CUP PRODUCTS IN COMPUTATIONAL TOPOLOGY

JONATHAN HUANG

Abstract. Topological persistence methods provide a robust framework for analyzing large point cloud datasets topologically, and have been applied with great success towards homology computations on simplicial complexes. In this paper, we apply the persistence algorithm towards calculating a set of invariants related to the cup product structure on the cohomology ring for a space. These invariants express the extent to which cohomology classes can be obtained as cup products of lower dimensional cohomology classes in the space. To calculate these invariants, we apply persistence to a chain complex associated with the Alexander-Whitney product in homology which dualizes to be the cup product in cohomology. We show that the method is practical to implement by showing results from an implementation for PLEX, a Matlab toolkit which was developed for such topological computations.

1. Introduction

In the last decade, there have been many advances in the young field of Computational Topology. Originally a branch of Computational Geometry, it is the development of algorithms motivated by topological ideas with applications extending from computer science, to molecular biology. While these algorithms are generally developed using algebraic topology as a guide towards various applications, they also have the potential to become a useful set of computational tools for algebraic topologists.

In general, it is the objective to design methods (motivated by results from algebraic topology) to compute topologically invariant combinatorial objects for a space, which can then be used to show that certain spaces are not homotopy equivalent to each other. These methods should be reasonably robust so as to not fail in the presence of noisy, or vast amounts of input data that one is often likely to work with in the field of computational geometry, and should be fully automated with an eye towards efficiency.

In the rest of this paper, I will describe a method for computing one such invariant, related to the cup product structure on a class of topological spaces, the simplicial complexes, which have the advantage of having a concise combinatorial representation well suited for computers. We will begin in the setting of simplicial complexes, but move towards more abstract algebraic representations in order to leverage some important tools from algebraic topology.

I will give a theoretical discussion about the motivation and strategy by which we compute these invariants. Then I will proceed to give implementation details and results to show the practicality of the algorithms involved.

Date: June, 2005.

Key words and phrases. Computational Topology, Algebraic Topology, Cup Product, Alexander-Whitney Chain Map.

2. The Reduction Algorithm for Homology

For computational simplicity, one often restricts to the *(finite) simplicial complexes* in computational topology. This class of spaces has the advantage that they can easily be represented in memory on a computer, and as will be discussed in this section, there is a well-understood algorithm for computing their homology groups.

To begin, we will recall some basic definitions from topology [8]. An *n-simplex* is defined as the convex hull of an ordered list of n affine independent points in a euclidean space. We can write an n-simplex σ as $\sigma = [v_0, \ldots, v_n]$. The unordered set $\{v_0, \ldots, v_n\}$ is called the *vertex set* for σ. If σ' is a simplex whose vertex set is a subset of the vertex set for σ, then we say that σ' is a *face* of σ. We say that σ is a *coface* of σ'.

Definition 2.1. A *finite simplicial complex* X is a finite set of simplices such that:

(1) For every $\sigma \in X$, every face of σ is also a member of X.
(2) For any two simplices $\sigma_1, \sigma_2 \in X$, $\sigma_1 \cap \sigma_2$ is either empty, or a common face of both σ_1 and σ_2.

For a simplicial complex, there is a standard algorithm for computing the homology groups in each dimension [7]. Suppose $\partial_n : C_n \to C_{n-1}$ is the n^{th} dimensional boundary operator for a simplicial complex X. Let $Z_n = \ker \partial_n$ and $B_{n-1} = \text{im } \partial_n$. Then $H_n(X)$ is defined as $H_n(X) = Z_n/B_n$. To develop the reduction algorithm, we first observe that a chain complex

$$\cdots \to C_{n+1} \xrightarrow{\partial} C_n \xrightarrow{\partial} C_{n-1} \to \cdots$$

splits into a direct sum of subcomplexes each with at most two nonzero terms, since each C_n splits as $C_n = Z_n \oplus B_n$ (where B_n is actually an isomorphic copy of im ∂_n in C_n). The splitting is as follows:

$$\begin{array}{ccccccccccccc}
 & & & & \longrightarrow & C_{n+1} & \longrightarrow & C_n & \longrightarrow & C_{n-1} & \longrightarrow & C_{n-2} \\
 & & & & & & & \vdots & & \vdots & & \\
 & & & & & 0 & \longrightarrow & B_n & \longrightarrow & Z_{n-1} & \longrightarrow & 0 \\
 & & & & & \oplus & & \oplus & & \oplus & & \\
 & & & 0 & \longrightarrow & B_{n+1} & \longrightarrow & Z_n & \longrightarrow & 0 & & \\
 & & & \oplus & & \oplus & & \oplus & & & & \\
0 & \longrightarrow & B_{n+2} & \longrightarrow & Z_{n+1} & \longrightarrow & 0 & & & & & \\
\vdots & & \vdots & & & & & & & & &
\end{array}$$

In the case where the chain complex arises from a finite simplicial complex, then these C_n are finitely generated and so one can deduce that a further splitting occurs for the sequence $0 \to B_{n+1} \to Z_n \to 0$. Let M_n be the matrix representation for ∂_n with respect to the standard basis of C_n. The reduction algorithm will be to use the elementary row and column operators to reduce M_n to a more manageable form. The allowed operations are:

(1) exchange row i and row j.
(2) multiply row i by -1.
(3) replace row i by (row i) + q(row j), where q is an integer and $j \neq i$.

and their analagous column operations.

The reduction algorithm reduces M_n to its *(Smith) Normal Form*, $\tilde{M}_n$, where all entries are zero except possibly a block at the top left corner which may contain nonzero entries down the diagonal.

$$\tilde{M}_n = \left(\begin{array}{ccc|c} b_{l_1} & & 0 & \\ & \ddots & & 0 \\ 0 & & b_{l_n} & \\ \hline & 0 & & 0 \\ & & & \end{array}\right)$$

The columns in this matrix with non-zero entries correspond to a basis for the image and each gives a summand of the form $0 \to \mathbb{Z} \xrightarrow{b_{l_k}} \mathbb{Z} \to 0$. The zero columns correspond to a basis for Z_n and each one gives a summand of the form $0 \to \mathbb{Z} \to 0$. It follows that by computing the normal form for the boundary operators in all dimensions, we can read off a characterization of H_n using the following rules:

(1) For each diagonal entry b_l of $\tilde{M}_n$ greater than one, there is a torsion summand in H_{n-1}, $\mathbb{Z}_{b_l}$. If the coefficient group is G, then the contribution is just G/b_lG.
(2) rank $Z_n = m_n - l_n$ where m_n is the number of columns of the matrix.
(3) rank $B_n = l_{n+1}$

With this algorithm, it is easy to compute betti numbers, as:

$$\beta_n = \text{rank } Z_n - \text{rank } B_n = m_n - l_n - l_{n+1}$$

As we will see presently, homology does not do as well as cohomology as an invariant, and the reduction algorithm itself has some shortcomings if implemented naively.

3. Simplicial Cohomology

For reasons of theoretical and practical importance in our computational setting, we will from this point deal only with cohomology with coefficients taken from a field F. In practice, F will be either $\mathbb{Z}_2$ or $\mathbb{Q}$.

The assumption that we are working over a field greatly simplifies matters because now the (co)chain groups are vector spaces. Because of this, one can apply standard techniques in linear algebra.

Despite the numerous applications of homology, one finds that it is rather limited in ways. Consider, for example, comparing the homology of the two spaces $X = S^1 \times S^1$ and $Y = S^2 \vee S^1 \vee S^1$. We see in both cases that:

$$H_n(X;F) = H_n(Y;F) = \begin{cases} F & n = 0 \\ F \oplus F & n = 1 \\ F & n = 2 \\ 0 & n \geq 3 \end{cases}$$

while it seems "obvious" that these spaces should not be homotopy equivalent. But it is not so clear how to prove that this is the case without resorting to a cohomology computation.

I will return to this example of the torus and the sphere with two handles shortly. The problem of course is that while it is true that two homotopy equivalent spaces must share the same homology groups, the converse of this statement is not true.

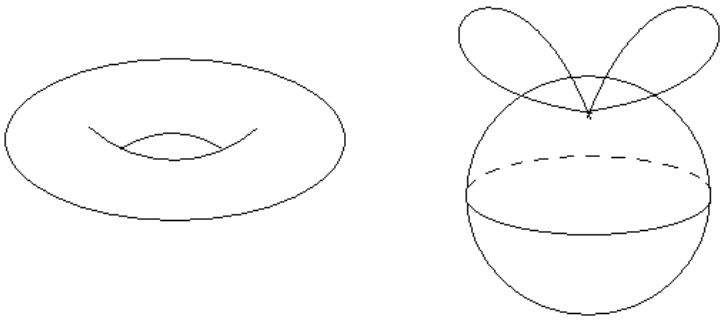

FIGURE 1. $S^1 \times S^1$ and $S^2 \vee S^1 \vee S^1$

In general, simplicial cohomology will have the same difficulty, but we would like to be able to calculate it because it can distinguish more spaces apart than homology can as it turns out to be a "finer" invariant.

Formulated as the "dual" to homology, it is surprising that one would get any extra information from cohomology at all. In fact, by the universal coefficient theorem [7], the sequence

$$0 \to Ext(H_{n-1}(C), R) \to H^n(C; R) \to Hom(H_n(C); R) \to 0$$

is exact, which says that as groups, the $H^n(C; R)$ are determined up to isomorphism by the homology groups $H_n(C; R)$. If $R = F$ is a field, this sequence implies an isomorphism $H^n(C; F) \approx Hom(H_n(C); F)$.

The "extra information" we desire can be found in the ring structure that cohomology is equipped with, where the multiplication comes from the so-called cup product.

Revisiting the previous example, we see that this cup product structure is sufficient to distinguish the torus, $X = S^1 \times S^1$, from the two-handled sphere, $Y = S^2 \vee S^1 \vee S^1$. Again, as groups, the cohomology of these groups is the same:

$$H^n(X; F) = H^n(Y; F) = \begin{cases} F & n = 0 \\ F \oplus F & n = 1 \\ F & n = 2 \\ 0 & n \geq 3 \end{cases}$$

Let α_1 and α_2 be cocycles generating $H^1(X; F)$. As it turns out, $\alpha_1 \smile \alpha_2$ is a generator for $H^2(X; F)$. This can be seen as a consequence of Poincare Duality since the torus is a closed orientable manifold. However, if β_1 and β_2 are cocycles generating $H^1(Y; F)$, then $\beta_1 \smile \beta_2 = 0$. The conclusion is that the cohomology *rings* $H^*(X; F)$ and $H^*(Y; F)$ are not isomorphic even though the groups at each degree of the graded rings are isomorphic.

4. The Cup Product

The *cup product* in cohomology is the operation which turns the cohomology groups of a space X into a commutative graded ring: $H^*(X; R) = \oplus_{k \geq 0} H^k(X; R)$.

In this section, I will present a formulation of the cup product via the Alexander-Whitney diagonal approximation and then discuss some invariants that are related to the cohomology ring resulting from the cup product.

The main map which we will be concerned with is known as the *Alexander-Whitney homomorphism* [6] and is conveniently defined on the level of chains. For a topological space X, denote the singular chain complex for X by $C_*(X)$. For a singular n-simplex σ in a simplicial complex, let ${}_i\sigma$ be the front i-face of σ. For example, if $\sigma = [v_0, \ldots, v_i, \ldots, v_n]$, then $\sigma_i = [v_0, \ldots, v_i]$. Similarly, σ_j is defined to be the back j-face of σ.

We define a tensor product of chain complexes as:

$$(C_*(X) \otimes C_*(Y))_n = \oplus_n^{i=0} C_i(X) \otimes C_{n-i}(Y)$$

The Alexander-Whitney homomorphism, $A : C_i(X \times Y) \to (C_*(X) \otimes C_*(Y))_i$ is then given on a simplex $(\sigma, \tau) \in X \times Y$:

$$A(\sigma, \tau) = \sum_{i=0}^{n} {}_i\sigma \otimes \tau_{n-i}$$

and induces a map in homology, $A_* : H_i(X \times Y) \to H_i(C_*(X) \otimes C_*(Y))$. As an example, suppose σ is a 3-simplex labeled with vertices $[0, 1, 2, 3]$. Then $\tau(\sigma) = [0] \otimes \sigma + [0, 1] \otimes [1, 2, 3] + [0, 1, 2] \otimes [2, 3] + \sigma \otimes [3]$.

Proposition 4.1. *Some properties of the Alexander-Whitney homomorphism are:*

1. *A is functorial in (X, Y).*
2. *A is a chain map.*
3. *A_* is an isomorphism in homology and in fact is a chain equivalence of $C(X \times Y)$ with $C_*(X) \otimes C_*(Y)$.*

The proof of the (1) and (2) of the proposition can be found in Greenberg [6], and (3) follows from the Eilenberg-Zilber theorem and shows that we can compute the homology of a product space by looking at the tensor product chain complex.

The definition of the cup product requires several steps. The cohomology *cross product* (also called the *external cup product*) relates the direct product of two cohomology groups to the cohomology group of their product space. To define it, we first define an *algebraic* cross product for chain complexes A_*, B_*,

$$\times^{alg} : H^i(A^*) \otimes H^j(B^*) \to H^{i+j}((A_* \otimes B_*)^*)$$

where $\alpha \times^{alg} \beta(\sum \sigma_i \otimes \tau_i) = \sum \alpha(\sigma_i) \cdot \beta(\tau_i)$

The *cohomology* cross product is the composition

$$H^i(X; R) \times H^j(Y; R) \xrightarrow{\times^{alg}} H^{i+j}((C_*(X) \otimes C_*(Y))^*) \xrightarrow{A^*} H^{i+j}(X \times Y; R)$$

where $\times^{alg}$ is the *algebraic* cohomology cross product and $A^* : C_*(X \times Y) \to C_*(X) \otimes C_*(Y)$ is the dual of the Alexander-Whitney chain map.

Let $\Delta : X \to X \times X$ be the diagonal map given by $x \mapsto (x, x)$. The cup product can now be defined as the composition of the cohomology cross product with the dualized diagonal map, $\Delta^* : H^*(X \times X) \to H^*(X)$:

$$\smile : H^i(X) \times H^j(X) \to H^{i+j}(X \times X) \to H^{i+j}(X)$$

The key reason why it can be defined for cohomology but not for homology turns out to be the contravariance of the cohomology functor. For a general space X, it is not clear how one would define a map $X \times X \to X$ without resulting in a trivial

product (which would be the case if we simply projected onto a factor), but A^* results in a map that gives a nontrivial product in cohomology. There are several important facts about the cup product which the next proposition will summarize.

Proposition 4.2. *Properties of the Cup Product*

1. *(Leibnitz Rule)* $\delta(\alpha \smile \beta) = \delta\alpha \smile \beta + (-1)^i \alpha \smile \delta\beta$ *for* $\alpha \in C^i(X)$ *and* $\beta \in C^j(X)$.
2. *(Identity) Let* $1 \in H^0(X)$ *be represented by the cocycle that maps every 0-simplex in* X *to the multiplicative identity in the field. Then for* $\alpha \in H^i(X)$, $1 \smile \alpha = \alpha \smile 1 = \alpha$.
3. *(Associativity) For cochains* α, β, γ, $(\alpha \smile \beta) \smile \gamma = \alpha \smile (\beta \smile \gamma)$
4. *(Graded Commutativity) For* $\alpha \in H^i(X)$ *and* $\beta \in H^j(X)$, $\alpha \smile \beta = (-1)^{ij} \beta \smile \alpha$.
5. *Let* $f : X \to Y$ *be a continuous map of spaces, and let* $f^* : H^i(Y) \to H^i(X)$ *be the induced maps in cohomology. Then,*

$$f^*(\alpha \smile \beta) = f^*(\alpha) \smile f^*(\beta)$$

The proofs can be found in any introductory text on algebraic topology. In particular, observe that the middle three properties make $H^*(X) = \oplus_{i \geq 0} H^i(X)$ into a graded commutative ring with identity, and the fourth shows the cup product structure to be a topological property of a space.

Let $I^*(X) \subset H^*(X)$ be the graded ideal generated by the elements of $H^*(X)$ with positive grading. To computationally extract information about the cup product structure on a space X, we will compute a combinatorial set of invariants given by $dim_F(I/I^2), dim_F(I^2/I^3), \ldots, dim_F(I^k/I^{k+1})$, where I^k is defined inductively as

$$I^k(X) = \left\{ \sum_{finite} \alpha_i \smile \beta_i \mid \alpha_i \in I^{k-1}(X) \text{ and } \beta_i \in I(X) \right\}$$

(Notice that if we had allowed I to be the entire cohomology ring, then I^k would contain the identity cocycle class for all k, and result in trivial invariants.) In particular, we would like to compute $dim_F(I/I^2)$ which has a nice interpretation as measuring the degree to which elements in the cohomology ring appear directly as cup products of elements with lower grading. The method by which these can be computed will be shown next after the diagonal approximation is defined.

4.1. **The Diagonal Approximation.** The Alexander-Whitney diagonal approximation, $\tau : C_*(X) \to C_*(X) \otimes C_*(X)$, comes from composing the diagonal map with A and can be explicitly written as:

$$\tau(\sigma) = A \circ \Delta(\sigma) = \sum_{i=0}^{n} {}_i\sigma \otimes \sigma_{n-i}$$

Notice that the cup product can now be rewritten more concretely for cochains $\alpha \in C^p(X)$ and $\beta \in C^q(X)$ as ([9])

$$\begin{aligned} \alpha \smile \beta([v_0, \ldots, v_{p+q}]) &= \tau^*(\alpha \times^{alg} \beta)([v_0, \ldots, v_{p+q}]) \\ &= \alpha([v_0, \ldots, v_p]) \cdot \beta([v_p, \ldots, v_{p+q}]) \end{aligned}$$

We can also define a reduced version of the map, $\bar{\tau} : \bar{C}_*(X) \to \bar{C}_*(X) \otimes \bar{C}_*(X)$, where $\bar{C}_*(X) = C_*(X)/C_*(x_0)$ for some basepoint $x_0 \in X$.

We can finally compute the invariant, $dim(I/I^2)$. To do this, we use a general fact from algebra. Given an R-module homomorphism $f : A \to B$ and its dual, $f^* : B^* \to A^*$, where A and B are both finitely generated, and $A^* = Hom_R(A, R)$, $B^* = Hom_R(B, R)$, we have

$$(\ker f)^* \approx \text{coker}(f^*)$$

To compute $dim_F(I/I^2)$ is the same as computing the dimension of the cokernel of the cup product map. Thus using the above fact, it is equivalent to computing the dimension of the kernel of the Reduced Alexander-Whitney diagonal approximation, $\bar{\tau}_*$. To actually compute this number, however, we will turn to a slightly unconventional application of topological persistence, which will be the topic of the next section.

5. Topological Persistence

Despite the simplicity and elegance of studying homology as a topological invariant, it tends to have several shortcomings, several of which have been discussed in the section on cohomology. Computationally speaking, the reduction algorithm for homology requires operations to be performed with exact integer arithmetic, and this is difficult if we would like to deal with large high dimensional complexes because entries in the matrices tend to get very large. Furthermore, calculating homology on large datasets naively may not always yield the result that we would like to have due to possible noisiness or incompleteness in the data. The presence of noise lends to the presence of many small topological features in the larger space and so it would be desirable to "filter" out the small features and to retain the more important ones. We are thus led to the notion of persistent homology [10].

In persistent homology, one considers a simplicial complex K which has been given a *filtration* that explains how K might be built up in steps. Formally we say that a *persistence complex* is an increasing sequence of simplicial complexes along with its boundary maps. In general, it is a family of chain complexes $\{C_*^i\}_{i\geq 0}$ (where i denotes the filtration index of a complex) and chain maps f_i that include C_*^i into C_*^{i+1}. Along with boundary maps, we have the diagram

$$\begin{array}{ccccccc}
 & & \downarrow & & \downarrow & & \\
C_2^0 & \longrightarrow & C_2^1 & \longrightarrow & C_2^2 & \longrightarrow & \\
\downarrow & & \downarrow & & \downarrow & & \\
C_1^0 & \longrightarrow & C_1^1 & \longrightarrow & C_1^2 & \longrightarrow & \\
\downarrow & & \downarrow & & \downarrow & & \\
C_0^0 & \longrightarrow & C_0^1 & \longrightarrow & C_0^2 & \longrightarrow &
\end{array}$$

By specifying how a complex is built using filtrations, there is now more topological information to consider. Of course we can begin by computing the homology of each complex at each filtration index, but the interesting thing now is what happens to topological features (cycles) as we carry them step by step through the filtration indices.

As an example, consider the filtered complex given in figure 2. At step 0, there are two contractible connected components, which become two circles at step 1. At step 2, the two components join to form just one component. Finally, one of the circles is filled, killing off a 1-cycle class in homology. Being in the context of a filtered complex, we can think of cycles as having "lifetimes", and at any given time step, the homology of the current complex is the sum of the cycles which are

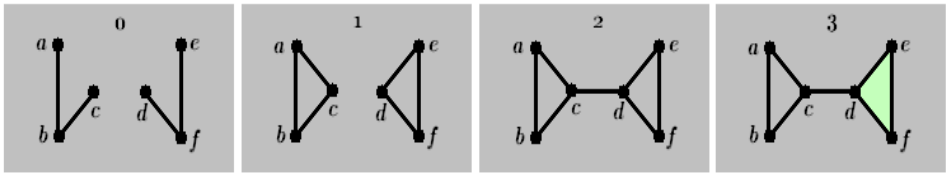

FIGURE 2. A Filtered Simplicial Complex

"alive" at that time. The cycles which never die give the homology of the entire complex at the end of the filtration. To formalize this notion, we give the following generalization of a homology group.

Definition 5.1. Let C^i be a filtered chain complex, with i denoting filtration index. Let ∂_k^i denote the k^{th} boundary operator for the i^{th} complex in the filtration, and let Z_k^i and B_k^i be the k^{th} cycle and boundary groups for the i^{th} complex, respectively. Then we define the *p-persistent k^{th}-homology group of C^i* as:

$$H_k^{i,p} = Z_k^i / (B_k^{i+p} \cap Z_k^i)$$

Alternatively, $H_k^{i,p}$ can be defined as the image of the injection $\eta_k^{i,p} : H_k^i \to H_k^{i+p}$ which sends a homology class to the one containing it.

Definition 5.2. We define a *persistence module* $\mathcal{M}$ to be a family of R-modules M^i with homomorphisms $\phi_i : M^i \to M^{i+1}$. This is written as $\mathcal{M} = \{M^i, \phi_i\}_i$.

The example to keep in mind here is the homology of a persistence complex, where the maps ϕ_i map homology classes to the classes that contain them.

Definition 5.3. We say that a persistence complex or module is of *finite type* if each component is finitely generated and if for sufficiently large i, the corresponding maps ϕ_i become R-module isomorphisms.

The setup is now as follows. We are given a persistence module $\mathcal{M} = \{\mathcal{M}^i, \phi^i\}_{i \geq 0}$ over a ring R, and we would like to have a simple classification for these modules and to develop a way to identify elements with corresponding elements at other timesteps of the filtration. In other words, we will need to choose bases which are compatible across the filtration in order to compute persistent homology. The main classification is supplied by Zomorodian and Carlsson [10] and comes in two main steps.

5.1. **The Artin-Rees Correspondence.** Let $R[t]$ be the polynomial ring in t equipped with the standard grading. We now combine all of the complexes in the filtration to get a single structure and encode the time step at which an element is born by a polynomial coefficient. To be precise, let $\alpha(\mathcal{M})$ be the graded $R[t]$-module defined by:

$$\alpha(\mathcal{M}) = \bigoplus_{i=0}^{\infty} M^i$$

The action of t is an upward shift in grading: i.e.,

$$t \cdot (m^0, m^1, m^2, \dots) = (0, \phi(m^0), \phi(m^1), \phi(m^2), \dots)$$

And the R-module structure is just the sum of the individual component R-modules. Loosely, consider a simplex σ which enters the filtration at time step 2. Then we would write it as σt^2. Shifted by one step along the filtration, σ exists at time 3 as σt^3.

The main correspondence given by the Artin-Rees theory in commutative algebra, is that α defines an equivalence (of categories) between the category of finite persistence modules over R and the category of finitely generated nonnegative graded modules over $R[t]$.

5.2. **A Structure Theorem for Graded Modules over a Graded PID.** If we assume that R is a field, then $R[t]$ becomes a principal ideal domain, which allows us to leverage a well-known classification theorem for graded modules over graded PIDs.

Theorem 5.4. *Let D be a graded PID and M a graded D-module. Then M decomposes as:*

$$M \approx \left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} D\right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} D/d_j D\right)$$

where $d_j \in D$ are homogeneous elements such that $d_j | d_{j+1}$, $\alpha_i, \gamma_j \in \mathbb{Z}$, and Σ^α is a shift upwards in grading by α.

In our case, for $R = F$ a field, we have that all graded ideals are of the form (t^n), so by the theorem, graded $F[t]$-modules can be decomposed as:

$$M \approx \left(\bigoplus_{i=1}^{n} \Sigma^{\alpha_i} F[t]\right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{\gamma_j} F[t]/(t^{n_j})\right)$$

Via the correspondence with finite-type persistence modules given above, the coefficients for this module decomposition can be made meaningful. The γ_j and α_j describe when a basis element is created along the filtration. The element then either persists along the filtration until the time $\gamma_j + n_j - 1$ when it dies, or it never dies if it lives in a free summand. Therefore, the "lifetime" of a basis element can be described by the pairing of its birth and death times, $(\gamma_j, \gamma_j + n_j)$ or (α_j, ∞). This pairing gives a way to parametrize the isomorphism classes of finitely generated $F[t]$-modules with a finite set of combinatorial invariants. With this in mind, we make the following definition.

Definition 5.5. A $\mathcal{P}$-interval is an ordered pair (i, j) with $0 \leq i < j \in \mathbb{Z} \cup \{+\infty\}$.

Define a map Q from $\mathcal{P}$-intervals to finitely generated graded $F[t]$-modules by $Q(i,j) = \sum^i F[t]/(t^{j-i})$. If $j = +\infty$, then $Q(i, +\infty) = \sum^i F[t]$. For a finite set of intervals $\mathcal{S} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$, we extend the definition of Q as:

$$Q(\mathcal{S}) = \bigoplus_{k=1}^{m} Q(i_k, j_k)$$

The correspondence with the classification show that Q is a bijection between finite sets of $\mathcal{P}$-intervals and the isomorphism classes of persistence modules of finite type over a field F.

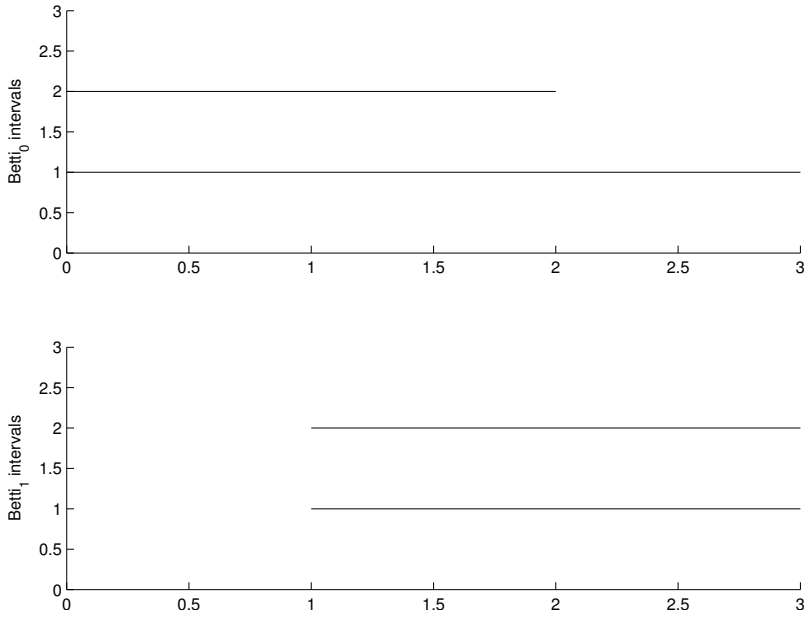

FIGURE 3. Plot of $\mathcal{P}$-intervals corresponding to the filtered simplicial complex from figure 2. The 0-intervals are $(0, \infty)$, $(0, 2)$, and the 1-intervals are $(1, \infty)$, and $(1, 3)$.

5.3. **The Persistence Algorithm.** In this section I will give an overview of the persistence algorithm over a field F, using the theory from the previous section. For details, consult [10]. The input to this algorithm will be a finite simplicial complex K equipped with a filtration. The output will be a finite set of $\mathcal{P}$-intervals as defined above which characterize the persistence module (the homology of the filtered complex) up to isomorphism.

Recall that since we are working over a field, the homology groups are in fact vector spaces. Further, by the structure theorem, there exists a basis for the persistence module which gives compatible bases for all of these vector spaces. It will be up to the algorithm to find a description for this.

The first step in the derivation of the algorithm is to represent the boundary operator $\partial_n : C_n \to C_{n-1}$ relative to the standard basis for C_n and a homogeneous one for Z_{n-1}. Note that relative to homogeneous bases, a matrix representation M_n of ∂_n has the property that:

$$\deg \hat{e}_i + \deg M_k(i, j) = \deg e_j$$

For example, consider again, the filtered complex given in figure (2). ∂_1 (with coefficients in $\mathbb{Z}_2$) is given by the matrix:

$$\partial_1 = \left(\begin{array}{c|ccccccc} & ab & ac & df & ef & bc & de & cd \\ \hline f & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ e & 0 & 0 & 0 & 1 & 0 & t & 0 \\ d & 0 & 0 & 1 & 0 & 0 & t & t^2 \\ c & 0 & 1 & 0 & 0 & t & 0 & t^2 \\ b & 1 & 0 & 0 & 0 & t & 0 & 0 \\ a & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

To arrive at the desired representation of ∂_n, we proceed inductively. The base case is simple as it is just the standard matrix representation of ∂_1. In the inductive step, we assume that a matrix M_n for ∂_n relative to a standard basis for C_n, and a homogeneous one for Z_{n-1} is given. The objective is to compute a basis for Z_n and a matrix M_{n+1} for ∂_{n+1} relative to the standard basis for C_{n+1} and the computed basis.

The motivation for this step is the reduction algorithm for homology which was described in section 2, where one computes M_n with respect to a nicer basis. One key distinction though, is that we will be able to accomplish our goal via only column operations. Instead of reducing the matrix completely to its (Smith) Normal form using both row and column operations, we can get "halfway" there by using *only* column operations (1) and (3) to reduce M_n to its *column-echelon form*, $\tilde{M}_n$. Column-echelon form is a lower staircase form as in the following example.

$$\tilde{M}_n = \begin{pmatrix} \mathbf{*} & 0 & 0 & 0 & 0 & 0 \\ * & 0 & 0 & 0 & 0 & 0 \\ * & \mathbf{*} & 0 & 0 & 0 & 0 \\ * & * & \mathbf{*} & 0 & 0 & 0 \\ * & * & * & \mathbf{*} & 0 & 0 \end{pmatrix}$$

All landings in the staircase have a width of one, and any non-zero element must lie below the staircase. The boldface elements in the example denote *pivot* elements, and the rows (columns) that they occur in are called *pivot rows (columns)*. The algorithm to change a matrix to its column-echelon form is simply Gaussian Elimination using only column operations of type (1) and (3).

In particular, the pivot elements are exactly the diagonal elements of the (Smith) Normal Form for M_n, which by section 2, indicated the presence of torsion and free summands in homology. The only difference here is that now these elements do not correspond to the component homology groups, but rather, they give torsion and free summands in the persistence module, which is all we need for the desired pairing.

Lemma 5.6. *Let $\tilde{M}_n$ be the column-echelon form for ∂_n relative to bases $\{e_j\}$ and $\{\hat{e}_i\}$ for C_n and Z_{n-1} respectively. If row i has pivot $M_n(i,j) = t^n$, it contributes the summand $\Sigma^{deg\ \hat{e}_i} F[t]/(t^n)$ to the description of H_{n-1}. Otherwise, it contributes the summand $\Sigma^{deg\ \hat{e}_i} F[t]$. In the language of $\mathcal{P}$-intervals, we get the pairs $(deg\ \hat{e}_i, deg\ \hat{e}_i + n)$ and $(deg\ \hat{e}_i, \infty)$ respectively for H_{n-1}.*

The basis elements corresponding to the non-pivot columns of the column echelon form comprise a basis for Z_n. From here, it is easy to obtain a matrix representation for ∂_{n+1} with respect to this basis.

Lemma 5.7. *To represent ∂_{n+1} by a matrix with respect to the computed basis for ∂_n as above, first let M_{n+1} be the matrix for ∂_{n+1} with respect to the standard basis. The desired representation is given by deleting the rows of M_{n+1} corresponding to the pivot columns of $\tilde{M}_n$.*

Proof. There are two main observations. The first one is that since $\partial_n \circ \partial_{n+1} = 0$, the relationship $M_n \cdot M_{n+1} = 0$ is not affected by elementary operations on the basis elements. Second since the domain of M_n is the codomain of M_{n+1}, any column operation on M_n also gives a corresponding row operation on M_{n+1}. Since $M_n \cdot M_{n+1} = 0$, the row operations corresponding to the column operations which brought M_n to its column echelon form must zero out the rows in M_{n+1} corresponding to the pivot columns in $\tilde{M}_{n+1}$.

To be precise, consider the type (3) column operation, which is the only one we use here which changes values in the matrix. The operation replaces a column i by $\text{col } i + q \cdot \text{col } j$. The corresponding row operation is to replace row j by $\text{row } j - q \cdot \text{row } i$. This operation only changes row j (not row i), but row j is eventually zero, and so we see that it is sufficient to simply delete these rows. □

The persistence algorithm is based on these two lemmas that show that a full reduction to normal form is unnecessary and that only column operations are needed. The full algorithm is detailed in [10], and has a worst-case complexity of $O(m^3)$ where m is the total number of simplices in the filtration. I will now give two interesting applications of persistence.

5.4. **Examples.**

Example 5.8 (The Rips Complex)**.** Consider a finite set of points, X, which have been sampled from a subspace $\mathbb{X} \subset \mathbb{R}^n$ (say, a manifold). If the sampling is sufficiently dense, we might hope to calculate topological information about $\mathbb{X}$ using only the *Point Cloud Data set* (PCD), X.

To capture this information about the underlying space, we can build a *Rips* complex. This simplicial complex $R_\epsilon(X)$ is constructed by declaring a set of vertices $V = \{v_0, \ldots, v_k\} \subset X$ to span a k-simplex σ whenever $d(v_i, v_j) \leq \epsilon \ \forall v_i, v_j \in V$. The vertex set of $R_\epsilon(X)$ is of course, the points of X. Notice that when $\epsilon_1 < \epsilon_2$, there is an inclusion $R_{\epsilon_1}(X) \hookrightarrow R_{\epsilon_2}$. Therefore, we obtain a persistence complex for every increasing sequence of nonnegative real numbers $\{\epsilon_i\}_{i \geq 0}$ [10].

Example 5.9 (The Filtered Tangent Complex)**.** Consider a subset $X \subset \mathbb{R}^n$. It is possible to construct a topological space, namely, the tangent complex, whose homotopy type is sensitive to geometric features that might not necessarily be topological. We define $T^0(X) \subset X \times S^{n-1}$ by

$$T^0(X) = \left\{ (x, \zeta) \mid \lim_{t \to 0} \frac{d(x + t\zeta, X)}{t} = 0 \right\}$$

The tangent complex, $T(X)$, is defined to be the closure of $T^0(X)$. $T(X)$ can be given a filtration parametrized by curvature, $\kappa(x)$. Let $T^0_\epsilon(X) \subset T^0(X)$ be the subset consisting of points (x, ζ) such that $\kappa(x) < \epsilon$. As before, we define

$T_\epsilon(X) = cl(T^0_\epsilon(X))$. The family of spaces $\{T_\epsilon(X)\}_{\epsilon>0}$ is called the filtered tangent complex, $T^{filt}(X)$.

Computing persistent homology on $T^{filt}(X)$ yields a set of intervals which can be used as a compact geometric shape descriptor [2, 3].

In the next section, we will define the algebraic mapping cylinder, and consider applying the persistence algorithm to a filtered version of this complex.

6. The Algebraic Mapping Cylinder

A common technique in algebraic topology used to study maps is to associate a space to it and to examine its topology. One such space is the mapping cylinder (there are others, like the mapping cone), and for a continuous map $f : X \to Y$, it is the identification space obtained by taking the disjoint union $(X \times I) \coprod Y$ and identifying the points $(x, 1) \in X \times I$ with corresponding points $f(x) \in Y$. This space is denoted by M^f. The two main facts about M^f is that it holds an embedded copy of X, but deformation retracts onto the subspace Y [7].

We can use this type of space to study the Alexander-Whitney chain map, but first it will be necessary to define a more algebraic notion of the mapping cylinder because we would like to explicitly define the mapping cylinder in terms of the chain complexes associated with X and Y. Given a chain map $f : A \to B$ between two chain complexes A and B, we define the *algebraic mapping cylinder* M^f as the chain complex ([4]):

$$M^f_n = A_{n-1} \oplus A_n \oplus B_n$$

with boundary maps given by $\partial_n : M^f_n \to M^f_{n-1}$, where

$$\partial^{M^f}_n = \begin{pmatrix} -\partial^A_{n-1} & 0 & 0 \\ -id^A & \partial^A_n & 0 \\ f & 0 & \partial^B_n \end{pmatrix}$$

Intuitively if $A = C_*(X)$ and $B = C_*(Y)$, then the A_n and B_n summands represent the X and Y, but the A_{n-1} summand represents the simplices of $X \times I$, and so they are shifted up by one dimension and attached to Y by f. We can easily check that this is indeed a chain complex using the fact that $\partial^2 = 0$ for A and B and that f and the identity both induce chain maps:

$$\begin{aligned} \partial^{M^f}_{n-1} \circ \partial^{M^f}_n &= \begin{pmatrix} -\partial^A_{n-2} & 0 & 0 \\ -id^A_{n-2} & \partial^A_{n-1} & 0 \\ f_{n-2} & 0 & \partial^B_{n-1} \end{pmatrix} \begin{pmatrix} -\partial^A_{n-1} & 0 & 0 \\ -id^A_{n-1} & \partial^A_n & 0 \\ f_{n-1} & 0 & \partial^B_n \end{pmatrix} \\ &= \begin{pmatrix} \partial^A_{n-2} \circ \partial^A_{n-1} & 0 & 0 \\ id^A_{n-2} \circ \partial^A_{n-1} - \partial^A_{n-1} \circ id^A_{n-1} & \partial^A_{n-1} \circ \partial^A_n & 0 \\ -f_{n-2} \circ \partial^A_{n-1} + \partial^B_{n-1} \circ f_{n-1} & 0 & \partial^B_{n-1} \circ \partial^B_n \end{pmatrix} \\ &= 0 \end{aligned}$$

Furthermore, there is a chain homotopy between the identity map on M^f and a map that embeds A in M^f (by f) and restricts to the identity on B, expressing the fact that the mapping cylinder is homotopy equivalent to Y and thus will have the same homology. The chain homotopy is given explicitly by $H : M^f_n \to M^f_{n+1}$ where:

$$H = \begin{pmatrix} 0 & id^A_n & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and is the diagonal operator in the following commutative diagram:

$$\begin{array}{ccccccccc} \longrightarrow & M^f_{n+1} & \xrightarrow{\partial} & M^f_n & \xrightarrow{\partial} & M^f_{n-1} & \longrightarrow \\ & \downarrow & \swarrow & \downarrow & \swarrow & \downarrow & \\ \longrightarrow & M^f_{n+1} & \xrightarrow{\partial} & M^f_n & \xrightarrow{\partial} & M^f_{n-1} & \longrightarrow \end{array}$$

To check that this is a chain homotopy:

$$\begin{aligned} \partial^{M^f}_{n+1} H_n + H_{n-1}\partial^{M^f}_n &= \begin{pmatrix} 0 & -\partial^A_n \circ id^A_n & 0 \\ 0 & -id^A_n & 0 \\ 0 & f_n & 0 \end{pmatrix} + \begin{pmatrix} -id^A_{n-1} \circ id^A_{n-1} & id^A_{n-1} \circ \partial^A_n & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} -id^A_{n-1} & 0 & 0 \\ 0 & -id^A_n & 0 \\ 0 & f_n & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & f_n & id^B_n \end{pmatrix} - \begin{pmatrix} id^A_{n-1} & 0 & 0 \\ 0 & id^A_n & 0 \\ 0 & 0 & id^B_n \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & f_n & id^B_n \end{pmatrix} - id^{M^f}_n \end{aligned}$$

6.1. **The Kernel of a Chain Map.** Let $f : A \to B$ be a chain map and let $f_* : H_*(A) \to H_*(B)$ be the map that it induces in homology. In this section, I will present a strategy to find the dimension of the kernel of f_*. To solve the problem, the solution will be to apply the persistence algorithm to the algebraic mapping cylinder in an unconventional way. Instead of looking for long lasting features in this filtered space, we will instead be concerned with the short lived ones.

Let M^f be the associated algebraic mapping cylinder for f. We now assign a filtration F to the basis elements for M^f by:

$$F(\sigma) = \begin{cases} 0 & \text{if } \sigma \in A \\ 1 & \text{if } \sigma \in B \end{cases}$$

Now if we run the persistence algorithm on M^f with this filtration, then the output will yield $\mathcal{P}$-intervals of the forms $(0, \infty)$, $(1, \infty)$, or $(0, 1)$.

Lemma 6.1. *The number of $\mathcal{P}$-intervals of the form $(0, 1)$ from the output of the algorithm is exactly the dimension of* $\ker f_*$.

Proof. By the correspondence from section 5, each $(0, 1)$-interval corresponds to a summand of the form $F[t]/(t)$. Thus, it corresponds to a basis element for the homology vector space that exists at time 0, but not at time 1. Now the basis elements that exist at time 0 are all basis elements for $H_*(A)$ by the way the filtration is defined. The subset of these which do not persist are those that make up the kernel of the inclusion map $H_*(A) \hookrightarrow H_*(M^f)$, which by the chain homotopy from the previous section is the same as the kernel of $f_* : H_*(A) \to H_*(B)$. □

Therefore, the strategy to compute the set of invariants: $\dim(I/I^2)$, $\dim(I^2/I^3)$,... will be as follows:

Given a simplicial complex C,

(1) Construct the tensor product complex $C_* \otimes C_*$.
(2) Construct the Reduced Alexander-Whitney Chain Maps $\bar{\tau} : C \to C_* \otimes C_*$.

(3) Construct the Algebraic Mapping Cylinder $M^{\bar{\tau}}$ for the Reduced Alexander Whitney map.
(4) Filter M^A by F.
(5) Run the persistence algorithm on the chain complex $M^{\bar{\tau}}$ with filtration F to yield a set of $\mathcal{P}$-intervals.
(6) Set $\dim(I/I^2) = \dim\ker\bar{\tau}_* = (\#\mathcal{P}\text{-intervals of the form } (0,1))$.

The calculation is similar for I^n/I^{n+1} but with a variant of the Alexander-Whitney map. For example, to compute I^2/I^3, we would use

$$\bar{\tau}\otimes id : \bar{C}_*\otimes\bar{C}_*\to\bar{C}_*\otimes\bar{C}_*\otimes\bar{C}_*$$

Of course one can always approximate the mapping cylinder with a simplicial complex and so we could have approached the problem by constructing explicit triangulations for the the mapping cylinder (given in Hatcher section 2.C [7]) but the algebraic formulations allow for computations to be done on far fewer basis elements.

7. Plex

Plex is a set of computational tools developed by Vin de Silva and Patrick Perry with a Matlab frontend for analyzing the topology of data sets which can be given the structure of a simplicial complex. Specifically, it provides methods for constructing finite simplicial complexes with associated boundary maps, and topological persistence methods.

I have implemented methods extending Plex for constructing an algebraic mapping cylinder for the Alexander-Whitney chain map. To do this, it was first necessary to extend Plex to handle general (finitely generated) chain complexes.

The advantage that simplicial complexes have is that they have an elegant combinatorial representation as a data structure and the fact that each simplex knows its faces and cofaces makes it unnecessary to store explicit boundary maps in memory.

While they can indeed capture a large class of spaces up to homotopy equivalence, they often must do so inefficiently and are not so easy to construct. Furthermore, one frequently works with complexes which have been defined algebraically and did not come directly from a geometric object. The tensor product of chain complexes is one example, and the algebraic mapping cylinder is another.

With a general chain complex, there is no notion of a face or coface, so at each dimension, it is just an ordered list of basis elements for the free abelian group. In the implementation (in C++), the *ChainComplex* class can be constructed in a number of ways. The most interesting will be by direct sum, tensor product, or mapping cylinder. These constructions can be concatenated to build arbitrarily large complexes. It can also be constructed by a *SimplicialComplex*, in which case it holds pointers to each simplex in the complex and constructs explicit boundary maps at each dimension up to the dimension of the complex. For storage efficiency, sparse matrices are used for these maps. Another important data structure which is derived from the ChainComplex class is the *FilteredChainComplex*, which simply maps each basis element in the ChainComplex to some filtration value.

Some extra functions were added to the existing sparse matrix library used in Plex. Most of these were helper functions for piecing linear maps together from maps of subspaces of the domain or maps to subspaces of the range. For the tensor product, it was necessary to implement a Kronecker product operation:

Given linear maps represented by matrices A, and B, the tensor product map is given by the Kronecker product of the matrices,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1m}B \\ a_{21}B & a_{22}B & \dots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \dots & a_{nm}B \end{pmatrix}$$

where $A = (a_{ij})$.

A tensor product of chain complexes can be constructed in the following way. Given chain complexes A_*, and B_*, we define the tensor product chain complex by:

$$(A_* \otimes B_*)_n = \bigoplus_{i=0}^{n} A_i \otimes B_{n-i}$$

where $A_i \otimes B_{n-i}$ is a tensor product of vector spaces since we always work over a field. If the basis elements of A_i and B_{n-i} are enumerated as $\{\alpha_1, \dots, \alpha_k\}$ and $\{\beta_1, \dots, \beta_l\}$ respectively, the basis for $A_i \otimes B_{n-i}$ is enumerated as $\{\alpha_1 \otimes \beta_1, \dots, \alpha_1 \otimes \beta_l, \alpha_2 \otimes \beta_1, \dots, \alpha_2 \otimes \beta_l, \dots\dots, \alpha_k \otimes \beta_1, \dots, \alpha_k \otimes \beta_l\}$. With this enumeration, given linear maps ϕ, ψ whose domains are A_i and B_{n-i} respectively, the matrix representation for the tensor product map $\phi \otimes \psi$ is given by their Kronecker product as above.

The boundary map $\partial : (A_* \otimes B_*)_n \to (A_* \otimes B_*)_{n-1}$ is specified on each summand $A_i \otimes B_{n-i}$ by:

$$\partial(\alpha \otimes \beta) = \partial_A \otimes \beta + (-1)^i \alpha \otimes \partial_B \beta$$

for $\alpha \in A_i$, and $\beta \in B_{n-i}$.

For example, the matrix for $\partial_4 : (A \otimes B)_4 \to (A \otimes B)_3$ is:

$$\partial_4 = \begin{pmatrix} id_0^A \otimes \partial_4^B & \partial_1^A \otimes id_3^B & 0 & 0 & 0 \\ 0 & -id_1^A \otimes \partial_3^B & \partial_2^A \otimes id_2^B & 0 & 0 \\ 0 & 0 & id_2^A \otimes \partial_2^B & \partial_3^A \otimes id_1^B & 0 \\ 0 & 0 & 0 & -id_3^A \otimes \partial_1^B & \partial_4^A \otimes id_0^B \end{pmatrix}$$

Since each summand maps to only two summands below it, we see one reason why it is a good idea to use sparse matrix representations here.

With the ability to handle tensor products of chain complexes, Plex now can compute the homology of a product space. Of course it is possible to simply give the product space the structure of a simplicial complex, but in general, the tensor product complex will be smaller.

Example 7.1. In the simple example $I \times I$, we can already see that this is true. Let I_1 be the unit intervals with vertices α_1 and α_2 with edge $(\alpha_1\ \alpha_2)$. Let I_2 be the unit interval with vertices β_1 and β_2 with edge $(\beta_1\ \beta_2)$. Now a triangulation of $I_1 \times I_2$ will result in 4 vertices, 5 edges (one diagonal), and 2 faces (one on each side of the diagonal edge). The tensor product however, will have 4 vertices, 4 edges, and 1 face.

$$\begin{aligned} (I_1 \otimes I_2)_0 &= \langle \alpha_1 \otimes \beta_1, \alpha_1 \otimes \beta_2 . \alpha_2 \otimes \beta_1, \alpha_2 \otimes \beta_2 \rangle \\ (I_1 \otimes I_2)_1 &= \langle \alpha_1 \otimes (\beta_1\ \beta_2), \alpha_2 \otimes (\beta_1\ \beta_2), (\alpha_1\ \alpha_2) \otimes \beta_1, (\alpha_1\ \alpha_2) \otimes \beta_2 \rangle \\ (I_1 \otimes I_2)_2 &= \langle (\alpha_1\ \alpha_2) \otimes (\beta_1\ \beta_2) \rangle \end{aligned}$$

7.1. **Results.** We now revisit the first example of $X = S^1 \times S^1$, and $Y = S^2 \vee S^1 \vee S^1$. After defining filtered chain complexes, and the reduced Alexander-Whitney chain map, we can use the following code in C++ to run the algorithm for computing $\ker(I/I^2)$ on the simplicial complex, *scomplex*. In the implementation, persistence is computed with coefficients in $\mathbb{Z}_2$.

```
ReducedAlexanderWhitneyMap<Alloc> F(scomplex);
FilteredChainComplex<Alloc> *fccomplex =
            new FilteredChainComplex<Alloc>(F);
Pivot_space_t * pairs = ComputeMCPersistence(fccomplex);
```

The output is a long list of intervals, but tallying the ones that take the form (0,1) in each dimension yields [0,2,0,0] for $S^1 \times S^1$, and [0,2,1,0] for $S^2 \vee S^1 \vee S^1$. As the program shows, all cohomology classes for the torus can be written as cup products of lower dimensional cocycles, while the product of any two cocycle classes in $H^1(Y)$ must necessarily be zero. Hence, $S^1 \times S^1$ and $S^2 \vee S^1 \vee S^1$ are not homotopy equivalent.

For a larger example, consider comparing $X = S^1 \times S^2$ and $Y = S^1 \vee S^2 \vee S^3$. Both complexes have the following homology groups,

$$H_n(X; F) = H_n(Y; F) = \begin{cases} F & 0 \leq n \leq 3 \\ 0 & 4 \leq n \end{cases}$$

But, running the algorithm on triangulations of X and Y give invariants [0,1,1,0,0,0] and [0,1,1,1,0,0] respectively, and so these spaces are not homotopy equivalent either.

8. Conclusions

In this paper, I have presented a method for computing a set of invariants which describe the cup product structure of a space. The motivation behind this is that the cohomology ring of a space is a finer invariant than homology groups. In certain classes of spaces, there are strong restrictions on the form that the cup product structure of the space can take. For example, Poincare Duality forces much structure onto the cohomology ring of a manifold.

The method for computing these invariants is based on the fact that one can study cup products in cohomology by studying the Alexander-Whitney map induced in homology. To determine the dimension of the kernel of this map, we constructed an algebraic mapping cylinder associated to it and gave it a particular filtration such that running the persistence algorithm yielded the number of basis elements in homology that died under the Alexander-Whitney chain map.

On a more computational level, I have explored the possibilities of doing chain complex computations within the software package, PLEX, as opposed to working with the standard simplicial complex. Simplicial complexes are convenient to work with in many ways. They are conceptually more intuitive by their geometric definition, and the boundary maps are easily specified as the set of faces for each individual simplex (with sign if $F \neq \mathbb{Z}_2$). Unfortunately, due to the various constraints that define a simplicial complex, these spaces are prone to getting very large. To fix this, there are several things that can be tried. A *simplicial set* representation is one possibility, where one drops the requirement that a simplex be uniquely determined by its vertices. In this paper, I have outlined automated methods to construct mapping cylinders, direct sums (associated to disjoint union

or wedge products of spaces), and tensor products (associated to product spaces). While all of these types of spaces can be given triangulations, the algebraic chain complex representation can be much more compact.

Future work on this project might involved searching for faster, more efficient algorithms for manipulation sparse matrices in order to improve the software implementation. Another direction to explore is tracking cup product structures along a filtered simplicial complex. With our method, there would then be two persistence dimensions to consider. One problem that remains is that the intermediate complexes can become very large, despite savings by using tensor products and algebraic formulations of the mapping cylinder. In the innocent example of running the algorithm on $S^1 \times S^2$, we see that boundary maps for the algebraic mapping cylinder can have on the order of 10,000x10,000 entries!

9. Acknowledgements

I would like to thank my advisor, Professor Gunnar Carlsson, for introducing me to Algebraic Topology. Over the course of the year, he has been patient in explaining concepts in a concrete way and pointing me to excellent sources. In working on this project, I have learned much through familiarizing myself with abstract topological concepts, then forcing myself to understand them in a concrete way by implementing these ideas on a computer.

I am also grateful to Vin de Silva and Patrick Perry for providing support and encouragement on the computational side. PLEX is a useful computational tool, and they were always willing to answer any questions I had about using or modifying it.

References

1. M.A. Armstrong, *Basic Topology* Springer-Verlag, New York, 1983
2. G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas, *Persistence Barcodes for Shapes*, International Journal of Shape Modeling (to appear)
3. A. Collins, A. Zomorodian, G. Carlsson, and L. Guibas, *A Barcode Shape Descriptor for Curve Point Cloud Data*, Computers and Graphics (to appear)
4. J. Davis and P. Kirk, *Lecture Notes in Algebraic Topology*, American Mathematical Society, 2001.
5. D. Dummit and R. Foote, *Abstract Algebra*, John Wiley and Sons, Inc., New York, NY, 1999.
6. M.J. Greenberg and J. Harper, *Algebraic Topology, a First Course*, Benjamin/Cummings, 1981.
7. Allen Hatcher, *Algebraic Topology*, Cambridge University Press, Cambridge, England, 2002.
8. J. J. Rotman, *An Introduction to Algebraic Topology*, Springer GTM 119, 1988.
9. James Vick, *Homology Theory*, 2nd edition, Springer GTM 145, 1994.
10. A. Zomorodian and G. Carlsson, *Computing Persistent Homology*, Discrete and Computational Geometry (to appear)

Department of Mathematics, Stanford University
E-mail address: jhuang11@stanford.edu