On the Incremental Computation of Simplicial Homology of Triangulated Surfaces TJHSST Senior Research Project Computer Systems Lab 2009-2010

Brian Hamrick

May 28, 2010

Abstract

Homology groups are a fundamental algebraic invariant in algebraic topology that allows the discrimination of topological spaces. Methods to compute the homology groups of a simplicial complex are known in general, but they require expensive matrix computations. For specific classes of simplicial complexes, both geometric and incremental methods are known. However, these classes generally preclude the presense of torsion in the homology groups. This paper investigates the possibility of an incremental homology type computation that can account for torsion and thus apply to a larger class of simplicial complexes than previous results.

Keywords: algebraic topology, homology, algorithmic complexity

1 Introduction

Topology is the study of geometric structure on sets. One of the big questions of topology is to determine when are two spaces essentially the same. Algebraic topology is a branch of topology which computes algebraic objects which remain invariant when the underlying structure remains the same. These algebraic invariants include homology groups which encapsulate certain characteristics about the underlying structure of the space. In this paper I consider the homology groups of a certain class of spaces known as simplicial complexes. Algorithms to compute these homology groups are known, but they are generally inefficient. This project will investigate the efficiency of various algorithms to compute these homology groups. Improving the efficiency of such algorithms is applicable in experimental mathematics, where computers can perform computations beyond the reach of human work.

2 Background

2.1 Previous Results

The computation of simplicial homology via reduction of the boundary matrices to Smith normal form is rather classic. For sparse matrices, [3] gives an efficient algorithm to compute the homology groups. This algorithm works in full generality; it will compute the homology of any simplicial complex regardless of factors such as orientability. However, the algorithm presented in [3] is probabilistic and the worst case analysis of the algorithm is unsatisfactory. [2] describes a method by which simplicial complexes embedded in $\mathbb{R}^3$ may be transformed into a homotopically equivalent three-dimensional manifold, for which the homology groups may be computed efficiently via classical results of the structure of such spaces. This method allows deterministic computation of the homology type of a simplicial complex embedded in $\mathbb{R}^3$ in linear time, even faster than the probabilistic methods proposed by [3] for general simplicial complexes. In this paper we investigate an incremental method that provides a compromise between the algebraic and geometric methods in time, simplicity, and flexibility.

2.2 Definitions

In this paper, we deal solely with triangulated spaces, so a general notion of a topological space is unnecessary. A k-simplex is the convex hull of k points in general position. We denote the k-simplex determined by $v_1, v_2, \ldots, v_k$ as $[v_1 v_2 \cdots v_k]$. A k-chain is a formal sum of k-simplices. The orientation of a k-simplex is taken into account in the k-chain such that $[v_1 v_2 \cdots v_k] = -[v_2 v_1 \cdots v_k]$, and similarly for any other transposition of two vertices. The boundary of a k-simplex $[v_1 v_2 \cdots v_k]$ is the $(k-1)$-chain $\sum_{i=1}^{k} (-1)^{i-1} [v_1 \cdots v_{i-1} \hat{v_i} v_{i+1} \cdots v_k]$, where $[v_1 \cdots v_{i-1} \hat{v_i} v_{i+1} \cdots v_k]$ is the $(k-1)$-simplex determined by all the vertices except for v_i. The boundary of a k-chain is the sum of the boundaries of the k-simplices that form the chain.

Definition 1. *The chain groups C_k of a simplicial complex are the free abelian groups generated by the k-simplices with addition as the group operation.*

Definition 2. *The boundary homomorphism $\partial_k : C_k \to C_{k-1}$ is defined by mapping each k-chain to its boundary.*

Definition 3. *The cycle groups Z_k of a simplicial complex are defined as the set of k-chains with a null boundary, so $Z_k = Ker\ \partial_k$.*

Definition 4. *The boundary groups B_k of a simplicial complex are defined as $B_k = Im\ \partial_{k+1}$.*

3 Preliminary Results

3.1 The Abstract Incremental Algorithm

In this section, we will establish an abstract algorithm for incrementally computing the homology type of a simplicial complex. We will work with a given simplicial complex $K =$

$\bigcup_{i=1}^{N} \sigma_i$, where σ_i ranges over all the simplexes of K such that $K_i = \bigcup_{j=1}^{i} \sigma_j$ is a valid simplicial complex for all i. In this incremental algorithm, we will compute the homology type of K_i for all i, starting from $i = 1$. The homology type of K_1 is trivial to compute: $H_n(K_1) = 0$ for all n.

Next, we need to analyze the incremental step. We wish to compute the homology type of $K_i = K_{i-1} \cup \sigma_i$ from the homology type of K_{i-1}. To do this, we will appeal to the following theorem.

Theorem 1 (The Mayer-Vietoris Sequence)**.** *Given two subspaces A and B of X whose interiors cover X, the following sequence is exact:*

$$\cdots \to H_{n+1}(X) \xrightarrow{\partial_*} H_n(A \cap B) \xrightarrow{(i_*, j_*)} H_n(A) \oplus H_n(B) \xrightarrow{k_* - l_*} H_n(X) \xrightarrow{\partial_*}$$
$$\xrightarrow{\partial_*} H_{n-1}(A \cap B) \to \cdots \to H_0(A) \oplus H_0(B) \xrightarrow{k_* - l_*} H_0(X) \to 0$$

Given two spaces A and B which cover a space X, the Mayer-Vietoris Sequence relates the homology groups of A, B, $A \cap B$, and X. We'll apply this to our simplicial complex with $A = K_{i-1}$, $B = \sigma_i$, and $X = K_i$. The Mayer-Vietoris Sequence then tells us that the sequence

$$H_n(K_{i-1} \cap \sigma_i) \to H_n(K_{i-1}) \oplus H_n(\sigma_i) \to H_n(K_i) \to H_{n-1}(K_{i-1} \cap \sigma_i) \to H_{n-1}(K_{i-1})$$

is exact for all n.

However, this relation can be simplified significantly. $K_{i-1} \cap \sigma_i = \partial\sigma_i$, the boundary of σ_i, as K_i must be a valid simplicial complex, so the simplices forming $\partial\sigma_i$ must be in K_i, but they are also not σ_i, so they were in K_{i-1} as well while the interior of σ_i was not. Furthermore, σ_i is homotopy equivalent to a single point, so $H_n(\sigma_i) = 0$ for all n. Using these simplifications, we arrive at the following result.

Corollary 1. *Given two simplicial complexes K_{i-1} and K_i and a simplex σ_i such that $K_i = K_{i-1} \cup \sigma_i$, the sequence*

$$H_n(\partial\sigma_i) \to H_n(K_{i-1}) \to H_n(K_i) \to H_{n-1}(\partial\sigma_i) \to H_{n-1}(K_{i-1})$$

is exact for all n.

Let us now consider three cases. If σ_i is a k-simplex, we consider $n = k - 1$, $n = k$, and $n \neq k - 1, k$ separately.

First, consider $n \neq k - 1, k$. Then we have $H_n(\partial\sigma_i) = H_{n-1}(\partial\sigma_i) = 0$. Therefore, corollary 1 tells us that $0 \to H_n(K_{i-1}) \to H_n(K_i) \to 0$ is exact, and it follows that $H_n(K_{i-1}) \simeq H_n(K_i)$. Since this map from $H_n(K_{i-1}) \to H_n(K_i)$ is induced by the inclusion map $\iota : K_{i-1} \to K_i$, the generating set for $H_n(K_i)$ is the same as the generating set for $H_n(K_{i-1})$.

Second, consider $n = k - 1$. Then we have $H_n(\partial\sigma_i) \simeq \mathbb{Z}$ and $H_{n-1}(\partial\sigma_i) = 0$. Then corollary 1 yields that $H_{k-1}(\partial\sigma_i) \xrightarrow{\iota_*} H_{k-1}(K_{i-1}) \to H_{k-1}(K_i) \to 0$ is exact, so $H_{k-1}(K_i) = H_{k-1}(K_{i-1}) / \iota_*(H_{k-1}(\partial\sigma_i))$, where ι_* is the map induced by the inclusion $\iota : \partial\sigma_i \hookrightarrow K_{i-1}$.

Finally, consider $n = k$. Then we have $H_n(\partial\sigma_i) = 0$ and $H_{n-1}(\partial\sigma_i) \simeq \mathbb{Z}$. Then corollary 1 yields that the sequence $0 \to H_k(K_{i-1}) \to H_k(K_i) \xrightarrow{\partial_*} \mathbb{Z} \xrightarrow{\iota_*} H_{k-1}(K_{i-1})$ is exact. $\partial_*(H_k(K_i))$ is a subgroup of $\mathbb{Z}$, all of which are isomorphic to $\mathbb{Z}$. Let x be an n-cycle that contains σ_i such that the coefficient of σ_i is minimal but positive. Suppose that the coefficient of σ_i in x is a. Then for any k-cycle y, the coefficient of σ_i will be a multiple of a, or else we can subtract an appropriate multiple of x to find a k-cycle with a smaller coefficient of σ_i. Therefore, we can write any k-cycle as $cx + y$, where x is our specific cycle containing σ_i and y is a k-cycle class in K_{i-1}. Then the homoology class of this k-cycle is simply $c[x] + [y]$ where $[x]$ is a new generator and $[y]$ is a homology class in $H_k(K_{i-1})$. All the requisite propeties of the addition operations can be verified to see that $H_k(K_i) \simeq \mathbb{Z} \times H_k(K_{i-1})$ where the new generator is our specific k-cycle x containing σ_i the minimum positive number of times.

Putting these together, we have the following tasks that need to be completed to create a bona fide algorithm for incrementally computing the homology type and generating set of a complex:

- Write the homology class of $\partial\sigma_i$ as a linear combination of the generating set for $H_{k-1}(K_{i-1})$.
- Compute the quotient of $H_{k-1}(K_{i-1})$ by the subgroup generated by the homology class of $\partial\sigma_i$.
- Detect whether a new k-simplex is part of a k-cycle.
- If it exists, compute a k-cycle with the minimal positive coefficient of σ_i.

3.2 The One Dimensional Algorithm

Let us first consider the simple case of a one dimensional finite simplicial complex. The result that we will expect from this algorithm is a number telling us the number of connected components and the number of independent cycles in the complex, as well as a list of those components and independent cycles.

Because our simplex is one dimensional, $H_n(K_i) = 0$ for all $n \geq 2$ and all i. Furthermore, we only need to consider two operations: adding a 0-simplex and adding a 1-simplex.

Assume we are working on the step to compute the homology type and generators for K_i from that of K_{i-1}. Let us first consider the case where σ_i is a 0-simplex. This 0-simplex was not already in our complex, so it is not part of the boundary of any 1-simplex. Therefore, this 0-simplex is independent of everything in $H_0(K_{i-1})$, so we have $H_0(K_i) \simeq H_0(K_{i-1}) \times \mathbb{Z}$, where the new generator is simply the 0-cycle composed of this simplex.

Second, suppose σ_i is a 1-simplex. We know that $\partial\sigma_i$ is of the form $y - x$ for two points x and y. Let us treat σ_i as a path from x to y. There are now two cases:

In the first case, the two bounding points of σ_i are in different connected components (they have distinct homology classes in K_{i-1}). Then we have that $H_0(K_i) \simeq H_0(K_{i-1})/\iota_*(H_0(\partial\sigma_i))$. Therefore, when we mod by the subgroup generated by $[\partial\sigma_i]$, we are simply modding by the relation $[x] \sim [y]$, so the result is that two elements of the generating set, corresponding to

$[x]$ and $[y]$, are associated. This simply corresponds to a reduction of the homology group and its generating set. No other elements of the generating set are affected.

In the second case, the two bounding points of σ_i are in the same connected component. Then if we add σ_i to an existing path from y to x, we have a 1-cycle that contains σ_i exactly once, so it is a cycle containing σ_i the minimal positive number of times. We then know that $H_1(K_i) \simeq H_1(K_{i-1}) \times \mathbb{Z}$ where the new generator is our found 1-cycle.

3.3 Implementation Details

Because of the low-dimensionality of this case, the task of writing $\partial\sigma_i$ as a linear combination of the generating set for $H_{k-1}(K_{i-1})$ is rendered moot by the fact that the generators simply correspond to the connected components that the two endpoints lie in. Updating the generator corresponding to each point is a relatively simple task. When we need to merge two generators, we choose either one of them and mark all of the 0-simplices in its connected component as belonging to the other connected component. In this way, each connected component retains a canonical name, so that we may easily find all the 0-simplices in any given component.

Therefore, the only remaining task is that of finding the path from y to x in the last step of the above section. To do this, we will use a floodfill algorithm. Breadth-first search is used so that the shortest path is found and used. Because of the low-dimensionality, these cycles are never modified in the generating set for the first homology group, so we use the shortest path to simplify the output.

Each of these two operations is at most linear in the number of simplices, so this algorithm takes at most quadratic time in the number of simplices.

4 The Main Result

In this section we present an algorithm for incrementally computing the homology type and generating sets for the homology groups of a 2-dimensional simplicial complex where the coboundary of each 1-simplex consists of at most two 2-simplices. We follow the same incremental method based on the Mayer-Vietoris sequence as before, making use of a few key lemmas.

Lemma 1. *If a 2-simplex is part of a 2-cycle, then there exists a 2-cycle containing that 2-simplex exactly once.*

Proof. We will call two 2-simplices *neighbors* if they share a bounding 1-simplex. Two 2-simplices σ and τ are *connected* if there is a sequence of 2-simplices $\sigma = \sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_l = \tau$ such that σ_i and σ_{i+1} are neighbors for all i. Let x be a 2-cycle containing a given 2-simplex σ with a coefficient of c. Let y be the set of simplices in x that are connected to σ. y is clearly a 2-cycle. I claim that all of the simplices in y must have the same coefficient up to negation.

Let c denote the coefficient of σ in y. For any simplex τ in y, consider a sequence of 2-simplices $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_l = \tau$. We will induct on i to show that the coefficient of σ_i is the same as that of σ. The base case is clear, so we will only show the inductive step. For a given i, consider the 1-simplex that is part of the boundary of both σ_i and σ_{i+1}. Call this 1-simplex υ_i.

Because y is a 2-cycle, it must have no boundary. However, the only contributions of υ_i to the boundary are from σ_i and σ_{i+1} by assumption. Since the contribution of υ_i from σ_i is equal to either the coefficient of σ_i or its negation. Similarly, the contribution from σ_{i+1} is equal to either the coefficient of σ_{i+1} or its negation. Since these two contributions must cancel out, the coefficients of σ_i and σ_{i+1} must be either the same or negations of each other. This completes the inductive step, so the coefficient of each σ_i is the same as that of σ, up to negation.

Therefore, if c is the coefficient of σ in y, y/c is also a 2-cycle and contains σ exactly once, so it is the desired cycle. □

4.1 The Two Dimensional Algorithm

As before, we begin with a sequence of simplicial complexes $K_0, K_1, K_2, \ldots$ such that $K_i = \bigcup_{n=0}^{i} \sigma_n = K_{i-1} \cup \sigma_i$ for all i. For each simplex K_i we will compute the following:

- An independent generating set for each of the homology groups of K_i.
- The torsion coefficients corresponding to each of the above generators.
- A cycle containing σ_i exactly once, if one exists. We will call this cycle S_i.
- The representation of each of the generators for $H_n(K_{i-1})$ in terms of the generators of $H_n(K_i)$.

In this algorithm, we will internally represent the generators as linear combinations of the S_i.

For the incremental step, we will look at three situations.

4.1.1 Adding a 0-simplex

If σ_i is a 0-simplex, then we know that σ_i is a 0-cycle by itself, so $H_n(K_i) \simeq H_n(K_{i+1})$ for $n \geq 1$ and $H_0(K_i) \simeq H_0(K_{i-1}) \times \mathbb{Z}$, where the new generator is $S_i = \sigma_i$.

4.1.2 Adding a 1-simplex

If σ_i is a 1-simplex, then we will proceed almost identically to the one dimensional algorithm. If the two vertices of σ_i are in different components, then we reduce the generating set for H_0 by identifying the two generators corresponding to the two endpoints. If the two vertices of σ_i are in the same component, then we use a breadth-first search to find a cycle containing σ_i exactly once. This cycle becomes S_i and we update the generating set for H_1 by augmenting S_i with a torsion coefficient of 0.

4.1.3 Adding a 2-simplex

If σ_i is a 2-simplex, then the first order of business is to determine whether it is part of a 2-cycle or not. If σ_i is part of a 2-cycle, then we know that $H_2(K_i) \simeq H_2(K_{i-1}) \times \mathbb{Z}$, where the new generator is C_i, the 2-cycle that we found from before. If σ_i is not part of a 2-cycle then we write $\partial\sigma_i$ as a linear combination of the generators for $H_1(K_{i-1})$ and then we have that $H_1(K_i) \simeq H_1(K_{i-1})/\langle[\partial\sigma_i]\rangle$.

4.2 Implementation Details

Definition 5. *A* fundamental cycle S_i *is a cycle of any dimension that contains* σ_i *exactly once and for any other simplex* $\sigma_j \in S_i$*, we have* $j < i$*.*

Theorem 2. *The fundamental cycles* $\{S_i\}_{i\leq n}$ *form generating sets for the cycle groups* C_k *of the simplicial complex* K_n*.*

Proof. We induct on n. The base case, $n = 0$, is trivial as K_0 is empty. Now suppose that the fundamental cycles $\{S_i\}_{i\leq n}$ form generating sets for the cycle groups of K_n. Given a cycle Z in K_{n+1}, consider the coefficient c of σ_{n+1}. If there is a cycle containing σ_{n+1}, then we know that S_{n+1} exists and contains σ_{n+1} exactly once. Then $Z - cS_{n+1}$ is a cycle completely contained in K_n, and thus can be written as a linear combination of $\{S_i\}_{i\leq n}$. By adding cS_{n+1} to this representation, we obtain a representation of Z as a linear combination of $\{S_i\}_{i\leq n+1}$, as desired. This completes our induction. □

Throughout the implementation of this algorithm, all cycles are kept as linear combinations of either simplices, fundamental cycles, or current generators for the homology groups. To translate between these, we keep a list of representations of each of the fundamental cycles S_i in terms of the generators G_i and use the theorem presented above to create representations in terms of the fundamental cycles from representations in terms of simplices.

To determine whether a 2-simplex is part of a cycle, a floodfill is used. Lemma 1 tells us that the only possible cycle containing a given 2-simplex is exactly the sum of the 2-simplices that are connected to the given one. Computation of this sum is implemented using a floodfill algorithm and the boundary of the sum determines whether or not the simplex is in fact part of a cycle or not.

If not, the boundary of the new simplex is written as a linear combination of fundamental cycles via the following reduction algorithm. At each step, find the last index such that the corresponding simplex has a nonzero coefficient. Call the index i, so that the corresponding term is $c_i\sigma_i$. Then we add c_iS_i to our representation and subtract c_iS_i from our cycle, so all the remaining terms are only composed of indices strictly smaller than i. After at most n steps, this process will reduce any cycle in K_n to the trivial cycle, at which point we have finished constructing the desired representation.

Finally, when computing the quotient of a homology group by a new relation, an elementary Smith Normal Form reduction algorithm is used so that we have a sequence of row and column operations that reduce the natrix to Smith Normal Form. Row operations simply

compose the relations in different ways, so they do not change the generators. However, when applying a column operation, we are taking two terms in each relation $c_iG_i + c_jG_j$ and replacing them with $c'_iG'_i + c'_jG'_j$. The column operations that we perform are made such that $c'_i = \gcd(c_i, c_j)$.

By Bezout's Identity, we have integers x and y such that $xc_i + yc_j = c'_i$. Letting $\alpha = \frac{c_i}{c'_i}, \beta = \frac{c_j}{c'_i}$, we get that we want to carry out the following transformation: $c'_i = xc_i + yc_j$, $c'_j = -\beta c_i + \alpha c_j$, $G'_i = \alpha G_i + \beta G_j$, $G'_j = -yG_i + xG_j$. When we apply these transformations, we see that $c'_iG'_i + c'_jG'_j = (xc_i + yc_j)(\alpha G_i + \beta G_j) + (-\beta c_i + \alpha c_j)(-yG_i + xG_j) = (xc_i\alpha + yc_j\alpha + y\beta c_i - y\alpha c_j)G_i + (x\beta c_i + y\beta c_j - x\beta c_i + x\alpha c_j)G_j = c_iG_i + c_jG_j$. Therefore, by applying these transformations to the generators and the representations of the fundamental cycles, we obtain new generators and new representations. The integers x and y are constructed via the Extended Euclidean Algorithm.

5 Results

This algorithm was implemented in the C++ programming language. For purposes of simplicity, a naïve implementation was used for the Smith Normal Form reduction step. Furthermore, for the purposes of ensuring correctness, as few optimizations were made as possible. This has the effect that the times reported here are a gross overestimate of the best time possible for incremental computations of homology via this algorithm. The program was tested on tori, Klein bottles, and real projective planes, triangulated with various finenesses.

5.1 Torus

Triangulations of the torus were made of $6n^2$ simplices, with $n \geq 3$. Of these, n^2 are 0-simplices, $3n^2$ are 1-simplices, and $2n^2$ are 2-simplices. To create the triangulation, an $(n+1) \times (n+1)$ grid of points was created. The opposite sides of the square are to be associated, so each point in row n and each point in column n was labeled with the same number as the point on the opposite side of the square on the same row or column. Then a triangulation of this square was created by connecting each point to the one directly to the right, directly below, and diagonally below it to the right. Finally, the small triangles drawn by this method are included into the complex to complete the surface. A schematic for this process for $n = 3$ is shown in Figure 1.

To visualize the final output of the procedure, the square representing the torus is placed in the uv-plane such that the square represents all values with $0 \leq u < 2\pi$ and $0 \leq v < 2\pi$. Then the torus is embedded into $\mathbb{R}^3$ with the parametric equations

$$\begin{aligned} x &= \cos u(1 - 0.3\cos v) \\ y &= \sin u(1 - 0.3\cos v) \\ z &= 0.3\sin v \end{aligned}$$

The final result of this method when run on the torus can be seen in Figure 3.

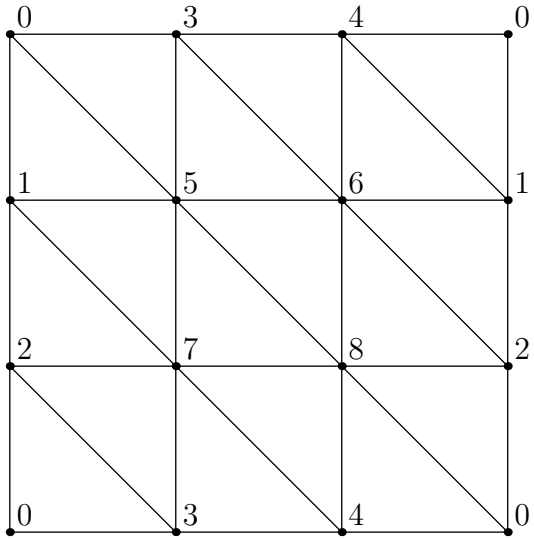
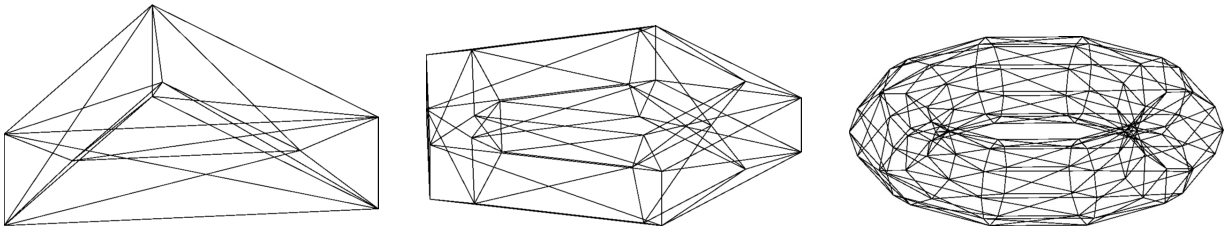


Figure 1: The schematic for a triangulation of a torus with $n = 3$



(a) $n = 3$ (b) $n = 5$ (c) $n = 10$

Figure 2: Tori in $\mathbb{R}^3$

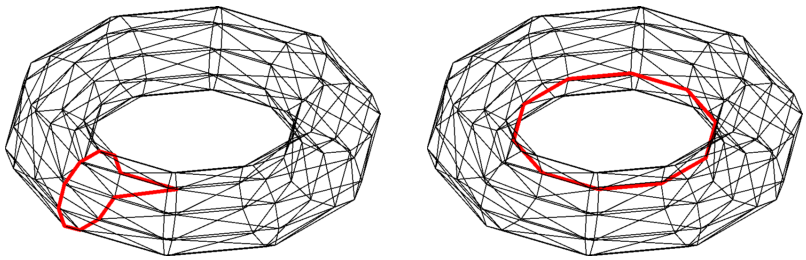


Figure 3: Generators for the first homology group of the torus

5.2 Klein Bottle

Triangulations of the Klein bottle were made of $6n^2$ simplices in a similar manner to those for the torus. A square was divided into n^2 smaller squares, each of which is divided into two triangles. The top and bottom sides of the square are associated in the same orientation as in the torus, but unlike the torus the left and right sides are associated in opposite orientations, creating a Klein bottle. In order for the triangulation to have a relatively simple immersion into $\mathbb{R}^3$, the left and right sides of the torus are associated so that the top of the left side corresponds to the middle of the right side. The schematic for the triangulation for $n = 4$ is shown in Figure 4.

To visualize the output, the square is placed in the uv-plane such that the square represents all values with $0 \leq u < 2\pi$ and $0 \leq v < 4\pi$. The immersion is then defined by the equations

$$x = \begin{cases} (2.5 - 1.5\cos v)\cos u, & 0 \leq v < \pi \\ (2.5 - 1.5\cos v)\cos u, & \pi \leq v < 2\pi \\ -2 + (2 + \cos u)\cos v, & 2\pi \leq v < 3\pi \\ -2 + 2\cos v - \cos u, & 3\pi \leq v < 4\pi \end{cases}$$

$$y = \begin{cases} (2.5 - 1.5\cos v)\sin u, & 0 \leq v < \pi \\ (2.5 - 1.5\cos v)\sin u, & \pi \leq v < 2\pi \\ \sin u, & 2\pi \leq v < 3\pi \\ \sin u, & 3\pi \leq v < 4\pi \end{cases}$$

$$z = \begin{cases} -2.5\sin v, & 0 \leq v < \pi \\ 3v - 3\pi, & \pi \leq v < 2\pi \\ (2 + \cos u)\sin v + 3\pi, & 2\pi \leq v < 3\pi \\ -3v + 12\pi, & 3\pi \leq v < 4\pi \end{cases}$$

An image of this immersion is shown in Figure 5, and the two generators are shown in Figure 6.

5.3 Runtime Analysis

As a final verification for the usefulness of this approach in computational homology, the runtimes for this naïve method were recorded for the torus and the Klein bottle for each n between 3 and 12, inclusive.The results can be seen in Figure 7. When plotted with logarithmic scales on both axes, the time taken is approximately linear with a slope of 8. Remembering that the complexes have $O(n^2)$ simplices, this suggests that the algorithm takes time proportional to the fourth power of the number of simplices. As this is a very primitive implementation, these results are encouraging because even the simplest methods result in a polynomial time algorithm.

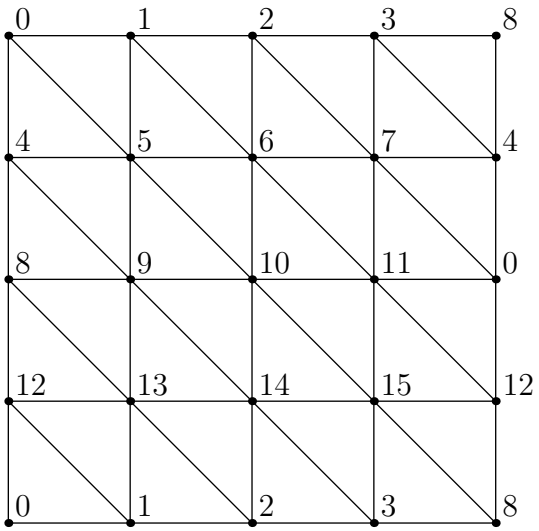


Figure 4: The schematic for a triangulation of a Klein bottle with $n = 4$

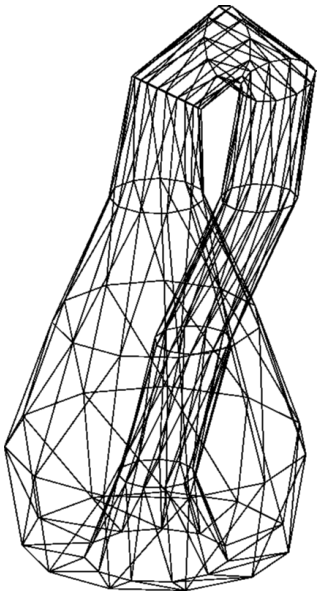
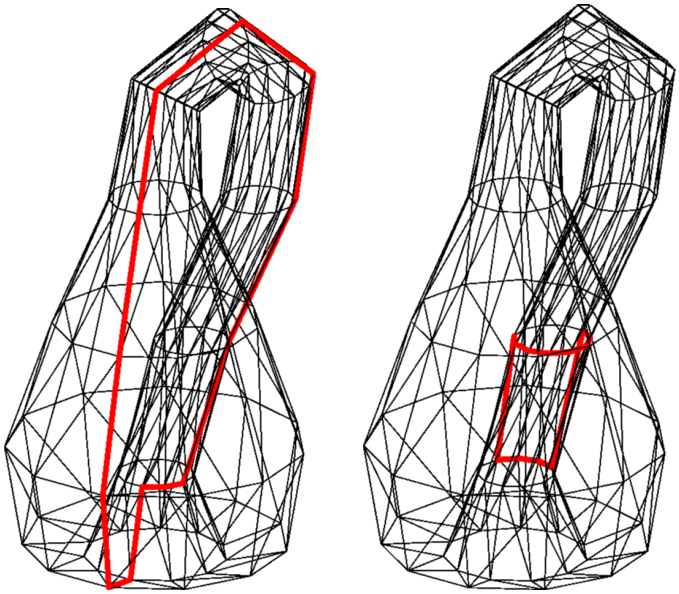


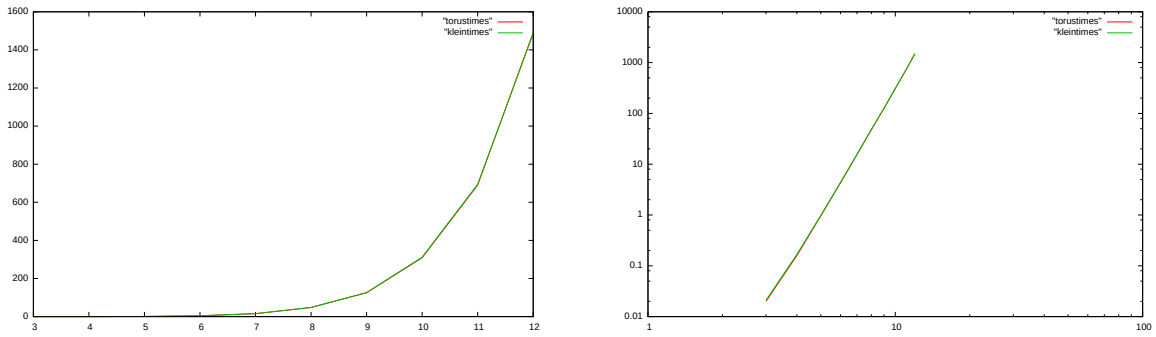
Figure 5: The Klein bottle immersed in $\mathbb{R}^3$



(a) Torsion 0 Generator (b) Torsion 2 Generator

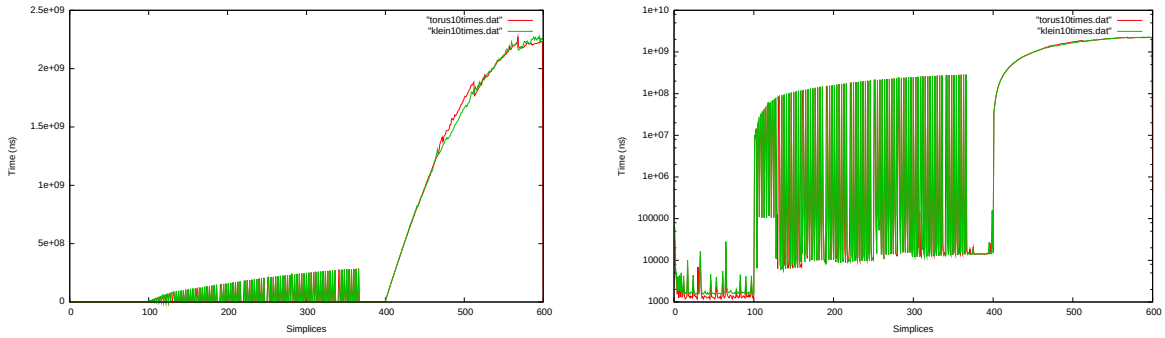
Figure 6: Generators for the first homology group of the Klein bottle

Additionally, the time per update was recorded and that data can be seen in Figure 8. The time taken can easily be seen to have three distinct behaviors, corresponding to 0-simplices, 1-simplices, and 2-simplices. Each update takes a relatively short amount of time, so for applications where the simplicial complex requires significant time to build, this method allows the homology computation to be done simultaneously, resulting in very small additional time requirements.



(a) Linear Axes (b) Logarithmic Axes

Figure 7: Total time required for a simplicial complex containing $6n^2$ simplices (seconds)



(a) Linear Axes

(b) Logarithmic Axes

Figure 8: Update time required for a simplicial complex containing n simplices (nanoseconds)

6 Conclusion

The incremental method for computing homology as described above serves as a useful intermediate between highly algebraic methods and highly geometric methods for computing homology type and homology generators in generality. While this method is not as generally applicable as purely algebraic methods, which rely on only the matrix representation of the boundary homomorphisms, it also surpasses geometric algorithms by removing the constraint that the triangulation should be embedded in $\mathbb{R}^3$, and particular should be orientable. Additionally, due to the incremental nature of the algorithm, on-line computation of homology is now possible. For applications where the simplicial complex takes a significant time to build the complex, this method can be run concurrently and so effectively the only time required is the time for the final update. Therefore, for such applications this method could surpass even the extremely refined algebraic methods, as the matrix reduced in each step is much simpler than that of the complete boundary homomorphism.

The results for this paper were obtained with an unrefined implementation of the algorithm described. Many simple optimizations that are possible were not made, such as removing generators of torsion 1 from consideration. Although the runtimes found in this paper appear to be extremely high, in practice a more sophisticated implementation should have vastly superior performance. Furthermore, the times shown here are representative of the worst case performance. If torsion 1 generators are removed and the simplices are added in an intelligent order, then the Smith Normal Form reductions will be performed on extremely small matrices compared to the size of the input.

Areas for future research include a more refined implementation of the algorithm including optimizations such as those listed above, an extension of this algorithm to remove the constraint that at most two 2-simplices may meet at a single 1-simplex, and an extension of the algorithm to higher dimensional complexes. Additional research could be done in the field of object recognition, using the incremental approach to create an on-line recognition algorithm as additional surfaces are recognized without recomputing the entire homology type a multitude of times.

References

[1] Delfinado, Cecil and Herbert Edelsbrunner. "An incremental algorithm for Betti numbers of simplicial complexes", *Proceedings of the ninth annual symposium on Computational geometry*, p.232-239, May 18-21, 1993, San Diego, California, United States

[2] Dey, Tamal K. and Sumanta Guha. "Computing Homology Groups of Simplicial Complexes in R^3", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, p.398-407, May 22-24, 1996, Philadelphia, Pennsylvania, United States.

[3] Donald, Bruce and David Chang. "On the complexity of computing the homology type of a triangulation", *Proceedings of the 32nd annual symposium on Foundations of computer science*, p.650-661, September 1991, San Juan, Puerto Rico.

[4] Hatcher, Allen. "Algebraic Topology". Cambridge University Press, Cambridge, 2002.

[5] Morandi, Patrick J. "The Smith Normal Form of a Matrix", Unpublished, February 17, 2005.

[6] Peltier, Samuel, Adrian Ion, Yll Haxhimusa, and Walter Kropatsch. "Computing Homology Group Generators of Images Using Irregular Graph Pyramids". *Graph-Based Representations in Pattern Recognition*. Springer Berlin, 2007. p.283-294.

[7] Peltier, Samuel, Sylvie Alayrangues, Laurent Fuchs, and Jacques-Olivier Lachaud. "Computation of Homology Groups and Generators". *Computers and Graphics* v.30 n.1, p.62-69, February, 2006.

[8] Storjohann, Arne. "Near Optimal Algorithms for Computing Smith Normal Forms of Integer Matrices". *Proceedings of the 1996 international symposium on Symbolic and algebraic computation*, p.267-274, July 24-26, 1996, Zurich, Switzerland.