lemma 3.1, $F(X)N(K,X) \cdot x$ is a d-chain in X with boundary x. By Lemma 3.2, $S(X,K)F(X)N(K,X)x$ is a d-chain in K with boundary x.

Applying $N(K,X)$ is equivalent to extending x to a larger vector by appending zeros. Also, $F(X)$ and $S(X,K)$ can be applied in $O(n_d)$ by Lemma 3.1 and Lemma 3.2, respectively. Thus, for any $x \in C_1[K]$, Ux can be computed in $O(n_d)$ time. ∎

We use the lemma above to obtain our up Laplacian solver. However, this lemma also implies a fast algorithm for testing the homology of cycles in K, if we have access to a collapsible super complex like X, formalized in Corollary 1.2.

3.3 Extending and Collapsing. We obtain a linear time algorithm if we have the collapsible X and its collapsing sequence. If such an X is not provided, we extend K to a collapsible supercomplex of at most quadratic complexity.

Chazelle and Palios [8] proved the following lemma for polyhedrons of genus zero. Later Chazelle and Shouraboura [9] showed that the same algorithm works for general polyhedrons holding the same bound on the number of generated tetrahedra and the running time of the algorithm. We refer the reader to Bern and Eppstein [1] for a more detailed explanation of these results and other related results.

LEMMA 3.10. (CHAZELLE, PALIOS AND SHOURABOURA [9, 8]) *Any polyhedron of complexity n, possibly with positive genus embedded in $\mathbb{R}^3$ can be triangulated with $O(n^2)$ tetrahedra in $O(n^2 \log n)$ time.*

We need to obtain a triangulation of a convex ball that includes K; a convex ball is collapsible by Lemma 3.11. To that end, we consider a tetrahedron T that contains K, and we compute a triangulation Δ_T of T that contains the given triangulation of K; in particular, edges and faces of K are not subdivided in Δ_T. Note that the algorithm of Lemma 3.10 may find a triangulation that subdivides the boundary of the input polyhedron. To keep the simplices of K intact in Δ_T, we build a polyhedron K''' that contains K in its interior. We then compute a triangulation, Δ'''_{out}, of the space bounded by $T \cup \partial K'''$ using Lemma 3.10. Note that the triangulation Δ'''_{out} may subdivide edges and faces on the boundary of K'''. Next, we compute a triangulation of K''' that is consistent with Δ'''_{out} and that does not subdivide faces and edges of K.

Our triangulation algorithm assumes each vertex and edge is incident to a triangle. If a vertex or edge is not incident to a triangle, we simply add a new triangle to K that is only incident to this vertex or this edge and its endpoints. Adding these new triangulations does not break our algorithm, as any triangulation containing this new space will also contain K.

We build K''' in three steps. The three steps will triangulate the space surrounding the triangles, edges, and vertices of K respectively.

- *Step 1: Building type I tetrahedra.* For each triangle $f = (u,v,w)$ of K, we add two tetrahedra to K, one on each side of f. We call these *type I tetrahedra*. Let c be the incenter of f, and let a be the point that is at a sufficiently small distance ε from c and f, on a given side of f; see Figure 2, left. The type I tetrahdra $\tau = (u,v,w,a)$ on the given side is the one with base f and apex a. The value ε must be small enough to ensure that τ intersects K only at its base. We call a a *type I apex*. We denote the complex obtained from adding all type I tetrahedra to K by K'. Note that in K' the interior of the triangles of K are not exposed, but the edges of K are still exposed.

- *Step 2: Building type II tetrahedra.* Let e be an edge in K, and let $(f_1, \ldots, f_k)$ be the triangles incident to e in K' in cyclic order. By the construction, we can assume that f_i belongs to K if and only if $i \equiv 1 \pmod 3$. Let f and f' be two consecutive triangles that belong to $K' \backslash K$. Let m be the midpoint of e, and

let b be the point at a sufficiently small distance ε' from m and e that is on the bisector half plane of f and f' that is between them with respect to the cyclic order mentioned above; see Figure 2, middle. For each e, f and f', we build tetrahedra, $\tau = (f, b)$ and $\tau' = (f', b)$, with apex b and bases f and f', respectively. We call these *type II tetrahedra*. The constant ε' must be small enough to ensure that τ and τ' intersect K' only at their bases. We refer to b as a type II apex. We denote the complex obtained by adding all type II tetrahedra to K' by K''. In K'', only the vertices of K are exposed, everything else is completely in the interior of K''.

- *Step 3: Removing pinch points.* The boundary of K'' is almost a surface, except possibly at the vertices of K where pinch points can exist. To remedy this issue, we add small tetrahedra around the pinched points and consider their union with K'' to obtain K'''. Provided that these tetrahedra intersect K'' only at the star of their corresponding vertices, $\partial K'''$ will be a surface.

  Let v be a pinch point in $\partial K''$, and let τ be a sufficiently small tetrahedron with center v. Note $\tau \cap K''$ is a collection of topological disks that are identified at v. Thus, $\partial\tau \cap K''$ is a set of disjoint cycles that partitions $\partial\tau$ into a set of planar regions. The interior of each of these regions is either completely inside K'' or completely outside K''. In the former case, we drop the region. In the latter case we triangulate the region and add it to K''. We refer to the triangles of these regions as *pinch point covering triangles*. To obtain K''' from K'', we apply this process to all pinch points of K''. In the end, all pinch points of K'' reside in the interior of K''', and $\partial K'''$ is a surface.

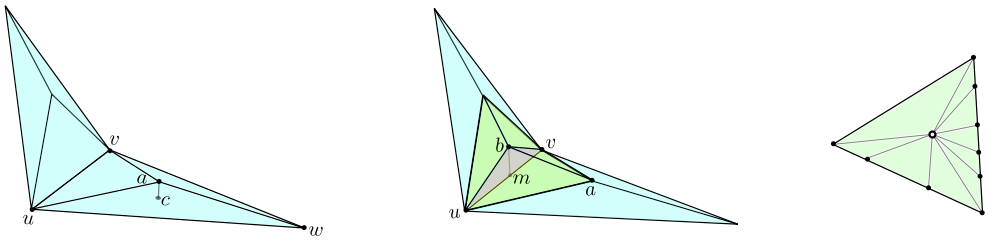


Figure 2: Left-to-right: type I tetrahedra, type II tetrahedra, two-dimensional star extension.

We can use Lemma 3.10 to compute a triangulation Δ'''_{out} of the volume bounded by $T \cup \partial K'''$. We now must make the triangulation Δ'''_{out} compatible with K.

- *Triangulating Pinch Point Covering Triangles.* We extend Δ'''_{out} to a triangulation Δ''_{out} of the volume bounded by $T \cup \partial K''$ as follows. For any pinch point v of K'', and for any of its pinch point covering triangles f we apply the following two steps. First, we extend f to a tetrahedron τ with base f and apex v. Second, we extend each triangle f' in the triangulation of f in Δ'''_{out} to a tetrahedron with base f' and apex v.

Next, we find a triangulation Δ_{in} of K'' that is consistent with Δ''_{out}. In Δ''_{out} the exposed edges and triangles of K'' are possibly subdivided. The exposed triangles of K'' are those incident to a type I and a type II apex. The exposed edges of K'' are those incident to a type I or a type II apex. We triangulate all type I and all type II tetrahedra in order to obtain Δ_{in}

In both of these steps, we find the following simple technique for extending a triangulation of a boundary of a (geometric) simplex to a triangulation of its interior helpful. Let τ be a simplex, let $b = \partial\tau$, and let Δb be a triangulation of b. We can extend Δb to a triangulation $\Delta\tau$ of τ by adding a vertex m in the middle of τ and extending each simplex σ on Δb to a simplex $\overline{\sigma}$ of one higher dimension in $\Delta\tau$ that has base σ and apex m. We call this way of extending a triangulation a *star extension*. See Figure 2, right. The crucial property of the star extension for our application is if an edge or face of τ is not subdivided in Δb, then it will not be subdivided in $\Delta\tau$. Further, the complexity of the triangulation obtained from star extension is within a constant factor of the complexity of the boundary triangulation, as each triangle or tetrahedron in $\Delta\tau$ has a base in Δb.

- *Triangulating type II tetrahedra.* Let $\tau = (u, v, a, b)$ be a type II tetrahedron, where a and b are type I and type II apices, respectively. We know that the faces (u, a, b) and (v, a, b) are possibly triangulated, and all

edges except (u, v) are possibly subdivided in Δ''_{out}. First, we extend the subdivision of the edges to the triangulation of (b, v, u) and (a, v, u) using star extension. So, we obtain a full triangulation of $\partial\tau$ that does not subdivided (u, v). Next, we use star extension to extend this triangulation to a triangulation of τ, in which again (u, v) is not subdivided.

- *Triangulating type I tetrahedra.* Let $\tau = (u, v, w, a)$ be a type I tetrahedra. The triangulation Δ''_{out} does not subdivided (u, v, w) or its edges. Triangulating type II tetrahedra preserve this property. So, after triangulating all type II tetrahedra, we have that $\partial\tau$ is fully triangulated, and the triangle (u, v, w) and its edges are not subdivided in this triangulation. Thus, we can apply star extension to obtain a triangulation of τ that does not subdivide (u, v, w)

For each face of K, we build two tetrahedra of type I, and six tetrahedra of type II. Also, in K''', we have at most six pinch point covering triangles corresponding to each triangle of K''. So, the complexity of K''' is within a constant factor of the complexity of K. Further, after star extension the complexity of the triangulation grows only by a constant factor. So, for each face (u, v, w), and its respective type I and type II tetrahedra, we know that the complexity of the final triangulation is proportional to the complexity of the Δ''_{out} on the boundary of the six type II tetrahdra, the only faces that are exposed to Δ''_{out}. We conclude that the total complexity of Δ_T is proportional to the complexity of Δ''_{out}, which is quadratic by Lemma 3.10. Therefore, we obtain the following corollary.

COROLLARY 3.3. *Let T be a tetrahedron in $\mathbb{R}^3$, and let K be any 2-simplicial complex of size n inside T. There is an algorithm to compute a triangulation of T with $O(n^2)$ tetrahedra that includes K in $O(n^2 \log n)$ time.*

Then, we can collapse the containing tetrahedron using the following lemma.

LEMMA 3.11. (CHILLINGWORTH [10, 11]) *Let X be a triangulation of a convex ball in $\mathbb{R}^3$. Then X is collapsible, and a collapsing sequence of X can be computed in linear time.*

Therefore, we obtain Corollary 3.4 by extending K into a triangulation of a tetrahedron, which is convex and so collapsible. This coroallary in turn implies a nearly quadratic time algroithm for testing homology for embedded complexes in $\mathbb{R}^3$, Corollary 3.5.

COROLLARY 3.4. *Let K be a simplicial complex embedded in $\mathbb{R}^3$. There is a $O(n^2 \log n)$ time algorithm to compute an operator $U : C_{d-1}[K] \to C_d[K]$, such that for any $x \in \mathrm{im}(\partial_d[K])$, $\partial_d[K]Ux = x$, where n is the total number of simplices in K. Further, for any $x \in C_{d-1}[K]$, Ux can be computed in the same asymptotic running time.*

COROLLARY 3.5. *Let K be a simplicial complex embedded in $\mathbb{R}^3$, there is an $O(n^2 \log n)$ time algorithm for testing whether a 1-cycle is null-homologous in K, where $n = n_0 + n_1 + n_2$ is the total number of 0-, 1- and 2-simplices in K.*

4 Laplacian Solver

In this section, we describe our solver and give a proof of Theorem 1.1, which in turn implies Corollary 1.1.

Let X be a collapsible 3-dimensional simplicial complex embedded in $\mathbb{R}^3$ with a known collapsing sequence, and let $K \subset X$ be a 2-dimensional simplicial complex such that $H_1(K) = 0$. Also, let $L_1 = L_1[K]$ be the 1-Laplacian of K in this section. We approximate $(L_1^{up})^+ = (\partial_2\partial_2^T)^+$ and $(L_1^{down})^+ = (\partial_1^T\partial_1)^+$ separately, to obtain an approximation for $(L_1)^+$ (by Lemma 2.1). Both of our approximations are based on the following lemma, whose proof is given in Section 4.1.

LEMMA 4.1. *Let B be a linear operator, let $0 \le \varepsilon < 1$, and let $\widetilde{\Pi}_{\mathrm{im}(B)}$ and $\widetilde{\Pi}_{\ker^\perp(B)}$ be symmetric matrices such that $(1-\varepsilon)\Pi_{\mathrm{im}(B)} \preceq \widetilde{\Pi}_{\mathrm{im}(B)} \preceq \Pi_{\mathrm{im}(B)}$, and $(1-\varepsilon)\Pi_{\ker^\perp(B)} \preceq \widetilde{\Pi}_{\ker^\perp(B)} \preceq \Pi_{\ker^\perp(B)}$. Also, let U be a linear map such that for any $y \in \mathrm{im}(B)$, $BUy = y$. We have,*

$$(1-(2\kappa+1)\varepsilon)(BB^T)^+ \preceq \widetilde{\Pi}_{\mathrm{im}(B)}U^T\widetilde{\Pi}_{\ker^\perp(B)}U\widetilde{\Pi}_{\mathrm{im}(B)} \preceq (1+\kappa\varepsilon)(BB^T)^+,$$

where κ is the condition number of BB^T within the image of B.

We use Lemma 4.1 with B being ∂_2 and ∂_1^T to obtain our up and down Laplacian solvers, respectively. To that end, we need approximate projection operators onto $\mathrm{im}(\partial_2)$, $\ker^{\perp}(\partial_2)$, $\mathrm{im}(\partial_1^T)$ and $\ker^{\perp}(\partial_1^T)$, all implied by Cohen et al. [12], Lemma 3.2, on K and it dual. Further, we need solvers for ∂_2 and ∂_1^T within their images. The former is described by Lemma 4.9, and the latter is described by Cohen et al. Lemma 4.2.

4.1 Proof of Lemma 4.1 We start by stating the following basic property of the Loewner partial order. (I) For two symmetric $n \times n$ matrices A and B, and any $n \times m$ matrix V,

$$A \preceq B \Rightarrow V^T A V \preceq V^T B V. \tag{4.1}$$

(II) For symmetric symmetric matrices A, B, C and D.

$$A \preceq B \text{ and } C \preceq D \Rightarrow A + B \preceq C + D. \tag{4.2}$$

In addition, we find the following lemma regarding the pseudo-inverse matrix useful in this section.

LEMMA 4.2. *Let A and B be matrices such that for any $y \in \mathrm{im}(A)$ we have $ABy = y$. Then, $A^+ = \Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}$ is the pseudo-inverse of A.*

Proof. Let $B' = \Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}$, we show by checking the four required properties that $B' = A^+$. We find the following identities useful in our arguments.

$$\Pi_{\mathrm{im}(A)} A = A \tag{4.3}$$

$$A \Pi_{\ker^\perp(A)} = A \tag{4.4}$$

Also, we find the following implications of the lemma statement useful.

$$ABA = A \tag{4.5}$$

$$AB\Pi_{\mathrm{im}(A)} = \Pi_{\mathrm{im}(A)} \tag{4.6}$$

Now, we check the required properties for B' to be a pseud-inverse.

(1) $AB'A = A$:

$$\begin{aligned} AB'A &= A\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)} A \\ &= ABA && \text{(by (4.3) and (4.4))} \\ &= A && \text{(by (4.5))} \end{aligned}$$

(2) $B'AB' = A$:

$$\begin{aligned} B'AB' &= \left(\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}\right) A \left(\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}\right) \\ &= \left(\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}\right) \left(A \Pi_{\ker_\perp(A)}\right) \left(B \Pi_{\mathrm{im}(A)}\right) \\ &= \left(\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}\right) AB\Pi_{\mathrm{im}(A)} && \text{(by (4.4))} \\ &= \left(\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)}\right) \Pi_{\mathrm{im}(A)} && \text{(by (4.6))} \\ &= \Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)} && (\Pi_{\mathrm{im}(A)} \text{ is projection}) \\ &= B' \end{aligned}$$

(3) $AB' = (AB')^T$, we show that AB' is an orthogonal projection matrix, hence symmetric.

$$\begin{aligned} AB' &= A\Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)} \\ &= AB\Pi_{\mathrm{im}(A)} && \text{(by (4.4))} \\ &= \Pi_{\mathrm{im}(A)} && \text{(by (4.6))} \end{aligned}$$

(4) $B'A = (B'A)^T$, we show that $B'A$ is an orthogonal projection, hence symmetric. By (4.6), we have $ABA = A$ by (4.5). That is, for any $x \in \mathbb{R}^m$, $A(BAx) = Ax$. Since applying A to them gives the same result, x and BAx must have the same projection into $\ker^{\perp}(A)$. In other words,

$$\Pi_{\ker_\perp} BA = \Pi_{\ker_\perp}. \tag{4.7}$$

Now, we have,

$$\begin{aligned} B'A &= \Pi_{\ker_\perp(A)} B \Pi_{\mathrm{im}(A)} A \\ &= \Pi_{\ker_\perp(A)} BA && \text{(by (4.3))} \\ &= \Pi_{\ker_\perp(A)} && \text{(by (4.7))} \end{aligned}$$

So, the proof is complete. ∎

The following lemma relates the 2-norm of matrices and the Loewner order.

LEMMA 4.3. *Let A be a symmetric matrix. We have $||A||_2 \le \alpha$ if and only if*

$$-\alpha I \preceq -\alpha \Pi_{\ker^\perp(A)} \preceq A \preceq \alpha \Pi_{\ker^\perp(A)} \preceq \alpha I$$

Proof. Let x be any vector in the domain of A. Let $x = x_{\ker} + x_{\ker^\perp}$ be the unique decomposition of x such that $x_{\ker} \in \ker(A)$ and $x_{\ker^\perp} \in \ker^{\perp}(A)$. Since $||A||_2 \le \alpha$, we have

$$-\alpha x_{\ker^\perp}^T \Pi_{\ker^\perp(A)} x_{\ker^\perp} = -\alpha x_{\ker^\perp}^T x_{\ker^\perp} \le x_{\ker^\perp}^T A x_{\ker^\perp} \le \alpha x_{\ker^\perp}^T x_{\ker^\perp} = \alpha x_{\ker^\perp}^T \Pi_{\ker^\perp(A)} x_{\ker^\perp}.$$

On the other hand,

$$-\alpha x_{\ker}^T \Pi_{\ker(A)} x_{\ker} = x_{\ker}^T A x_{\ker} = \alpha x_{\ker}^T \Pi_{\ker^\perp(A)} x_{\ker} = 0.$$

By adding the last two sequences of inequalities, we obtain $-\alpha x^T \Pi_{\ker(A)} x \le x^T A x \le x^T \Pi_{\ker^\perp(A)} x$, or equivalently, $-\alpha \Pi_{\ker(A)} \preceq A \preceq \alpha \Pi_{\ker^\perp(A)}$.

Finally, since $\Pi_{\ker^\perp(A)}$ is an orthogonal projection, we have $-x^T x \le x^T \Pi_{\ker^\perp(A)} x \le x^T x$, for any x, or equivalently, $-I \le \Pi_{\ker^\perp(A)} \le I$, which completes the proof. ∎

The following lemma is very similar to Cohen et al. [12], Lemma 4.5.

LEMMA 4.4. *Let A be a symmetric matrix, and let Π be an orthogonal projection such that* $\mathrm{im}(\Pi) \subset \mathrm{im}(A)$*, and let $\widetilde{\Pi}$ be a linear operator such that $(1-\varepsilon)\Pi \preceq \widetilde{\Pi} \preceq \Pi$. Then, we have*

$$(1 - 2\varepsilon\kappa^{\Pi}(A))\Pi A \Pi \preceq \widetilde{\Pi} A \widetilde{\Pi} \preceq (1 + 2\varepsilon\kappa^{\Pi}(A))\Pi A \Pi,$$

where $\kappa^{\Pi}(A)$ is the condition number of A restricted to the subspace of $\mathrm{im}(\Pi)$*.*

Proof. Since Π is a projection matrix, and so positive semidefinite, we have $0 \preceq \Pi \preceq I$. Together with the assumption of the lemma, we obtain

$$0 \preceq (1-\varepsilon)\Pi \preceq \widetilde{\Pi} \preceq \Pi \preceq I.$$

In particular, $\mathrm{im}(\widetilde{\Pi}) \subset \mathrm{im}(\Pi)$. Further, by Lemma 4.3,

$$||\Pi||_2 \le 1, ||\widetilde{\Pi}||_2 \le 1, ||\widetilde{\Pi} - \Pi||_2 \le \varepsilon \tag{4.8}$$

Now, we bound $||\Pi A \Pi - \widetilde{\Pi} A \widetilde{\Pi}||_2$

$$\begin{aligned} ||\Pi A \Pi - \widetilde{\Pi} A \widetilde{\Pi}||_2 &= ||\Pi A \Pi - \Pi A \widetilde{\Pi} + \Pi A \widetilde{\Pi} - \widetilde{\Pi} A \widetilde{\Pi}||_2 \\ &\le ||\Pi A \Pi - \Pi A \widetilde{\Pi}||_2 + ||\Pi A \widetilde{\Pi} - \widetilde{\Pi} A \widetilde{\Pi}||_2 && \text{(triangle inequality)} \\ &= ||\Pi A - \Pi A_{|\,\mathrm{im}(\Pi)} (\Pi - \widetilde{\Pi})||_2 + ||(\Pi - \widetilde{\Pi}) A_{|\,\mathrm{im}(\Pi)} \widetilde{\Pi}||_2 && (*) \\ &\le ||\Pi||_2 ||A_{|\,\mathrm{im}(\Pi)}||_2 ||(\Pi - \widetilde{\Pi})||_2 + ||\Pi - \widetilde{\Pi}||_2 ||A_{|\,\mathrm{im}(\Pi)}||_2 ||\widetilde{\Pi}||_2 && (**) \\ &\le 2\varepsilon ||A_{|\,\mathrm{im}(\Pi)}||_2 && \text{(Equation 4.8)} \\ &\le 2\varepsilon \lambda_{\max}^{\Pi}(A) \end{aligned}$$

Line (**) holds because for any two matrices A and B where AB is defined, we have $||AB|| \leq ||A|| \cdot ||B||$. In the calculation above, by $A_{|\,\mathrm{im}(\Pi)}$ we mean the linear map that acts as A on the $\mathrm{im}(\Pi)$ and as the zero map on $\mathrm{im}^{\perp}(\Pi)$. We can replace A with $A_{|\,\mathrm{im}(\Pi)}$ in (*) as $\mathrm{im}(\widetilde{\Pi}) \subset \mathrm{im}(\Pi)$, and, so, $\mathrm{im}(\Pi - \widetilde{\Pi}) \subset \mathrm{im}(\Pi)$.

Then, by Lemma 4.3, we have

$$-2\varepsilon\lambda^{\Pi}_{\max}(A)\Pi \preceq \Pi A\Pi - \widetilde{\Pi}A\widetilde{\Pi} \preceq 2\varepsilon\lambda^{\Pi}_{\max}(A)\Pi,$$

as $\Pi A\Pi - \widetilde{\Pi}A\widetilde{\Pi}$ is zero in $\mathrm{im}^{\perp}(\Pi)$.

By Property 4.1, the fact that $\mathrm{im}(\Pi) \subset \mathrm{im}(A)$, and that $I \preceq A/\lambda^{\Pi}_{\min}(A)$ in the image of Π, we have,

$$\Pi = \Pi I\Pi \preceq \frac{\Pi A\Pi}{\lambda^{\Pi}_{\min}(A)}$$

Since $\kappa^{\Pi}(A) = \lambda^{\Pi}_{\max}/\lambda^{\Pi}_{\min}(A)$, we conclude

$$\begin{aligned} -2\varepsilon\kappa^{\Pi}(A)\Pi A\Pi \preceq \Pi A\Pi - \widetilde{\Pi}A\widetilde{\Pi} \preceq 2\varepsilon\kappa^{\Pi}(A)\Pi A\Pi \Rightarrow \\ (1 - 2\varepsilon\kappa^{\Pi}(A))\Pi A\Pi \preceq \widetilde{\Pi}A\widetilde{\Pi} \preceq (1 + 2\varepsilon\kappa^{\Pi}(A))\Pi A\Pi, \end{aligned}$$

which completes the proof. ∎

We need one more auxiliary lemma before the main proof of this section.

LEMMA 4.5. *Let U be any linear operator, Q a symmetric linear operator, and Π an orthogonal projection such that $\Pi U^T QU\Pi$ is defined. Also, let $\widetilde{\Pi}$ and $\widetilde{Q}$ be symmetric matrices such that $(1-\varepsilon)Q \preceq \widetilde{Q} \preceq Q$ and $(1-\varepsilon)\Pi \preceq \widetilde{\Pi} \preceq \Pi$. We have:*

$$(1 - (2\kappa + 1)\varepsilon)\Pi U^T QU\Pi \preceq \widetilde{\Pi}U^T\widetilde{Q}U\widetilde{\Pi} \preceq (1 + 2\kappa\varepsilon)\Pi U^T QU\Pi,$$

where κ is the condition number of $U^T QU$ restricted to $\mathrm{im}(\Pi)$.

Proof. Since $(1-\varepsilon)Q \preceq \widetilde{Q} \preceq Q$, and $(U\widetilde{\Pi})^T = \widetilde{\Pi}U^T$, by Property 4.1 we have,

$$(1-\varepsilon)\widetilde{\Pi}U^T QU\widetilde{\Pi} \preceq \widetilde{\Pi}U^T\widetilde{Q}U\widetilde{\Pi} \preceq \widetilde{\Pi}U^T QU\widetilde{\Pi}$$

Then, by Lemma 4.4, we have,

$$(1 - 2\kappa\varepsilon)\Pi U^T QU\Pi \preceq \widetilde{\Pi}U^T QU\widetilde{\Pi} \preceq (1 + 2\kappa\varepsilon)\Pi U^T QU\Pi.$$

where κ is the condition number of $U^T QU$ in the subspace of boundary cycles, $\mathrm{im}(\Pi)$.

Putting everything together, we have,

$$(1 - (2\kappa + 1)\varepsilon)\Pi U^T QU\Pi \preceq (1-\varepsilon)(1 - 2\kappa\varepsilon)\Pi U^T QU\Pi \preceq \widetilde{\Pi}U^T\widetilde{Q}U\widetilde{\Pi} \preceq (1 + 2\kappa\varepsilon)\Pi U^T QU\Pi,$$

as desired. ∎

Now, we are ready to present our proof.

Proof. [Proof of Lemma 4.1] Lemma 4.2, we have that $B^+ = \Pi_{\ker^{\perp}(B)}U\Pi_{\mathrm{im}(B)}$, therefore,

$$(BB^T)^+ = (\Pi_{\ker^{\perp}(B)}U\Pi_{\mathrm{im}(B)})^T(\Pi_{\ker^{\perp}(B)}U\Pi_{\mathrm{im}(B)}) = \Pi_{\mathrm{im}(B)}U^T\Pi_{\ker^{\perp}(B)}U\Pi_{\mathrm{im}(B)}.$$

The last equality holds as $\Pi_{\mathrm{im}(B)}$ and $\Pi_{\ker^{\perp}(B)}$ are orthogonal projection matrices. By Lemma 4.5 we have the orders in the lemma statement with κ being the condition number of $U^T\Pi_{\ker^{\perp}(B)}$ within $\mathrm{im}(B)$, which is equal to the condition number of $(BB^T)^+$ within $\mathrm{im}(B)$, which is the condition number of BB^T within $\mathrm{im}(B)$. ∎

4.2 Laplacian Solver, Detailed Proof Our Laplacian up and down solvers are based on Lemma 4.1. To that end, we need approximate projection operators onto $\mathrm{im}(\partial_2)$, $\ker^{\perp}(\partial_2)$, $\mathrm{im}(\partial_1^T)$ and $\ker^{\perp}(\partial_1^T)$, as well as solvers for ∂_2 and ∂_1^T within their images. The following lemma in K and its dual provides the needed projection operators.

LEMMA 4.6. (COHEN ET AL. [12], LEMMA 3.2) *Let K be a simplical complex with $n = n_1 + n_0$ total number of edges and vertices, and $\varepsilon > 0$. In $\tilde{O}(n \log^2(n/\varepsilon))$, we can compute symmetric matrices $\widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\varepsilon)$ and $\widetilde{\Pi}_{\ker(\partial_1)}(\varepsilon)$ such that,*

$$(1-\varepsilon)\Pi_{\mathrm{im}(\partial_1^T)} \preceq \widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\varepsilon) \preceq \Pi_{\mathrm{im}(\partial_1^T)} \tag{4.9}$$

$$(1-\varepsilon)\Pi_{\ker(\partial_1)} \preceq \widetilde{\Pi}_{\ker(\partial_1)}(\varepsilon) \preceq \Pi_{\ker(\partial_1)} \tag{4.10}$$

Moreover, for any 1-chain x, $\widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\varepsilon) \cdot x$ and $\widetilde{\Pi}_{\ker(\partial_1)}(\varepsilon)x$ can be computed in the same asymptotic running time.

Furthermore, Cohen et al. describe an exact solver for ∂_1^T that we use to approximate $(L_1^{down})^+$.

LEMMA 4.7. (COHEN ET AL. [12], LEMMA 4.2) *For a simplicial complex K, there is a map $U : C_1(K) \to C_0(K)$ such that $\partial_1^T U x = x$, for any $x \in \mathrm{im}(\partial_1^T)$. Further, for any $y \in C_1$, Uy can be computed in $O(n_1 + n_0)$ time, where n_1 and n_0 are the number of edges and vertices of K.*

LEMMA 4.8. (DOWN LAPLACIAN SOLVER) *For a simplicial complex K and $\varepsilon > 0$, there is a map $DownLaplacianSolver(\varepsilon)$ such that*

$$(1-\varepsilon)(L_1^{down}[K])^+ \preceq DownLapSolver(K,\varepsilon) \preceq (L_1^{down}[K])^+.$$

Further, for $x \in C_1$, $DownLapSolver(K,\varepsilon) \cdot x$ can be computed in $\tilde{O}(n \log^2(n/\varepsilon))$ time, where κ is the condition number of $L_1^{down}[K]$ within the space of coboundary cycles, and n is the total number of 0- and 1-simplices in X.

Proof. Recall $L_1^{down}[K] = (\partial_1[K])^T \partial_1[K]$. By Lemma 4.7, we can compute a map $U : C_1[K] \to C_0[K]$ such that for any $y \in \mathrm{im}(\partial_1^T[K])$, we have $\partial_1^T[K] U y = y$. We define,

$$DownLapSolver(K,\varepsilon) = \widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\delta) U^T \Pi_{\ker^{\perp}(\partial_1^T)} U \widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\delta),$$

for $\delta = \varepsilon/(2\kappa+1)$. Suppose, that the 1-skeleton of K is connected, the proof is similar for other cases. Note that $\Pi_{\ker^{\perp}(\partial_1^T)}$ is projection into the space orthogonal to the all 1 vector, so $\Pi_{\mathrm{im}(\partial_1)} = I - J_n/n$, where J_n is the all one $n \times n$ matrix. In addition, we use Lemma 4.6 to approximate $\widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\delta)$. Finally, we use Lemma 4.1 to obtain the statement of the lemma.

$$\begin{aligned}(1-\varepsilon)(L_1^{down}[K])^+ &= (1-(2\kappa+1)\delta)(L_1^{down}[K])^+ \\ &\preceq DownLapSolver(K,\varepsilon) \\ &\preceq (1+2\kappa\delta)(L_1^{down}[K])^+ \preceq (1+\varepsilon)(L_1^{down}[K])^+.\end{aligned}$$

By Lemma 4.6, $\widetilde{\Pi}_{\mathrm{im}(\partial_1^T)}(\delta)$ can be applied to a vector in $\tilde{O}(n \log^2(n\kappa/\varepsilon))$ time. By Lemma 4.7, U can be applied to a vector in $O(n_1)$ time. In addition, $\Pi_{\mathrm{im}(\partial_1)} x$ can be computed for any $x \in C_0$ in $O(n)$ time by computing the average and subtract it from all elements of x. Finally, κ is the condition number of $L_1^{down}[K] = \partial_1^T \partial_1$ within the space of coboundary chains, which is $\lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum non-zero eigenvalues of $\partial_1^T \partial_1$. These are equal to the maximum and minimum non-zero eigenvalues of $\partial_1 \partial_1^T$, the graph Laplacian, which are known to be polynomially bounded from below and above with respect to the size of the graph (see Lemma 2.3 and Theorem 6.5 of Zhang's survey paper [33]). Hence, κ is polynomially bounded and $\log(\kappa n/\varepsilon) = O(\log(n/\varepsilon))$. So, we can compute $DownLapSolver(K,\varepsilon) \cdot x$ in $\tilde{O}(n \log^2(n/\varepsilon))$. ∎

Finally, we use the exact solver of Lemma 4.9 of the previous section for ∂_2 to approximate $(L_1^{up})^+$.

LEMMA 4.9. (UP LAPLACIAN SOLVER) *Let X be a collapsible simlplicial complex embedded in $\mathbb{R}^3$, and let $K \subset X$ such that $H_1(K) = 0$. For any $\varepsilon > 0$, there is a map $UpLaplacianSolver(\varepsilon)$ such that*

$$(1-\varepsilon)(L_1^{up}[K])^+ \preceq UpLaplacianSolver(X, K, \varepsilon) \preceq (1+\varepsilon)(L_1^{up}[K])^+.$$

Further, for any $x \in C_1$, $UpLaplacianSolver(X, K, \varepsilon) \cdot x$ can be computed in $O(n \log^2(n\kappa/\varepsilon))$ time, where κ is the condition number of $L_1^{up}[K]$ within the space of boundary cycles, and n is the total number of 0-, 1-, and 2-simplices in X.

Proof. Recall $L_1^{up}[K] = (\partial_2[K])\partial_2[K]^T$. By Lemma 3.9, we can compute a map $U : C_1[K] \to C_2[K]$ such that for any $y \in \mathrm{im}(\partial_2[K])$, we have $\partial_2[K]Uy = y$. We define,

$$UpLapSolver(K, \varepsilon) = \widetilde{\Pi}_{\mathrm{im}(\partial_2)}(\delta)U^T\widetilde{\Pi}_{\ker^\perp(\partial_2)}(\delta)U\widetilde{\Pi}_{\mathrm{im}(\partial_2)}(\delta),$$

for $\delta = \varepsilon/(2\kappa+1)$. Since $H_1(K) = 0$, we have $\mathrm{im}(\partial_2) = \ker(\partial_1)$, thus, we use Lemma 4.6 to approximate $\widetilde{\Pi}_{\mathrm{im}(\partial_2)}(\delta)$. In addition, we have $\ker(\partial_2)$ is the set of 2-cycles that is generated by the Lefschetz set. Thus, it is isomorphic to the set of coboundary 1-chains in the dual K^*, $\mathrm{im}(\partial_1^T[K^*])$. Therefore, $\ker^\perp(\partial_2[K])$ is isomorphic to $\ker(\partial_1[K^*])$. It follows that we can use Lemma 4.6 on K^* to approximate $\widetilde{\Pi}_{\ker^\perp(\partial_2)}$. Then, by Lemma 4.1, we obtain the desired statement.

$$\begin{aligned}(1-\varepsilon)(L_1^{up}[K])^+ &= (1-(2\kappa+1)\delta)(L_1^{up}[K])^+ \\ &\preceq UpLapSolver(K, \varepsilon) \\ &\preceq (1+2\kappa\delta)(L_1^{up}[K])^+ \preceq (1+\varepsilon)(L_1^{up}[K])^+.\end{aligned}$$

By Lemma 4.6, $\widetilde{\Pi}_{\ker^\perp(\partial_2)}(\delta)$ and $\widetilde{\Pi}_{\mathrm{im}(\partial_2)}(\delta)$ can be applied to a vector in $\tilde{O}(n\log(n\kappa/\varepsilon))$ time. By Lemma 3.9, U can be applied to a vector in $O(n_2 + n_1)$ time. So, we can compute $UpLapSolver(K, \varepsilon) \cdot x$ in the required time bound. ∎

Proof. [**Proof of Theorem 1.1.**] We now have all the ingredients to prove our main theorem of this section that shows that the 1-Laplacian of $K \subset X$ can be approximated in nearly linear time. Let

$$LaplacianSolver(X, K, \varepsilon) = UpLaplacianSolver(X, K, \varepsilon) + DownLaplacianSolver(K, \varepsilon).$$

By Lemma 2.1, Lemma 4.9 and Lemma 4.8, we have that $LaplacianSolver(X, K, \varepsilon)$ has the desired bounds. The running time requirement for applying this operator is implied by Lemma 4.9 and Lemma 4.8. ∎

5 Solver for Coboundary Cocycles

Let X be a collapsible $(d+1)$-dimensional simplicial complex embedded in $\mathbb{R}^{d+1}$ with a known collapsing sequence, and let K be a d-dimensional subcomplex of X. In this section, we describe an algorithm for computing an operator $V : C_d[K] \to C_{d-1}[K]$ such that for any coboundary $y \in \mathrm{im}(\partial_d^T[K])$, $\partial_d^T[K]Vy = y$. Readers might recognize the similarity between the operator V and the operator U from Section 3; in fact, V is the transpose of U.

The operator V is composed of three operators, $V = P(X, K) \circ F^T(X) \circ E(K, X)$.

(i) The Expand Operator $E(K, X) : C_d[K] \to C_d[X]$: Given a d-coboundary f_K in K, find a d-coboundary f_X in X such that $f_K[\sigma] = f_X[\sigma]$ for each d-simplex $\sigma \in K_d$.

(ii) The Cofill Operator $F^T(X) : C_d[X] \to C_{d-1}[X]$: Given a d-coboundary y of X, then $\partial_d^T F^T(X)y = y$. Here, $F(X)$ is the Fill operator from Section 3.

(iii) The Project Operator $P(X, K) : C_{d-1}[X] \to C_{d-1}[K]$: Given a $(d-1)$-chain $x \in C_{d-1}[K]$, then $P(X, K)x[\sigma] = x[\sigma]$ for each $(d-1)$-simplex $\sigma \in K_{d-1}$.

Compare the operator $V = P(X, K) \circ C(X) \circ E(K, X)$ with the operator $U = S(X, K) \circ F(X) \circ N(K, X)$ from Section 3. We claim that the operators V and U are transpose. The operators $N(K, X)$ and $P(X, K)$ are transpose as $N(K, X)$ is the inclusion of $C_{d-1}(K)$ into $C_{d-1}(X)$ and $P(X, K)$ is the projection of $C_{d-1}(X)$ into $C_{d-1}(K)$. We prove in the next section that $E(K, X)$ and $S(X, K)$ are transpose (Lemma 5.5.) The operators $F^T(X)$ and $F(X)$ are transpose, but it is not immediate that $F^T(X)$ has the property listed above. Cohen et al. prove that $F^T(X)$ has this property. They also give the running time of computing $F^T(X)$. We summarize these results in Lemma 5.1.

LEMMA 5.1. (COHEN ET. AL. [12], THEOREM 5.2) *Let X be a $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$ that collapses into its $(d-1)$-skeleton via a known collapsing sequence. There is an $O(n_d)$ time algorithm to compute the operator $F^T(X) : C_d[X] \to C_{d-1}[X]$, such that for any $y \in \text{im}(\partial_d^T[X])$, $\partial_d^T[X] \cdot F^T(X) \cdot y = y$. Further, for any $y \in C_d[X]$, $F^T(X) \cdot y$ can be computed in $O(n_d)$ time.*

Proof. Cohen et al. show that $F^T(X)$ has the property given in the lemma statement. They also show that $F^T(X)$ can be computed in $O(m)$ time, where m is the length of the collapsing sequence. In the proof of Lemma 3.1, we show that $m \in O(n_d)$ ∎

5.1 The Expand Operator Consider the following problem. We are given a d-coboundary f_K in K. We want to find a d-coboundary f_X in X that equals f_K on simplices in K. We present an algorithm for solving this problem: the Expand operator. The Expand operator can be thought of as being the the reverse of the Squeeze operator from Section 3.1. While the Squeeze operator would iteratively remove d-simplices in $K \setminus X$ and adjust the value of f_K, the Expand operator will iteratively add d-simplices in $K \setminus X$ and adjust the value of f_K.

We begin by proving that a chain f_X actually exists.

LEMMA 5.2. *Let $f_K \in C_d(K)$ be a coboundary. There is a coboundary $f_X \in C_d(X)$ such that $f_X[\sigma] = f_K[\sigma]$ for all $\sigma \in K_d$.*

Proof. Let $v_K \in C_{d-1}(K)$ such that $\partial_d^T[K]v_K = f_K$. Consider the chain $v_X \in C_{d-1}(X)$ such that $v_X[\tau] = v_K[\tau]$ for each $\tau \in K_{d-1}$, and let $f_X = \partial_d^T[X]v_X$ Let $\sigma \in K_d$. Then

$$f_X[\sigma] = (\partial_d^T[X]v_X)[\sigma] = v_X^T(\partial_d[X]\sigma) = v_K^T(\partial_d[K]\sigma) = f_K[\sigma]$$

as $\partial\sigma \in C_{d-1}(K)$. ∎

Even though we know f_X exists, it is not obvious how we can compute f_X efficiently. The following lemma gives a sufficient condition for finding f_X.

LEMMA 5.3. *Let $f_X \in C_d(X)$. The chain f_X is a coboundary iff $f_X \perp \ker \partial_d[X]$.*

Proof. The rank-nullity theorem states that for any matrix A, $\text{im}\, A \perp \ker A^T$. The lemma follows as the space of coboundaries is $\text{im}\, \partial_d^T[X]$, and $\text{im}\, \partial_d^T[X] \perp \ker \partial_d[X]$ ∎

We give an algorithm that converts f_K into a chain $f_X \in C_d(X)$ that ensures that $f_X \perp \ker \partial_d[X]$. We will add in the d-simplices in the reverse order they were removed in Section 3.1, i.e. add σ_m, then σ_{m-1}, and so on. Recall that the complex $X(i) = X^d \setminus \{\sigma_1, ..., \sigma_i\}$ and $\text{V}(i)$ is a Lefschetz set for $X(i)$. Recall also that $\text{V}(i)$ is a generating set for $\ker \partial_d[X(i)]$. We will compute a d-chain $f(i)$ that is a coboundary in $X(i)$. The idea of the algorithm is that each time we add a simplex σ_i, we set the value of $f(i-1)[\sigma_i]$ so that $f(i-1)$ is orthogonal to all d-cycles in $X(i-1)$, i.e. $f(i-1)$ is orthogonal to all elements of $V(i-1)$. Initially, $f(0) = f_K$.

We compute $f(i-1)$ from $f(i)$ in the following way. If $\text{V}(i-1) = \text{V}(i)$, we set $f(i-1) = f(i)$. If $\text{V}(i) = (\text{V}(i-1) \setminus \{v_1, v_2\}) \cup \{v_1 + v_2\}$ for two d-cycles v_1 and v_2, we set $f(i-1)[\sigma_i] = -v_1[\sigma_i]v_1^T f(i)$ and $f(i-1)[\sigma] = f(i)[\sigma]$ for all other $\sigma \in X(i)_d$. Lemma 25 proves that these updates work and that $f(i)$ is a coboundary in $X(i)$

LEMMA 5.4. *The chain $f(i)$ is a coboundary in $X(i)$.*

Proof. Lemma 5.3 states that $f(i)$ is a coboundary in $X(i-1)$ iff $f(i)$ is orthogonal to $\ker \partial_d[X(i)]$. Recall that $V(i)$ is a generating set for $\ker \partial_d[X(i)]$. We prove that $f(i)$ is orthogonal to each element of $\text{V}(i)$ by induction on the iteration i. As $f(0)$ is a coboundary in K, then $f(0)$ is orthogonal to each element of $\text{V}(0)$ by Lemma 5.3.

Now assume that $f(i)$ is orthogonal to each element of $V(i)$. If $\text{V}(i-1) = \text{V}(i)$, then we set $f(i-1) = f(i)$. As $\text{V}(i-1) = \text{V}(i)$, then $f(i-1)$ is orthogonal to all elements of $\text{V}(i-1)$ by assumption.

Assume instead that $\text{V}(i) = (\text{V}(i-1) \setminus \{v_1, v_2\}) \cup \{v_1 + v_2\}$; then $\text{V}(i-1) = (\text{V}(i) \cup \{v_1, v_2\}) \setminus \{v_1 + v_2\}$. We claim the chain $f(i-1)$ is orthogonal to all chains in $\text{V}(i-1)$ except v_1 and v_2; indeed, any other chain $v \in \text{V}(i-1) \setminus \{v_1, v_2\}$ is zero on σ, so $f(i-1)^T v = f(i)^T v = 0$ by the inductive hypothesis. We now show that $f(i-1)$ is orthogonal to v_1 and v_2.

We have that $f(i-1)^T v_1$ is

$$\begin{aligned} f(i-1)^T v_1 =& f(i)^T v_1 + f(i-1)[\sigma_i] \cdot v_1[\sigma_i] \\ =& f(i)^T v_1 - (v_1[\sigma_i])^2 f(i)^T v_1 \\ =& f(i)^T v_1 - f(i)^T v_1 \\ =& 0 \end{aligned}$$

as $(v_1[\sigma_i])^2 = (\pm 1)^2 = 1$. Before we calculate $f(i-1)^T v_2$, we make two observations. First, as $f(i-1)^T(v_1+v_2) = 0$ because $(v_1 + v_2) \in \mathrm{V}(i)$, then $f(i-1)^T v_1 = -f(i-1)^T v_2$. Second, as $v_1[\sigma_i] = -v_2[\sigma_i] = \pm 1$, then $v_1[\sigma_i] \cdot v_2[\sigma_i] = -1$. We have that $f(i-1)^T v_1$ is

$$\begin{aligned} f(i-1)^T v_2 =& f(i)^T v_2 + f(i-1)[\sigma_i] \cdot v_2[\sigma_i] \\ =& f(i)^T v_2 - v_1[\sigma_i] \cdot v_2[\sigma_i] \cdot f(i)^T v_1 \\ =& f(i)^T v_2 - f(i)^T v_2 \\ =& 0. \end{aligned}$$

□

The Expand and Squeeze operators appear to be opposites to one another. The Expand operator iteratively add the simplices $X_d \setminus K_d$ to the complex while updating the a d-chain f, while the Squeeze operator iteratively removes the simplices $X_d \setminus K_d$ from the complex while updating a d-chain β. In fact, the Expand operator and the Squeeze operator are transpose linear operators.

LEMMA 5.5. *The Expand operator $E(K, X) : C_d(K) \to C_d(X)$ is the transpose linear operator to the Squeeze operator $S(X, K) : C_d(X) \to C_d(K)$.*

Proof. We can express each step of the Expand operator as a linear operator $E(i) : C_d(X(i)) \to C_d(X(i-1))$. If $\mathrm{V}(i-1) = \mathrm{V}(i)$, then $E(i) = i_{C_d(X(i-1))}$, the inclusion operator of $C_d(X(i))$ into $C_d(X(i-1))$. If $\mathrm{V}(i) = (\mathrm{V}(i-1) \setminus \{v_1, v_2\}) \cup \{v_1 + v_2\}$, then we can express $E(i) = (I - \Pi_{\sigma_i} v_1 v_1^T) i_{C_d(X(i-1))}$. As $i_{C_d(X(i-1))} = \Pi^T_{C_d(X(i))}$, we see that $E(i) = S(i)^T$, the linear operator from the ith step of the Squeeze Algorithm. Therefore,

$$\begin{aligned} E(K, X) =& E(m) \circ \cdots \circ E(1) \\ =& S(m)^T \circ \cdots \circ S(1)^T \\ =& S(X, K)^T. \end{aligned}$$

□

We are now ready to prove the main lemma of the section, Lemma 5.6.

LEMMA 5.6. *Let X be a $(d+1)$-simplicial complex embedded in $\mathbb{R}^{d+1}$ such that $H_d(X) = 0$, and let $K \subset X$. Let n_d be the total number of d-simplices of X. There is an operator $E(K, X) : C_d[K] \to C_d[X]$ such that for any $f \in \operatorname{im} \partial_d^T[K]$, the chain $E(K, X)f$ has the following properties,*

(i) $E(K, X)f \in \operatorname{im} \partial_d^T[X]$

(ii) $E(K, X)f[\sigma] = f[\sigma]$ *for* $\sigma \in K_d$.

Further, for any $f \in C_d[K]$, $E(K, X)f$ can be computed in $O(n_d)$ time.

Proof. Let $f_K = f(m)$ and $f_X = f(0)$. Lemma 5.4 proves that f_X is a coboundary in X, so Property (i) is true. Moreover, the Expand operator does not change the value of σ in $f(i)$ for any $\sigma \in K_d$, so Property (ii) is satisfied. Finally, as the Expand operator is the transpose of the Squeeze operator by Lemma 5.5, then the two operators have the same running time, which is $O(n_d)$ by Lemma 3.8. □

5.2 Summing up Now, we prove the main lemma of this section.

LEMMA 5.7. *Let X be a $(d+1)$-dimensional simplicial complex that collapses into its $(d-1)$-skeleton via a known collapsing sequence. Let $K \subset X$. Let n_d be the number of d-simplices of X. There is an operator $V : C_d[K] \to C_{d-1}[K]$ such that for any $y \in \mathrm{im}(\partial_d^T[K])$, $\partial_d^T[K]Vy = y$. Further, for any $y \in C_{d-1}[K]$, Vy can be computed in $O(n_d)$.*

Proof. We show that $V = P(X,K) \circ F^T(X) \circ E(K,X)$ has the property of the algorithm. Let $x \in \mathrm{im}(\partial_{d-1}^T[K])$. Then $E(K,X) \cdot x$ is in $\mathrm{im}(\partial_{d-1}^T[X])$. So, by Lemma 5.1, $F^T(X) \circ E(K,X) \cdot x$ is a $(d-1)$-chain in X with coboundary x. But, the value assigned by $\partial_d^T[X] \circ F^T(X) \circ E(K,X) \cdot x$ to any d-simplex in K is only determined by the value of $F^T(X) \circ E(K,X) \cdot x$ on $(d-1)$-simplices inside K. It follows that $P(X,K) \circ F^T(X) \circ E(K,X) \cdot x$ is a $(d-1)$-chain in K with coboundary x.

Applying $P(X,K)$ takes $O(n_d)$ time, as we just need to restrict a $(d-1)$-chain in X into its part in K. Also, $F^T(X)$ and $E(X,K)$ can be applied in $O(n_d)$ by Lemma 5.1 and Lemma 5.6, respectively. Thus, for any $x \in C_d[K]$, Vx can be computed in $O(n_d)$ time. ∎

6 Computing a Cohomology Basis

Let X be a collapsible $(d+1)$-dimensional simplicial complex embedded in $\mathbb{R}^{d+1}$ with a known collapsing sequence, and let K be a d-dimensional subcomplex of X. A ***(d-1)-homology basis*** for K is a set of β_{d-1} linearly independent $(d-1)$-cycles B such that no linear combination of cycles in B is in $\mathrm{im}\,\partial_d$. A ***(d-1)-cohomology basis*** for K is a set of β_{d-1} linearly independent cocycles B such that no linear combination of B is in $\mathrm{im}\,\partial_{d-1}^T$. Cohomology bases are of interest for their use in ***homology annotation*** [6]. If $\{r_1, \ldots, r_{\beta_{d-1}}\}$ is a $(d-1)$-cohomology basis, then for any two cycles h_1 and h_2, the vectors $(r_1^T h_1, \ldots, r_{\beta_{d-1}}^T h_1) = (r_1^T h_2, \ldots, r_{\beta_{d-1}}^T h_2)$ if and only if h_1 and h_2 are homologous.

In this section, we give an algorithm to convert a homology basis of K to a cohomology basis of K in $O(\beta_{d-1} n_d)$ time, where n_d is the number of simplices in K. Combined with a result of Dey [15] to compute a homology basis of K in $O(n \log n)$ time, we can compute a cohomology basis for K in $O(n \log n + \beta_{d-1} n_d)$ time, if $d = 2$.

6.1 Computing a Homology Basis in $\mathbb{R}^3$ Dey showed that a homology basis over $\mathbb{Z}_2$ coefficients for a 2-complex K embedded in $\mathbb{R}^3$ can be computed in $O(n \log n + k)$ time, where k is the length of the set of cycles [15]. It is implied by the universal coefficient theorem and the fact that K is embedded in $\mathbb{R}^3$ (so $H_1(K,\mathbb{Z})$ is torsion-free) that this basis is also a basis for $\mathbb{R}$-homology; a more careful argument of this fact follows.

A cycle $\gamma \in C_1(K,\mathbb{Z}_2)$ is an Eulerian subgraph of an undirected graph, so we can represent γ as a closed walk along the edges $\gamma = (e_0, e_1, \ldots, e_k)$. To obtain a cycle in $C_1(K,\mathbb{Z})$ we view γ as a closed walk along the edges in a directed graph where we can traverse along an edge in either the forwards or backwards direction. We define the ***lift*** of γ to be $\hat{\gamma} = \sum_{i=1}^k \alpha_i e_i$ where $\alpha_i = 1$ if e_i was taken in the forwards direction, $\alpha_i = -1$ if e_i was taken in the backwards direction, and $\alpha_i = 0$ if e_i was not included in the walk. Given a set of linearly independent cycles $\{\gamma_i\}$ over $\mathbb{Z}_2$ it is straightforward to show that their lifts $\{\hat{\gamma}_i\}$ over $\mathbb{Z}$ are also linearly independent [24, Lemma 2.4], so we can lift a cycle basis over $\mathbb{Z}_2$ to a cycle basis over $\mathbb{Z}$. Constructing the lift takes $O(n\beta_1)$ time where β_1 is the dimension of $H_1(K,\mathbb{Z})$.

Let H be a homology basis for $H_1(K,\mathbb{Z}_2)$. We can extend H to a cycle basis by adding the set of cycles B, where B is the set of boundaries of each triangle in K. The resulting set $H \cup B$ is a cycle basis for $C_1(K,\mathbb{Z}_2)$. By the argument in the preceding paragraph we know that $H \cup B$ can be lifted to a cycle basis $\hat{H} \cup \hat{B}$ for $C_1(K,\mathbb{Z})$. A cycle basis contains a homology basis, and we know that the lifted cycles in $\hat{B}$ are the boundaries of triangles in K. It follows that the set of lifted cycles $\hat{H}$ contains a basis for $H_1(K,\mathbb{Z})$. We assume that K admits an embedding into $\mathbb{R}^3$, which implies that $H_1(K,\mathbb{Z})$ is torsion-free [21, Corollary 3.46]. Since $H_1(K,\mathbb{Z})$ is torsion-free the universal coefficient theorem implies that the dimensions of $H_1(K,\mathbb{Z})$ and $H_1(K,\mathbb{Z}_2)$ are equal, so $\hat{H}$ is indeed a basis for $H_1(K,\mathbb{Z})$.

We now show that $\hat{H}$ is also a basis for $H_1(K,\mathbb{R})$. The elements of $H_1(K,\mathbb{R})$ are circulations, and every circulation can be decomposed into a sum $\sum_{i=1}^n \alpha_i \gamma_i$ where $\alpha_i \in \mathbb{R}$ and where γ_i is a simple cycle; that is, each γ_i is an element of $H_1(K,\mathbb{Z})$ such that the coefficients on γ_i are contained in $\{-1, 0, 1\}$. Hence, it follows that we can write any circulation in $H_1(K,\mathbb{R})$ as a linear combination of the cycles in $\hat{H}$. We summarize this section with the following lemma.

THEOREM 6.1. (DEY [15]) *For a two dimensional simplicial complex K in $\mathbb{R}^3$ there exists an algorithm computing a basis for $H_1(K, \mathbb{Z}_2)$ and a basis for $H_1(K, \mathbb{R})$ in $O(n \log n + n\beta_1)$ time, where n is the complexity of K.*

6.2 An Intermediate Subcomplex Our algorithm for computing a cohomology basis uses an intermediate space T where $K \subset T \subset X$. In this section, we describe the complex T, some of properties T, and how to find T.

We will construct T by removing certain d-simplices in $X_d \setminus K_d$. We need to distinguish between two types of d-simplices in $X_d \setminus K_d$. Let $Y \subset X$ be any subcomplex of X, and let $V(Y)$ be a Lefschetz set for Y. A ***constructive simplex*** in Y is a simplex $\sigma \in Y_d$ such that σ in the support of two elements of $\mathrm{V}(Y)$. A ***destructive simplex*** in Y is a simplex $\sigma \in Y_d$ such that σ is in the support of no elements of $\mathrm{V}(Y)$. Lemma 6.1 gives a characterization of constructive and destructive simplices.

LEMMA 6.1. *If σ is a constructive simplex of Y, then $H_{d-1}(Y) = H_{d-1}(Y \setminus \{\sigma\})$ and $H_d(Y) \cong H_d(Y \setminus \{\sigma\}) \oplus \mathbb{R}$. If σ is a destructive simplex of Y, then $H_{d-1}(Y) \oplus \mathbb{R} \cong H_{d-1}(Y \setminus \{\sigma\})$ and $H_d(Y) \cong H_d(Y \setminus \{\sigma\})$*

Proof. If σ is constructive simplex, then $v_1[\sigma] = 1$ for some cycle $v_1 \in \mathrm{V}(Y)$. Therefore, $\partial\sigma$ is linearly dependent on $\operatorname{im} \partial_d[Y]$, and $\operatorname{im} \partial_d[Y] = \operatorname{im} \partial_d[Y \setminus \{\sigma\}]$. The spaces $\ker \partial_{d-1}[Y] = \ker \partial_{d-1}[Y \setminus \{\sigma\}]$ as $X(i)$ and $X(i-1)$ have the same $(d-1)$-simplices, so removing σ does not change the $(d-1)$-homology group. Likewise, removing σ decreases the rank of $\ker \partial_d$ by 1, so removing σ decreases the rank of the d-homology group by 1.

If σ is a destructive simplex, then σ is orthogonal to $\ker \partial_d[Y]$ as σ is orthogonal to each element of $\mathrm{V}(Y)$. Therefore, removing σ does not decrease the rank of the kernel of ∂_d, and we conclude that $H_d(Y) = H_d(Y \setminus \{\sigma\})$. Likewise, as σ is orthogonal to $\ker \partial_d[Y]$, then $\partial_d\sigma$ is linearly independent from $\operatorname{im} \partial_d[Y \setminus \{\sigma\}]$, so removing σ decrease the rank of the image of ∂_d by 1. The space $\ker \partial_{d-1}[Y] = \ker \partial_{d-1}[Y \setminus \{\sigma\}]$ as Y and $Y \setminus \{\sigma\}$ have the same $(d-1)$-simplices, so removing σ increases the rank of the $(d-1)$-homology group by 1. ∎

The complex T is defined to be a subcomplex of X^d such that (1) $T^{d-1} = X^{d-1}$ (2) $H_{d-1}(T) = 0$, and (3) all simplices in $T_d \setminus K_d$ are destructive. Intuitively, the complex T contains "just enough" extra simplices to fill all the $(d-1)$-cycles in K.

We can use the following simple algorithm to find T: while there is a constructive d-simplex $\sigma \in X_d \setminus K_d$ in the complex, remove it.[3]The resulting complex T is guaranteed to only have destructive simplices from $X_d \setminus T_d$. Moreover, as removing a constructive simplex leaves the $(d-1)^{\text{st}}$ homology group unchanged, then $H_{d-1}(T) = H_{d-1}(X) = 0$.

We conclude this section with a useful property of destructive simplices.

LEMMA 6.2. *Let T be a d-dimensional simplicial complex, and let σ be a destructive d-simplex in T. Then $\sigma \in \operatorname{im} \partial_d^T[T]$.*

Proof. As $H_{d-1}[T] \oplus \mathbb{R} = H_{d-1}[T \setminus \{\sigma\}]$, there is a vector $p \in C_{d-1}$ such that $\partial_d^T[T \setminus \{\sigma\}] \cdot p = 0$ but $\partial_d^T[T] \cdot p \neq 0$. We conclude $\partial_d^T[T] \cdot p$ is non-zero only on σ. ∎

6.3 Algorithm Let $U : C_{d-1}(T) \to C_d(T)$ and $V : C_d(T) \to C_{d-1}(T)$ be the operators guaranteed by Lemmas 3.9 and 5.7 respectively. Consider the operator

$$C = N(K,T)^T \circ V \circ \Pi_{C_d(K)^\perp} \circ U \circ N(K,T),$$

where $N(K,T) : C_{d-1}(K) \to C_{d-1}(T)$ is the inclusion operator and $\Pi_{C_d(K)^\perp} : C_d(T) \to C_d(T)$ is the projection onto the orthogonal complement of $C_d(K)$, which is exactly the space spanned by the simplices $T_d \setminus K_d$. We use the operator C to convert a homology basis to a cohomology basis, thereby obtaining Lemma 1.1. Before we prove Lemma 1.1, we walk through each operator of C to get some idea of what this operator is doing.

- Let γ be a $(d-1)$ cycle in K. The inclusion $N(K,T)\gamma$ just includes γ into T, which amounts to adding zeros to the vector γ. As $H_{d-1}(T) = 0$, then $\gamma \in \operatorname{im} \partial_d[T]$.

[3]Observe that the complex T is not unique and is dependent on the order we remove d-simplices from X. Indeed, a simplex may be destructive in one subcomplex of X and constructive in another. The non-uniqueness of T is not a problem, though. We only need some subcomplex with the properties given in the definition of T, and the above algorithm is guaranteed to find such a complex.

- The operator U returns a d-chain x in T with $\partial_d[T]x = \gamma$.
- The operator $\Pi_{C_d(K)^\perp}$ projects x to its coordinates in $T_d \setminus K_d$. Let $c = \Pi_{C_d(K)^\perp} x$. We have the following facts.
  (1) The $(d-1)$-chain $\partial c \in C_{d-1}(K)$. We can orthogonally decompose the chain $x = c + c^\perp$, where we define $c^\perp = \Pi_{C_d(K)} x$. As $\partial x = \gamma$ and $\partial c^\perp$ are both in $C_{d-1}(K)$, we conclude that ∂c is in $C_{d-1}(K)$ as well.
  (2) The $(d-1)$-cycles ∂c and γ are homologous in K. This is because $\gamma = \partial c + \partial c^\perp$ where $c^\perp$ is a d-chain in K.
  (3) The d-chain c is a coboundary in T. Recall that each simplex in $T_d \setminus K_d$ is destructive. We make the following observation about destructive simplices. The chain c is a coboundary by Lemma 6.2 as it is the linear combination of destructive simplices.
- Finally, let $p = Vc \in C_{d-1}(T)$ and $r = N(K,T)^T Vc \in C_{d-1}(K)$. We need two more facts for the main proof of this section.
  (4) The chain p has the property that $\partial_d^T[T]p = c$. This follows from Lemma 5.7 as $c \in \operatorname{im} \partial_d^T[K]$.
  (5) The coboundaries $\partial_d^T[T]p[\sigma] = \partial_d^T[T]r[\sigma]$ for each $\sigma \in K_d$. In particular, this implies that $\partial_d^T[K]r = 0$ as the coboundary $\partial_d^T[T]p = c$ is 0 on all d-simplices in K.
  (6) For any simplex $\sigma \in K_{d-1}$, $p[\sigma] = r[\sigma]$. This follows as r is the projection of p onto $C_{d-1}(K)$

We are now ready to prove the main result of this section, Lemma 1.1.

Proof. [Proof of Lemma 1.1] We need to verify that the set $\{C\gamma_1, \ldots, C\gamma_{\beta_{\beta_{d-1}}}\}$ is a cohomology basis assuming $\{\gamma_1, \ldots, \gamma_{\beta_{d-1}}\}$ is a homology basis. We know that each chain $C\gamma_i$ is a cocycle as $\partial_d^T[K] \circ C \cdot \gamma_i = 0$ by Fact (5) above, so we only need to show that no linear combination of $\{C\gamma_1, \ldots, C\gamma_{\beta_{\beta_{d-1}}}\}$ is a coboundary.

Suppose $\{C\gamma_1 \ldots, C\gamma_{\beta_{d-1}}\}$ is not a cohomology basis; that is, suppose there is a non-zero linear combination $\sum_{i=1}^{\beta_{d-1}} a_i C\gamma_i$ that is a coboundary of K. We first define a few vectors associated with each γ_i. Let $c_i = \Pi_{C_d(T)^\perp} \circ U \circ N(K,T) \cdot \gamma_i$, and let $p_i = Vc_i$. Observe that

$$
\begin{aligned}
\text{(By (4))} \qquad \left(\sum_{i=1}^{\beta_{d-1}} a_i c_i\right)^T \left(\sum_{i=1}^{\beta_{d-1}} a_i c_i\right) &= \left(\sum_{i=1}^{\beta_{d-1}} a_i p_i\right)^T \partial_d[T] \left(\sum_{i=1}^{\beta_{d-1}} a_i c_i\right) \\
&= \left(\sum_{i=1}^{\beta_{d-1}} a_i p_i\right)^T \left(\sum_{i=1}^{\beta_{d-1}} a_i \partial_d[T] c_i\right) \\
(*) \qquad &= \left(\sum_{i=1}^{\beta_{d-1}} a_i C\gamma_i\right)^T \left(\sum_{i=1}^{\beta_{d-1}} a_i \partial_d[T] c_i\right) \\
(**) \qquad &= 0,
\end{aligned}
$$

The line (*) follows as $\sum_{i=1}^{\beta_{d-1}} a_i \partial_{d-1}[T] c_i$ is contained in $C_{d-1}(K)$ by Fact (1), and p_i restricted to $C_{d-1}(K)$ is $C\gamma_i$ by Fact (6). The line (**) follows from the fact that $\sum_{i=1}^{\beta_{d-1}} a_i C\gamma_i \in \operatorname{im} \partial_{d-1}^T[K]$ by assumption, and $\sum_{i=1}^{\beta_{d-1}} a_i \partial_d[T] c_i \in \ker \partial_{d-1}[K]$ as all boundaries are cycles.

As $\sum_{i=1}^{\beta_{d-1}} a_i c_i = 0$, this implies $\sum_{i=1}^{\beta_{d-1}} a_i \partial c_i = 0$. However, this is contradiction. Each ∂c_i is homologous to γ_i in K, so $\sum_{i=1}^{\beta_{d-1}} a_i \partial c_i = 0$ implies that $\sum_{i=1}^{\beta_{d-1}} a_i \gamma_i$ is a boundary. This cannot be the case, as no linear combination of $\{\gamma_1, \ldots, \gamma_{\beta_{d-1}}\}$ is a boundary as $\{\gamma_1, \ldots, \gamma_{\beta_{d-1}}\}$ is a homology basis. Therefore, $\{C\gamma_1, \ldots, C\gamma_{\beta_{d-1}}\}$ is a cohomology basis.

We now analyze the time complexity of implementing C. The operator C is the composition $N(K,T)^T \circ V \circ \Pi_{C_d(T)^\perp} \circ U \circ N(K,T)$. The operators $N(K,T)$, $N(K,T)^T$, and $\Pi_{C_d(T)^\perp}$ add or drop zeros from a vector and can be performed in $O(n_d)$ time. The operator U and V can be performed in $O(n_d)$ time be Lemmas 3.9 and 5.7 respectively. Therefore, the entire operator C can be performed in $O(n_d)$ time in total. ∎

Acknowledgements

We would like to thank Tamal Dey for his helpful comments about Theorem 6.1.

References

[1] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. In *Lecture Notes Series on Computing*, pages 23–90. World Scientific, sep 1992.

[2] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 27(4):930–951, December 2005.

[3] Erik G. Boman, Doron Chen, Bruce Hendrickson, and Sivan Toledo. Maximum-weight-basis preconditioners. *Numerical Linear Algebra with Applications*, 11(8-9):695–721, 2004.

[4] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, March 2003.

[5] Erik G. Boman, Bruce Hendrickson, and Stephen Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *SIAM J. Numer. Anal.*, 46(6):3264–3284, October 2008.

[6] Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In Fedor V. Fomin and Petteri Kaski, editors, *Algorithm Theory – SWAT 2012*, pages 189–200, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] S. L. (Stephen La Vern) Campbell. Surveys and reference works in mathematics. Pitman, London ;, 1979.

[8] Bernard Chazelle and Leonidas Palios. Triangulating a nonconvex polytope. *Discrete Comput. Geom.*, 5(5):505–526, December 1990.

[9] Bernard Chazelle and Nadia Shouraboura. Bounds on the size of tetrahedralizations. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, SCG '94, page 231–239, New York, NY, USA, 1994. Association for Computing Machinery.

[10] D. R. J. Chillingworth. Collapsing three-dimensional convex polyhedra. *Mathematical Proceedings of the Cambridge Philosophical Society*, 63(2):353–357, 1967.

[11] D. R. J. Chillingworth. Collapsing three-dimensional convex polyhedra: correction. *Mathematical Proceedings of the Cambridge Philosophical Society*, 88(2):307–310, 1980.

[12] Michael B. Cohen, Brittany Terese Fasy, Gary L. Miller, Amir Nayyeri, Richard Peng, and Noel Walkington. Solving 1-laplacians in nearly linear time: Collapsing and expanding a topological ball. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, page 204–216, USA, 2014. Society for Industrial and Applied Mathematics.

[13] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving sdd linear systems in nearly mlog1/2n time. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 343–352, New York, NY, USA, 2014. Association for Computing Machinery.

[14] Cecil Jose A. Delfinado and Herbert Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes. In *Proceedings of the ninth annual symposium on Computational geometry*, SoCG '93, pages 232–239, New York, NY, USA, 1993. ACM.

[15] Tamal K. Dey. *Computing Height Persistence and Homology Generators in* $\mathbb{R}^3$ *Efficiently*, pages 2649–2662.

[16] Tamal K. Dey and Sumanta Guha. Computing homology groups of simplicial complexes in r3. *J. ACM*, 45(2):266–287, March 1998.

[17] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction.* American Mathematical Society, 2010.

[18] Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1166–1176. SIAM, 2011.

[19] Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1038–1046, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[20] Joel Friedman. Computing betti numbers via combinatorial laplacians. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 386–391, New York, NY, USA, 1996. ACM.

[21] Allen Hatcher. *Algebraic topology.* Cambridge Univ. Press, Cambridge, 2000.

[22] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, USA, 2nd edition, 2012.

[23] Arun Jambulapati and Aaron Sidford. *Ultrasparse Ultrasparsifiers and Faster Laplacian System Solvers*, pages 540–559.

[24] Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Survey: Cycle bases in graphs characterization, algorithms, complexity, and applications. *Comput. Sci. Rev.*, 3(4):199–243, November 2009.

[25] Jonathan A. Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 911–920, New York, NY, USA, 2013. ACM.

[26] Ioannis Koutis and Richard Miller, Gary L. and 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.

[27] Ioannis Koutis and Richard Miller, Gary L. and 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.

[28] Solomon Lefschetz. Planar graphs and related topics. *Proceedings of the National Academy of Sciences of the United States of America*, 54(6):1763–1765, 1965.

[29] 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.

[30] John Stillwell. *Classical Topology and Combinatorial Group Theory*, volume 72 of *Graduate Texts in Mathematics*. Springer, second edition, 1993.

[31] 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.

[32] N.K. Vishnoi. *Lx=b: Laplacian Solvers and Their Algorithmic Applications*. Foundations and trends in theoretical computer science. Now Publishers, 2013.

[33] Xiaodong Zhang. The Laplacian eigenvalues of graphs: a survey. *arXiv: Combinatorics*, 2011.