Persistence and Computation of the Cup Product

Andrew Yarmola
Senior Honors Thesis
Department of Mathematics, Stanford University
Advisor: Mikael Vejdemo-Johansson

June 9 2010

Abstract

In this paper we discuss the computation of the cup product for the cohomology of a finite simplicial complex over a field. Working with the theory of persistent (co)homology, we study an algebraic substructure of the cohomology ring of a simplicial complex within a filtration. We study this structure in a persistence setting, allowing us to classify it by a series of barcodes. Further, we introduce and implement several algorithms for computing the cup product for a simplicial complex within the persistent and non persistent settings.

Contents

1 Introduction

Topological data analysis is a developing field of mathematics focused on providing methods for computing topological features of data sets. A topological approach allows us to find both global and local features of complex data sets while working in an abstract mathematical framework. Homology and cohomology have been particularly powerful tools for data analysis.

In this paper, we study the computation of the cup product for the cohomology ring of a filtered simplicial complex over a finite field. A filtered simplicial complex is simply a sequence of simplicial complexes ordered by inclusion. We are interested in the analysis of both the cup product structure of the total simplicial complex, the final complex in the sequence, and the changes to this structure throughout this sequence. A filtered simplicial complex gives rise to an system of graded cohomology rings $\{R^i\}_{i\geq 0}$ with homomorphisms $R^{i+1} \to R^i$. We study the cup product by analyzing the powers of the ideal $I_i = \bigoplus_{j>0} R^i_j$ within system of graded rings.

Our results are meant to enrich the already established framework of persistent homology and cohomology [2][6][4][5][1]. Persistent homology and cohomology characterize the birth and death of topological features in a filtered simplicial complex. This framework has been applied to tackle problems such as shape recognition, data simplification, sensor network coverage, and finding local circular coordinates for data sets. By introducing the cup product, we can further enrich this framework by allowing the detection of a new range of topological features.

We begin this paper with a few motivational points and then provide the necessary background in (co)homology and the cup product in sections 2 and 4. In section 3 we review the framework of persistent (co)homology and show an algorithm for persistent cohomology. In section 5 we discuss the computation of the cup product in the persistence setting and provide our main theoretical results. In section 6 we develop several algorithms for the computation of the cup product. We discuss our implementations in section 7.

1.1 Motivation

The problem reconstructing a space from some point cloud sample has received much attention in the field of computation sciences. Algorithms for persistent homology and cohomology of simplicial complexes provide an ad-

vanced approach to high dimensional analysis and reconstruction of such samples. Given a point cloud in $\mathbb{R}^n$, there are a variety of methods of constructing a nested sequence of simplicial approximations of the data, called a filtration (e.g. Čech, Rips, or Witness complexes [2][6]). In general, a numerical parameter is varied to construct a filtration. Persistence algorithms allow us to track the evolution of topological features within the filtration and find stable global features of the data set. From this analysis, it is possible to find a range of "good" simplicial approximations of the data and get valuable topological information about its structure. In an attempt to further refine this range, we are interested in looking at the evolution of the cohomology ring in the filtration.

The structure of the cohomology ring enriches our analysis in several ways. Example 1 shows that having information about the cohomology ring allows us to distinguish between certain topological spaces with the same (co)homology. This provides us with a more precise understanding of the topological structure of the data set. In fact, we get even more structural information. The cohomology ring is closely tied to the intersections of submanifolds [7] and so provides us with an assessment of how "manifold like" our data set is. In certain cases, this ability so say something about the "manifold like" structure of our simplicial approximations should allow us to notice undesired cusping in our data set. That is, cusping would correspond to a collapse in the manifold like structure of our simplicial complexes. Such collapses could indicate that we should take a new sample of points, modify our filtration in a specific way, or allow us to choose a more preferred approximation to our data

Example 1. Consider the spaces $T = S^1 \times S^1$ and $M = S^2 \vee S^2 \vee S^1$. The cohomology of the these spaces is the same with $H^1(T\,\mathbb{F}) \simeq H^1(M;\mathbb{F}) \simeq \mathbb{F}^2$ and $H^2(T;\mathbb{F}) \simeq H^2(M;\mathbb{F}) \simeq \mathbb{F}$. These spaces have very different topological structures since T is a 2-manifold, while M is not. The cup product allows us to differentiate between these two spaces. The cup product of the non trivial cohomology classes in dimension 1 is non trivial for T, while it is trivial for M.

Examples 2. Besides allowing us to differentiate between spaces, studying the change in cup product structure throughout a filtration can be interesting. Morse theory on high dimensional manifolds is an example where understanding the cup product structure on the level sets could allow us understand the high dimensional manifold itself.

2 Background

2.1 Simplicial Complexes

A k-dimensional simplex σ, or *k-simplex*, is defined to be a set $\sigma = \{v_0, \ldots, v_k\}$, where the element of σ are called vertices. One can think of a k-simplex geometrically as the convex hull of $k+1$ affinely independent points in $\mathbb{R}^n$ where $n \geq k$. A set of points $\{v_0, \ldots, v_k\}$ in $\mathbb{R}^n$ is called affinely independent if and only if $\sum_i a_i v_i = 0$ and $\sum_i a_i = 0$ implies that $a_i = 0$ for all i. For $k = 0, 1, 2, 3$ the corresponding geometric realizations are a point, a line segment, a triangle, and a tetrahedron. A *face* of a simplex σ is a set τ with $\tau \subseteq \sigma$. We also call σ a *coface* of τ.

For a simplex, we can define a notion of *orientation* if we assign a linear ordering to the vertices. Two orderings are considred the same if the permutation taking one to the other has positive sign, and distinct if the sign is negative. That is $[v_0, \ldots, v_k] \sim [v_{\gamma(0)}, \ldots, v_{\gamma(k)}]$ if and only if $\text{sign}(\gamma) = 1$. Thus, for $k > 0$, each k-simplex can be given two possible orientations corresponding to equivalence classes of orderings. Note that an orientation on σ gives an orientation to all of its faces by suborderings. We use the $[\ldots]$ notation when listing vertices to indicate that an orientation for σ has been chosen.

A *simplicial complex* K is a set of simplices such that if $\sigma \in K$ and $\tau \subseteq \sigma$ then $\tau \in K$. This amounts to the need for all of the faces of a simplex to be included in the complex K. A *subcomplex* L of K is a subset of K that is also simplicial complex. The dimension of K is the largest dimension of a simplex in K. If K is a finite set, then K is called a finite simplicial complex. We can construct a simplicial complex from any simplex σ by taking the set of all faces of σ.

A geometric realization of K, denoted as $|K|$, can be constructed by taking geometric realization of all $\sigma \in K$ and gluing them along common faces, which are faces of the form $\sigma \cap \sigma'$. In the geometric realization, it is important that any two simplices that meet, do so at a common face. In particular, $|K| = \prod_{\sigma \in K} |\sigma| / \sim$ where $\sim$ is the equivalent relation specifying which faces are identifies. For a finite simplicial complex K, we can always construct a geometric realization in $\mathbb{R}^m$ where m is the number of 0-simplices of K. We can consider K to be a subcomplex of an m-simplex, whose geometric realization lies in $\mathbb{R}^m$.

Remark What we call a simplicial complex, some authors prefer to call

an *abstract* simplicial complex. We chose our definition to emphasize the combinatorial nature of simplicial complexes. The geometry of the simplicial complex or an embedding are not necessary for the general framework of our computation approach.

For a simplicial complex K, we can assign orientations to all of its simplices by providing a partial ordering of the 0-simplices, i.e. vertices, of K such that the restriction to the vertices of every simplex in K is linear. We call such an assignment of orientations *compatible*. If K is finite, we can simply choose a linear ordering of all the 0-simplices. The well-ordering theorem allows us to do the same for infinite simplicial complexes.

A *filtration* of a complex K is a nested sequence $S = \{K^i\}$ of simplicial complexes $\emptyset \subseteq K^0 \subseteq K^1 \subseteq \ldots \subseteq K^m = K$. In general, we may treat this sequence as infinite by letting $K^i = K$ for $i \geq m$. A simple example is given by $K^i = \{\sigma \in K \mid \dim(\sigma) \leq i\}$. We say that S is a *refinement* of T if T is a subsequence of S. We refer to any refinement of the above example as being *ordered by dimension*.

A map $f : K \to L$ between to simplicial complexes is called *simplicial* if takes 0-simplices to 0-simplexes and if $\sigma = \{v_0, \ldots, v_k\} \in K$ then $f(\sigma) = \{f(v_0), \ldots, f(v_k)\} \in L$. Note that for simplicity, we are treating 0-simplices and vertices as the same thing. An inclusion $i : K \hookrightarrow L$ is an example of a simplicial map.

2.2 Chain Complexes

A *chain complex of abelian groups* is a collection $\{C_k\}_{k\in\mathbb{Z}}$ of abelian groups along with homomorphisms $\partial_k : C_k \to C_{k-1}$ such that $\partial_{k-1} \circ \partial_k = 0$. C_k is called the k^{th}-*chain group* and its elements are called k-*chains*. The homomorphism ∂_k are called *boundary operators* for reasons that will become clear. To denote an entire chain complex we use the notation C_* or (C_*, ∂_*) if we need to specify the boundary operators.

Given a simplicial complex K with a compatible ordering, we let the $C_k(K; \mathbb{Z})$ be the free abelian group generated by k-simplices of K. The elements of $C_k(K; \mathbb{Z})$ are finite sums of the form $\sum_i a_i \sigma_i$ with $a_i \in \mathbb{Z}$ and $\sigma_i \in K$. For $\sigma = [v_0, \ldots, v_k] \in K$, we define

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

where that $\hat{v_i}$ indicates the removal of that vertex. Note that this is the alternating sum of the $(k-1)$-dimensional faces of σ, giving ∂_k the name *boundary operator*. The alternating sign is meant to account for the orientation. Since the ordering for K was compatible, $\partial_k(\sigma)$ is an element of $C_{k-1}(K;\mathbb{Z})$. As ∂_k is well defined on the generators of $C_k(K;\mathbb{Z})$, we can extend it to a homomorphism $\partial_k : C_k(K;\mathbb{Z}) \to C_{k-1}(K;\mathbb{Z})$.

Lemma 2.2.1. *For a simplicial complex K we have that $\partial_{k-1} \circ \partial_k = 0$.*

Proof. It is enough to check that this hold for generators $\sigma = [v_0, \ldots, v_K]$ of $C_k(K;\mathbb{Z})$.

$$(\partial_{k+1} \circ \partial_k)(\sigma) = \sum_{i>j} (-1)^i (-1)^j [v_0, \ldots, \hat{v_j}, \ldots, \hat{v_i}, \ldots, v_k]$$

$$+ \sum_{i<j} (-1)^i (-1)^{j-1} [v_0, \ldots, \hat{v_i}, \ldots, \hat{v_j}, \ldots v_k] = 0$$

once we pull out a negative sign from the second summand and interchange i and j. □

Now, if we let $C_k(K;\mathbb{Z}) = 0$ and $\partial_k = 0$ for $k < 0$, then $\{C_k(K;\mathbb{Z})\}_{k\in\mathbb{Z}}$ is indeed a chain complex.

Given an abelian group G, we can construct a *cochain complex* by dualizing $\{C_k(K;\mathbb{Z})\}_{k\in\mathbb{Z}}$ to $C^k(K;G) = \mathrm{Hom}(C_k(K;\mathbb{Z}), G)$ and $\delta_k = \partial_{k+1}^* : C^k(K;G) \to C^{k+1}(K;G)$, where $(\delta_k\phi)(\sigma) = \phi(\partial_{k+1}\sigma)$. Note that $\delta_k \circ \delta_{k-1} = (\partial_k \circ \partial_{k+1})^* = 0$, so the cochain complex is simply a chain complex with arrows going in a different direction. We call δ_k a *coboundary operator*.

Using the boundary and coboundary operators we identify several subgroups of $C_k(K;\mathbb{Z})$ and $C^k(K;G)$. The subgroups $Z_k(K;\mathbb{Z}) = \ker \partial_k$ and $Z^k(K;G) = \ker \delta_k$ are called *cycle* and *cocycle* groups respectively. Similarly, the $B_k(K;\mathbb{Z}) = \mathrm{im}\ \partial_{k+1}$ and $B^k(K;G) = \mathrm{im}\ \delta_{k-1}$ are *boundary* and *coboundary* groups respectively. Observe that $\partial_k \circ \partial_{k+1} = 0$ and $\delta_k \circ \delta_{k-1} = 0$ imply that

$$B_k(K;\mathbb{Z}) \subseteq Z_k(K;\mathbb{Z}) \subseteq C_k(K;\mathbb{Z})$$

and

$$B^k(K;G) \subseteq Z^k(K;G) \subseteq C^k(K;G).$$

Definition 2.2.1. *A chain map between two complexes (C_*, ∂_*) and (C'_*, ∂'_*) is a collection of homomorphisms $f_k : C_k \to C'_k$ such that $\partial_k \circ f_k = f_{k-1} \circ \partial_k$.*

A cochain map can be defined similarly. Chain maps behave well with respect to the subgroups we defined above because $f_k(\ker \partial_k) \subseteq \ker \partial'_k$ and $f_k(\text{im } \partial_{k+1}) \subseteq \text{im } \partial'_{k+1}$. This behavior allows (co)chain maps to induce maps of (co)homology.

For most of the paper, we will drop the indices on ∂ and δ for simplicity.

2.3 Simplicial Homology and Cohomology

For a simplicial complex K and an abelian group G, we define the k^{th}-*homology group* to be $H_k(K;\mathbb{Z}) = Z_k(K;\mathbb{Z})/B_k(K;\mathbb{Z})$ and the k^{th}-*cohomology group with coefficient in* G to be $H^k(K;G) = Z^k(K;G)/B^k(K;G)$. Elements of $H_k(K;\mathbb{Z})$ and $H^k(K;G)$ are called *homologous cycles* and *cocycles* respectively and are classes of (co)cycles up to (co)boundaries.

Remark. Homology and cohomology can be abstractly defined for any chain and cochain complex as simply $\ker \partial/\text{im } \partial$ and $\ker \delta/\text{im } \delta$ respectively. Note the implied indices. With this in mind, we will talk about (co)homology of a (co)chain complex.

Homology and cohomology groups are topological invariants in the sense that if there is a bijective simplical map between K and L, then the (co)homology groups for K and L are naturally isomorphic. In fact, any simplicial map $f : K \to L$ gives rise to a homomorphisms $f_* : H_k(K;\mathbb{Z}) \to H_k(L;\mathbb{Z})$ and $f^* : H^k(L;G) \to H^k(K;G)$ for all k. This homomorphism arrises from the induced (co)chain maps by f on the (co)chain complexes of K and L. The details of these facts can be found in [7]. For this paper, we are only concerned with homomorphisms induced by an inclusion $i : L \hookrightarrow K$ with regard to filtrations of simplicial complexes. We use $i_\#$ and $i^\#$ for the induced maps on the level of chains and cochains respectively. For homology, $i_\#(\phi) = \phi$ for $\phi \in C_k(K;\mathbb{Z})$. For cohomology $(i^\#\psi)(\sigma) = \psi(i(\sigma))$ for $\psi \in C^k(K;G)$. Note that a bijective simplicial map implies that there is a homeomorphism between $|L|$ and $|K|$ as topological spaces.

Intuitively, (co)homology groups indicate the presence of k-dimensional analogues of *holes*. For examples a sphere (or boundary of a tetrahedron) S^2 has $H_2(S^2) \simeq \mathbb{Z}$, indicating a 2-dimensional hole, or void, in the space. From the perspective of data analysis, these features contribute to the understanding of the data set as a whole.

In general, homology and cohomology groups do not have a trivial relationship, such as being isomorphic. However, their relationship is well understood by the universal theorem of cohomology the details of which may

be found in [7]. An important distinction to be made between the two is that cocycles are global functions $\psi : C_k(K;\mathbb{Z}) \to G$ while cycles are elements of $C_k(K;\mathbb{Z})$, creating the notion that cohomology carries more global information than homology.

2.4 (Co)homology Coefficients

Before we introduce the persistence (co)homology, we would like to address the nature of coefficients in (co)homology. As we have seen, cohomology carries a notion of coefficient in a group G. However, if $G = R$ was chosen to be a commutative ring with unity, the cochains $C^k(K;R) = \mathrm{Hom}(C_k(K;\mathbb{Z}), R)$ become modules over R, making $H^k(K;R)$ an R-module. In particular, if $R = \mathbb{F}$ is a field, then $H^k(K;R)$ is vector space. The dimension of $H^k(K;\mathbb{F})$ over $\mathbb{F}$ is called the k^{th} *betti number*. Over a field, cohomology groups can then be completely described by their betti numbers.

There is also a notion of homology with coefficients in an abelian group G. Instead of taking the free abelian group $C_k(K;\mathbb{Z})$ generated by k-simplices, we may use $C_k(K;G)$, which is the collection of all finite sums of the form $\sum_i a_i\sigma_i$ for $a_i \in G$ and σ_i k-simplices. If $G = R$ is a commutative ring with unity, then $C_k(K;R)$ is a free R-module generated by k-simplices. The homology groups with coefficient in G are denoted as $H_k(K;G)$ and are R-modules if $G = R$ and vector spaces if $G = \mathbb{F}$ is a field. Computationally, we deal with (co)homology over finite fields. In the case of fields, the universal coefficient theorem for cohomology tells us that there is a natural isomorphism $H^k(K;\mathbb{F}) \simeq \mathrm{Hom}_{\mathbb{F}}(H_k(K;\mathbb{F}),\mathbb{F})$ [7]. For a finite simplicial complex, we can also construct $\mathrm{Hom}_{\mathbb{F}}(H_k(K;\mathbb{F}),\mathbb{F}) \simeq H_k(K;\mathbb{F})$. Note that this isomorphism requires a choice of basis and is not natural. For (co)homology computations over fields, we are then simply interested in the betti numbers for a finite simplicial complex.

For a finite simplicial complex, taking coefficients in a principal ideal domain R, homology and cohomology groups are finitely generated modules whose structure we can understand by the following structure theorem.

Theorem 2.4.1. *If M is a finitely generated module over a principal ideal domain R, then M is a direct sum of cyclic R-modules. In particular,*

$$M \simeq R^d \oplus \left(\bigoplus_{i=1}^{n} R/(r_i) \right)$$

where $d, n \in \mathbb{N}$, and $r_i \neq 0, 1 \in R$ with r_i dividing r_{i+1} for all i. Further, d and the list $r_1, \ldots, r_n$ are unique.

The module M decomposes into a free module and a torsion module, the left and right hand side respectively. Uniqueness of the decomposition lets us classify M by its rank d and torsion coefficients $r_1, \ldots, r_n$. M is therefore quite similar to a vector space, except that some of its dimensions are not full copies of R but are quotient rings by principal ideals. Another way to think of about a finitely generated module over a principal ideal domain is as a quotient of a free module by a free submodule, that is $M \simeq R^{d+n}/N$ where N is the free submodule generated by $\{r_1, \ldots, r_n\}$. Computationally, we are interested in finding the rank and torsion coefficients of (co)homology groups over a principal ideal domain due to this nice structure.

3 Persistence

As we have previously mentioned, the construction of a simplicial complex from data is, in general, a parametrized process. For point cloud a data in $\mathbb{R}^n$, the Čech, Rips, and Witness algorithms allow us to produce filtrations of simplicial complexes that capture topological information along a numeric parameter. Persistent (co)homology is a computational tool that allows us to observe the variation of homological features within the filtration. In this section we provide the basic framework for persistent (co)homology from which will we build up our analysis of cup product structure in a persistence setting.

3.1 Persistent Homology

We first discuss persistent homology for a filtration $F = \{K_i\}$ as it is more natural than persistent cohomology. Computing persistent homology over a field produces a *barcode*, a set of time interval, that encodes the birth and death of features in a filtration of a simplicial complex. Intuitively, a new topological feature is born when a chain becomes a cycle in a filtration and dies when a cycle becomes a boundary. From this intuition, we have the following definition.

Definition 3.1.1. *The p^{th} persistence k^{th} homology group of K_i is*

$$H_{k,p}(K_i; G) = im\ (i^* : H_k(K_i; G) \to H_k(K_{i+p}; G)).$$

Since the map i^* is induced by inclusion, the group $H_{k,p}(K_i; G)$ is precisely those classes of cycles that are not boundaries in K_{i+p}, that is

$$H_{k,p}(K_i; G) \simeq Z_k(K_i; G)/(Z_k(K_i; G) \cap B_k(K_{i+p}; G)).$$

Over a principal ideal domain, persistent homology groups are finitely generated modules and are therefore classified by their rank and torsion coefficients. We call the rank of a persistent homology group $H_{k,p}(K_i; \mathbb{R})$ the *persistent betti number* $\beta_k^{i,p}$. Using matrix reduction algorithms it is possible to compute the persistent betti number and torsion coefficients for individual-persistent homology groups [2].

With coefficients in a field $\mathbb{F}$, the persistent homology groups become vector spaces and we may ask the question of whether we can find a compatible set of basis elements across all of the persistent homology groups of a fixed dimension k. In particular, we are interested in whether we can find a compatible basis for $H_k(K_i; \mathbb{F})$ and $H_k(K_{i+p}; \mathbb{F})$ for all i and p. To do this, the persistence in dimension k is combined into one algebraic structure called a *persistence module*.

Definition 3.1.2. *A persistence module over R is a collection of R-modules $\{M_i\}_{i\in\mathbb{N}}$ along with homomorphisms $\phi_i : M_i \to M_{i+1}$. It is of finite type if M_i is finitely generated for each i and there is n such that ϕ_j is an isomorphism for $j \geq n$.*

Definition 3.1.3. *The category of persistence modules (of finite type) over R is the collection of all persistence modules (of finite type) over R and morphisms $(M_*, \phi_*) \to (N_*, \psi_*)$, which are collections of homomorphism $f_i : M_i \to N_i$ such that $\psi_i \circ f_i = f_{i+1} \circ \phi_i$.*

The collection $\{H_k(K_i; R)\}_{i\in\mathbb{N}}$, where K_i is in a filtration F, becomes a persistence module with the maps $i^* : H_k(K_i; R) \to H_k(K_{i+1}; R)$ induced by inclusion. Note that we assume there is n such that $K_n = K_j$ for all $j \geq n$. If F is a filtration of a finite simplicial complex, then our persistence module is of finite type. We call this module the *persistence homology module* of F.

We can build persistence modules more abstractly given a collection of chain complexes $\{C_*^i\}_{i\in\mathbb{N}}$ and chain maps $f^i : \mathbb{C}_*^i \to C_*^i$

$$C_*^0 \to C_*^1 \to \dots C_*^n \to \dots$$

We can take the k^{th} homology of each chain complex to to get a persistence module with homomorphisms induced by f^i.

Definition 3.1.4. *The direct sum of two persistence modules* (M_*, ϕ_*) *and* (N_*, ψ_*) *is the persistence module* $(M \oplus N)_* = \{M_i \oplus N_i\}_{i \in \mathbb{N}}$ *with the homomorphisms* $\phi_* \oplus \psi_*$.

Over a field $\mathbb{F}$, a persistence module of finite type V_* looks like

$$V_0 \to V_1 \to \ldots \to V_n \xrightarrow{\simeq} \ldots$$

where V_i are finite vector spaces over $\mathbb{F}$. Our goal is to find a basis for all of these vector spaces that respects the maps $V_i \to V_{i+1}$. This amounts to expressing the persistence module V_* as a direct sum of persistence modules of the form

$$I_{i,j} = \{0 \to \ldots \to 0 \to \mathbb{F} \xrightarrow{\simeq} \ldots \xrightarrow{\simeq} \mathbb{F} \to 0 \to \ldots\}$$

and

$$I_{i,\infty} = \{0 \to \ldots \to 0 \to \mathbb{F} \xrightarrow{\simeq} \mathbb{F} \xrightarrow{\simeq} \ldots\}$$

where i indicates the position of the first $\mathbb{F}$ and $j-1$ the position of the last. We will find such a decomposition by transforming a persistence module into a graded module.

A *graded ring* is a ring S with a decomposition $S \simeq \bigoplus_{i \in \mathbb{Z}} S_i$, with S_i abelian groups, such that multiplication in S is defined by bilinear maps $S_n \times S_m \to S_{n+m}$. A polynomial ring is a good example where $\mathbb{F}[t] = \bigoplus_{i \geq 0} \mathbb{F}t^i$. Elements of S_i are called *homogeneous*. If S is an algebra over a ring R, then S is called a graded R-algebra. We can similarly define a graded module M over a graded ring S as an S-module with a decomposition $M \simeq \bigoplus_{i \in \mathbb{Z}} M_i$ where the action of S is given by bilinear maps $S_n \times M_m \to M_{n+m}$.

Given a persistence module (M_*, ϕ_*) over R, we collect it into an $R[t]$ graded module

$$\alpha(M_*) = \bigoplus_{i \geq 0} M_i$$

where the action of t is given by applying ϕ_*. That is,

$$t \cdot (m_0, m_1, \ldots) = (0, \phi_0(m_0), \phi_1(m_1), \ldots).$$

This construction establishes a correspondence between persistence modules of finite type and finitely generated non-negatively graded $R[t]$-modules. From [2], we have that the Artin-Rees theorem provides us with the result:

Theorem 3.1.1. *The correspondence defined by α is a functor and an equivalence of categories of persistence modules of finite type over R and finitely generated non-negatively graded modules over $R[t]$ as a graded ring.*

This correspondence allows us to use our understanding of finitely generated non-negatively graded $R[t]$-modules to analyze persistence modules.

Recall that $\mathbb{F}[t]$ is a principal ideal domain for a filed $\mathbb{F}$. Theorem 2.4.1 provides us with an understanding of the stucture of finitely generated modules over a principal ideal domain. For graded modules, the structure theorem becomes

Theorem 3.1.2. *If M is a finitely generated graded module over a graded principal ideal domain R then*

$$M \simeq \left(\bigoplus_{i=1}^{n} \Sigma^{a_i} R \right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{b_j} R/(r_j) \right)$$

where $n, m, a_i, b_i \in \mathbb{Z}$ and r_j are homogeneous elements of R such that $r_j \mid r_{j+1}$. The notation $\Sigma^a R$ means the a upward shift of the gradation of R. The numbers n, m, a_i the list of pairs (b_j, r_j) are unique.

Given a persistence homology module $M_* = \{H_k(K_i; \mathbb{F})\}_{i \in \mathbb{N}}$ for a filtration of a finite simplicial complex, we can apply the above theorem to see that

$$\alpha(M_*) \simeq \left(\bigoplus_{i=1}^{n} \Sigma^{a_i} \mathbb{F}[t] \right) \oplus \left(\bigoplus_{j=1}^{m} \Sigma^{b_j} \mathbb{F}[t]/(t^{n_j}) \right)$$

where $(r_j) = (t^{n_j})$ since r_j is a homogeneous element of $\mathbb{F}[t]$. Analyzing this structure, we see that pulling back the right hand for each j we get a subspace of M_{b_j} isomorphic to $\mathbb{F}$ that is propagated by t until $M_{b_j+n_j}$, as which point its dies. On the left hand side, for each i we get an subspace isomorphic to $\mathbb{F}$ is born in M_{a_i} and never dies is it is moved up by t in the gradation. This is precisely the decomposition we were looking for in terms of a direct sum of persistence modules. Each $\Sigma^{a_i}\mathbb{F}[t]$ piece corresponds to a persistence module $I_{a_i,\infty}$ and each $\Sigma^{b_j}\mathbb{F}[t]/(t^{n_j})$ corresponds to a persistence module I_{b_j,b_j+n_j}. Thus, we have shown that a compatible basis for the persistence module over a field exists.

A persistence interval is simply a pair (p, q) with $0 \leq p \leq q \leq \infty$ for $p, q \in \mathbb{Z} \cup \{\infty\}$. A collection (multiset) of persistence intervals is a barcode

that we will denote as $\mathscr{P}$. For each i is the decomposition above, we get a persistence interval (a_i, ∞) and for each j the interval $(b_j, b_j + n_j)$. Form this, we have that

$$(\{H_k(K_i; \mathbb{F})\}_{i \in \mathbb{N}}, i_*) \simeq \bigoplus_{(p,q) \in \mathscr{P}} I_{p,q}.$$

Thus, to each persistence module over a field we can assign a barcode that describes the structure of that module. A barcode is a very elegant and concise structure that each easy to display and encompasses the persistence of a filtration of a finite simplicial complex.

Remark. We chose to construct the associated graded module of a persistence homology module directly form the homology groups. An alternate approach is to construct graded $\mathbb{F}[t]$ modules from chain complexes themselves, that is we can define $C_k(F; \mathbb{F}) = \bigoplus_i C_k(K_i; \mathbb{F})$ for our filtration F with t acting as the inclusion homomorphism. From this, we attain a chain complex $\{C_k(F; \mathbb{F})\}_{k \in \mathbb{Z}}$ of modules over $\mathbb{F}[t]$ where the boundary maps are simply the original boundary maps on each grading. The homology of this chain complex is precisely the associated graded module to the persistence homology module. Note that each $C_k(F; \mathbb{F})$ is a free graded $\mathbb{F}[t]$ module because t acts by inclusion, this will not be the case for cohomology.

The computation of persistence intervals in all dimensions can be done concurrently using an adapted reduction algorithm on the matrices produced by boundary maps. We call this algorithm the PH algorithm. The running time is $O(n^3)$, the same as reduction to Smith Normal Form. The algorithm computes one simplex at a time by taking a sparse filtration of the input. The details and proof of validity can be found in [2]. With regard to the above remark, this algorithm relies on the fact that $C_k(F; \mathbb{F})$ are free graded $\mathbb{F}[t]$ modules.

Remark. For those familiar with spectral sequences, there is a connection to persistent homology. There is a very natural spectral sequence that arrises from a filtration with

$$E^0_{p,q} = C_{p+q}(K_p; \mathbb{F}) / C_{p+q}(K_{p-1}; \mathbb{F}).$$

If our filtration is sparse, then persistence interval corresponds to a non-trivial differential of the spectral sequence. In particular, for every interval of length l there is a non-trivial differential d_{l+1} that arrises from the computation of E^{l+1} [2]. Running the spectral sequence therefore, allows us to pick off persistence intervals in order of ascending length.

Remark. Towards the end of this section, we focused entirely on persistence over of a field. The reason for this, is that modules over an arbitrary graded ring $R[t]$ can be very complex and there is not general theorem describing their structure [2]. What we see happening in the case of fields is that the associated graded module of a persistence module decomposes into free and torsional components allowing us to find a compatible basis. For persistence modules over principal ideal domains, the homology groups themselves carry torsion, so decomposing the associated graded module is not an easy task.

3.2 Persistent Cohomology

Persistent cohomology is motivated in the same way as homology. We are interested in analyzing the how cocycles change throughout the filtration. Since cohomology is a contravariant functor, that is the inclusion $i : K_i \to K_j$ induces a reverse map $i^* : H^k(K_i; G) \to H^k(K_i; G)$, the birth and death analogies don't cleanly apply as they do for homology.

With cohomology, we are looking more at the life of a cochain in $\psi \in C^k(K; \mathbb{F})$ where $K = \cup_{K_i \in F} K_i$ for a filtration F. Recall that F stabilizes after a finite amount of time. We look at what happens to ψ as we apply i^*. The event of "death" occurs for ψ first, that is when $(i^*)^p(\psi)$ becomes a cocycle. The event of "birth" arrives later, when $(i^*)^q(\psi)$ becomes a coboundary. Since filtrations are ordered by inclusions, we are guaranteed this behaviors. A persistence interval (p, q) will correspond to a cochain that is a cocycle for time $i < q$ and a coboundary for time $i < p$. We use these slightly inverted notions of death and birth as they will make it easier to understand the persistent cohomology algorithm we present in the next section.

As with persistence homology, we are interested in finding a compatible basis for the structure:

$$H^k(K_0; \mathbb{F}) \leftarrow H^k(K_1; \mathbb{F}) \leftarrow \ldots \leftarrow H^k(K_n; \mathbb{F}) \leftarrow \ldots.$$

We can call such a structure a dual-persistence module specifically because it looks like the dual of a persistence module and $H^k(K_i; \mathbb{F}) \simeq \mathrm{Hom}_{\mathbb{F}}(H_k(K_i; \mathbb{F}), \mathbb{F})$ in a natural way. As before, we would like to decompose it into a direct sum of $I^*_{i,j}$ and $I^*_{i,\infty}$, where these are the duals of the modules discussed in the previous section.

Give a dual-persistence module (M^*, ψ^*) of finite type over $\mathbb{F}$, we can construct a graded $\mathbb{F}[u]$-module

$$\beta(M^*) = \bigoplus_{i \geq 0} M^i$$

where the action of u is given by ψ^*

$$u \cdot (m^0, m^1, \ldots) = (\psi^0(m_1), \psi^1(m_2), \ldots).$$

Theorem 3.1.2 implies that there is a unique decomposition of this module into well understood free and torsion components. As with persistence homology, we attain a barcode that describes the persistence. The decomposition provides us an expression of (M^*, ψ^*) as a direct sum of $I^*_{i,j}$ and $I^*_{i,\infty}$ modules. This proves the existence of a compatible basis for the dual-persistence module.

If we let $(M^*, \psi*) = (\{H^k(K_i; \mathbb{F})\}_{i \geq 0}, i^*)$ then it is natural to ask whether the persistence intervals of homology and cohomolgoy are the same.

Theorem 3.2.1. *The persistence intervals of homology and cohomology are the same for a filtration of a finite simplicial complex.*

Proof. This follows from the naturally of the isomorphism $H^k(K_i; \mathbb{F}) \simeq \mathrm{Hom}_{\mathbb{F}}(H_k(K_i; \mathbb{F}), \mathbb{F})$. The dual of the persistence homology module is then isomorphic to the dual-persistence cohomology module. Observe that given a direct sum decomposition of the persistence homology module

$$(M_*, i_*) = \bigoplus_{(p,q) \in \mathscr{P}} I_{p,q}$$

its dual has a natural decomposition $\bigoplus_{(p,q) \in \mathscr{P}} I^*_{p,q}$. Uniqueness in Theorem 3.1.2 guarantees that this decomposition is unique up to the set of intervals $\mathscr{P}$ and therefore the dual-persistense cohomology module has the same persistence intervals.

□

Given that the persistence intervals are the same, it would be natural to assume that there is a simple way to adapt the PH algorithm in [2] to compute the actual cocycles for the compatible basis. Running the PH algorithm

backwards would be the natural idea. However, the PH algorithm relies on the fact that

$$C_k(F;\mathbb{F}) = \bigoplus_i C_k(K_i;\mathbb{F}) \simeq \bigoplus_i \Sigma^{a_i} F[t]$$

is a free $F[t]$-module, where t acts by $i_\#$. For cohomology however, the $C^k(F;\mathbb{F}) = \bigoplus_i C^k(K_i;\mathbb{F})$ viewed as an $F[u]$-module with the action of u by $i^\#$ is not free.

To resolve this problem, [3] proposes modifying the filtration F to some F'. This modification is based on the observation that $\mathrm{Hom}_{\mathbb{F}[t]}(C_k(F;\mathbb{F});\mathbb{F}[t])$ is free. However, the cohomology of the complex $\{\mathrm{Hom}_{\mathbb{F}[t]}(C_k(F;\mathbb{F});\mathbb{F}[t])\}_{k\geq 0}\}$ produces what is called relative persistent cohomology. Using the relationship between relative cohomology and cohomology of a simplicial complex, it is possible to construct a new filtration F' such that part of the cohomology of $\{\mathrm{Hom}_{\mathbb{F}[t]}(C_k(F';\mathbb{F});\mathbb{F}[t])\}_{k\geq 0}\}$ describes the persistent cohomology of F [3].

In a computational setting, modifying the filtration of simplicial complex is costly and ineffective. In the next section we discuss an alternate algorithm for computing persistent cohomology.

3.3 Algorithm for Persistent Cohomology

Let F_K denote a filtration of a simplex K and and let $\mathbb{F}$ be a finite field. We consider F_K sparse in that $K_i \setminus K_{i-1} = \{\sigma_i\}$ has only one element. Note that every filtration has a sparse refinement. In a sparse filtration, each simplex has a unique index, which is the time at which it enters the filtration. For a simplex $\sigma \in K$ we let σ^* denote the cochain that vanishes everywhere outside of σ and $\sigma^*(\sigma) = 1$. We would also like to give indices to cochains in the following manner.

Definition 3.3.1. *For a cochain α we let the index i_α be smallest value of i for which $\alpha(\sigma_i) \neq 0$ for $\sigma_i \in F_K$.*

In our algorithm, for every simplex σ_i we may also store a pointer to a cochain $\sigma_i.cocycle$ or $\sigma_i.pivot$. These values are no essential for this algorithm, but we will use them later in our algorithms for cohomology. Note that the "replacement" step in the algorithm replaces these values as well.

We use Algorithm 1 to compute persistent cohomology. This is a modified version of the algorithm found in [4].

Algorithm 1 *PersistentCohomology*$(F_K, \mathbb{F})$

```
Let lists H, R be initially empty
for σ_i ∈ F_K do
  Let α ∈ H be the cocycle of greatest index with v_α = (δα)(σ_i) ≠ 0.
  if α exists then
    Remove α from H
    For all β ∈ H v_β = (δβ)(σ_i) ≠ 0 replace β with β − (v_β/v_α)α
    Add α to R
    Clear σ_{i_α}.cocycle { This value should be α}
    Set σ_i.pivot = α.
    Print out the interval [i_α, i)
  else
    Add σ_i^* to H
    Set σ_i.cocycle = σ_i^*
  end if
end for
```

We denote B_k, H_k, R_k to be the elements of dimension k in $\delta R = \{\delta\alpha \mid \alpha \in R\}, H, R$ respectively. After the algorithm terminates, the array H_k contains representatives a basis for $H^k(K;\mathbb{F})$ for every k. In fact, at any point in time j in the algorithm, H lists a basis for the cohomology of the simplicial complex $K_j = \{\sigma_i \in F_K \mid i \leq j\}$. Further, B_k is a basis for the coboundary group $B^k(K_j;\mathbb{F})$. We make this precise with the following proof of algorithm:

Theorem 3.3.1. *Let $K_j = \{\sigma_i \in F_K \mid i \leq j\}$. After the algorithm runs for $\sigma_j \in F_K$, the following are true:*

1. *H_k contains a list of representatives for a basis for $H^k(K_j;\mathbb{F})$.*
2. *B_k is a basis for $B^k(K_j;\mathbb{F})$.*
3. *$H_k \cup B_k \cup R_k$ is a basis for $C^k(K_j;\mathbb{F})$.*

Proof. We present a proof by induction on j. For $j = 1$, we have that $H = \emptyset$ and σ_1^* is added to H at the end of the loop. It is clear that the three statements hold as $H^0(K_1 = \{\sigma_1\};\mathbb{F}) \simeq \mathbb{F}$ and is generated by σ_1.

Assuming our theorem holds for $j-1$, we prove it holds for j. We have that $K_{j-1} \cup \{\sigma_j\} = K_j$. Let σ_j be of dimension d. We let H', R' denote lists

we attain for K_j and H, R the lists we have for K_{j-1}. We also let $\mathbb{F}S$ indicate the vector space spanned by S.

Since $\sigma \notin K_{j-1}$ is follows that $(\delta\sigma_j^*)(\alpha) = \sigma_j^*(\partial\alpha) = 0$, so σ_j^* is a cocycle and $\dim_{\mathbb{F}} Z^d(K_j;\mathbb{F}) = \dim_{\mathbb{F}} Z^d(K_{j-1};\mathbb{F}) + 1$. Let us first assume that σ_j^* is a coboundary. Then, there is some cochain $\gamma \in C^{d-1}(K_j;\mathbb{F}) = C^{d-1}(K_{j-1};\mathbb{F})$ with $\delta\gamma = \sigma^*$. Item three of our theorem implies that $\gamma = a\alpha + b\beta + r\rho$ for some $\alpha \in \mathbb{F}H_{d-1}$, $\beta \in \mathbb{F}B_{d-1}$, $\rho \in \mathbb{F}R_{d-1}$ and $a, b, r \in \mathbb{F}$. Thus $\sigma^* = a\delta\alpha + r\delta\rho$ as $\delta\beta = 0$. If $(\delta\alpha)(\sigma) = 0$ for all $\alpha \in \mathbb{F}H_{d-1}$ then $\sigma^* \in \delta(\mathbb{F}R_{d-1})$. However, since σ^* increased the dimension of d-cocycles this would increase the dimension of R_{d-1}, which is not possible. Therefore, there must be some $\alpha \in \mathbb{F}H_{d-1}$ for which $(\delta\alpha)(\sigma) \neq 0$. We can can guarantee that we can pick $\alpha \in H_{d-1}$ of highest index for which this holds. Since $(\delta\alpha)(\sigma) \neq 0$, α is no longer a cocycle. Since H_{d-1} was a set of representatives for the basis for $H^{d-1}(K_{j-1};\mathbb{F})$, if we remove α from H_{d-1} and adjust all other $\beta \in H_{d-1}$ which do not vanish on σ as in the algorithm, we attain a new set representatives H'_{d-1} for a basis of $H^{d-1}(K_j,\mathbb{F})$. Further, $\delta\alpha = c\sigma^*$ is a new basis element for the d-coboundaries. It follows that we have a new H'_{d-1} that has one element less, $B'_d = B_d \cup \{\delta\alpha\}$, $R'_{d-1} = R_{d-1} \cup \{\alpha\}$ with other parts of the lists unchanged. It is clear that the three items of the theorem hold for these new lists.

If σ^* is not a coboundary, it a new basis element for $H^d(K_j;\mathbb{F})$, so we have $H'_d = H_d \cup \{\alpha\}$. Again, the conclusions of the theorem hold and so our inductive step is verified, completing the proof. □

For the computation of persistent cohomology, we do not need to keep track of the lists B and R in general, we do so for the convince of the reader and as a baseline for our future algorithms.

Corollary 1. *Algorithm 1 produces the persistence intervals for the filtration F_K.*

Proof. Every time the algorithm prints an interval at time j, we have that a cochain α has stopped being a cocycle in any later step in the filtration. The cochain α is a non-bounding cocycle when it was first created as an indication function σ_i^*. Since α is chosen to be the youngest cochain that has ceased being a cocycle we have

$$\alpha = \sigma_i^* + \sum_{i<l<j} b_l \sigma_l^*.$$

So $i_\alpha = i$ and $[i_\alpha, j)$ is indeed a persistence interval.

□

As a last remark, we would like to discuss the values $\sigma_i.cocycle$ and $\sigma_i.pivot$. First note that no simplex will ever have both values set. Also, the value does not need to be set. At time after at the end of the main for-loop of the algorithm, the list of all cochains for $\sigma_i.cocycle$ that are set is equal to H. The list of all cochains $\sigma_i.pivot$ that are set is equal to R. The $\alpha = \sigma_i.pivot$ cochain correspond to the cochain that is killed by σ_i, the coboundary of this cochain is a basis element for the boundary subgroup of the dimension of σ_i. For a simplex where $\alpha = \sigma_i.cocycle$ is set, we have that $i = i_\alpha$, that is, σ_i the oldest simplex on which α does not vanish. We can think of σ_i giving birth to α as σ_i^*. This mirrors our birth and death explanation of persistence cohomology: cocycles are born, then transformed by reduction and then die when they are no longer cocycles.

4 The Cup Product

The structure of cohomology over a commutative ring with unity R allows us to define a product of two cochains. For cochains $\phi \in C^k(K;R)$ and $\psi \in C^l(K;R)$ we define a cochain $\phi \smile \psi : C_{k+l}(K;\mathbb{Z}) \to R$ by values on $\sigma = [v_0, \ldots, v_{k+l}]$ as

$$(\phi \smile \psi)(\sigma) = \phi([v_0, \ldots, v_k]) \cdot \psi([v_k, \ldots, v_{k+l}])$$

where we take multiplication in R. This product is clearly associative and distributive. It is also trivial to see that the cup product is bilinear. To show that we can extend this product to the level of cohomology groups, we must show that this product induces a product on homologous cocycles. For this, we must show that the product of two cocycles in again a cocycle and the product of a cocycle and a coboundary is a coboundary. Explicitly, we would like $(\phi + \delta\psi) \smile (\phi' + \delta\psi')$ to be of the form $\eta + \delta\xi$. The following lemma, which is essentially the Leibniz rule, gives us the result

Lemma 4.0.1. *$\delta(\phi \smile \psi) = \delta\phi \smile \psi + (-1)^k \phi \smile \delta\psi$ for $\phi \in C_k^*$ and $\psi \in C_l^*$*

Proof. For $\sigma = [v_0, \ldots, v_{k+l+1}]$ we have

$$(\delta\phi \smile \psi)(\sigma) = \sum_{i=0}^{k+1} (-1)^i \phi([v_0, \ldots, \hat{v_i}, \ldots, v_{k+1}]\psi([v_{k+1}, \ldots, v_{k+l+1}])$$

$$(-1)^k(\phi \smile \delta\psi)(\sigma) = \sum_{i=k}^{k+l+1} (-1)^i \phi([v_0, \ldots, v_k])\psi([v_k, \ldots, \hat{v_i}, \ldots, v_{k+l+1}])$$

Adding these two equations, we can see that the last term of the first sum cancels the first term of the second sum. What remains is

$$(\phi \smile \psi)(\sum_{i=0}^{k+l+1} (-1)^i [v_0, \ldots, \hat{v_i}, \ldots, v_{k+l+1}]) = (\phi \smile \psi)(\partial\sigma) = \delta(\phi \smile \psi)(\sigma).$$

Thus, $\delta(\phi \smile \psi) = \delta\phi \smile \psi + (-1)^k \phi \smile \delta\psi$ as desired. □

This lemma show us that if $\delta\phi = \delta\psi = 0$ then $\delta(\phi \smile \psi) = \delta\phi \smile \psi \pm \phi \smile \delta\psi = 0$. Therefore, the cup product of two cycles is a cocycle. Further, if $\delta\phi = 0$ then $\delta(\phi \smile \psi) = \pm\phi \smile \delta\psi$ and if $\delta\psi = 0$ then $\delta(\phi \smile \psi) = \delta\phi \smile \psi$ giving us that the product of a cocycle (or coboundary) with a coboundary is again a coboundary. These observation allow us to extend the cup product to a product

$$\smile : H^k(K; R) \times H^l(K; R) \to H^{k+l}(K; R).$$

The following lemma shows that the cup product is indeed a topological invariant

Lemma 4.0.2. *For a simplicial map $f : K \to L$ the induced maps $f^* : H^k(L; R) \to H^k(K; R)$ satisfy $f^*(\alpha \smile \beta) = f^*(\alpha) \smile f^*(\beta)$.*

Proof. Taking representatives $\phi \in \alpha$ and $\psi \in \beta$, we have that

$$(f^\# \phi \smile f^\# \psi)(\sigma) = (f^\# \phi)([v_0, \ldots, v_k])(f^\# \psi)([v_k, \ldots, v_k + l]) =$$

$$= \phi(f[v_0, \ldots, v_k])\psi(f[v_k, \ldots, v_{k+l}]) = (\phi \smile \psi)(f\sigma) = f^\#(\phi \smile \psi)(\sigma).$$

where we use the fact that $f[v_0, \ldots, v_{k+l}] = [f(v_0), \ldots, f(v_{k+l})]$.

□

With regard to inclusions, this lemma will be important in analyzing the change in cup product structure in a filtration.

A natural question to ask is whether the cup product is commutative. The following theorem answers the question:

Theorem 4.0.2. *The identity $\alpha \smile \beta = (-1)^{kl}\beta \smile \alpha$ holds for all $\alpha \in H^k(K; R)$ and $\beta \in H^l(K; R)$ over a commutative ring R*

Proof. See [7]. The proof relies on showing that the permutation map

$$[v_0, \ldots, v_n] \to [v_n, \ldots, v_0]$$

induces a chain map that is chain homotopic to the identity. Once this fact is shown, it is easy to see that $\alpha \smile \beta$ and $\beta \smile \alpha$ differ up to a sign by the permutation of the vertices by the map above.

□

We will refer to this property as *graded-commutative*, though some authors choose to simply use commutative. Note that for $R = \mathbb{Z}_2$ the cup product becomes commutative in a traditional sense.

Since we will be dealing with arbitrary cup products in this paper, let us make a few remarks about notation. The *length* of a cup product $\alpha_1 \smile \ldots \smile \alpha_n$ is the number of constituents, in this case n. We will also use the notation $\smile_i \alpha_i$ and $\smile_{i=1}^n \alpha_i$ for arbitrary and fixed length cup products respectively.

4.1 The Cohomology Ring

In this section, we introduce the cohomology ring and discuss the ideal $I = \bigoplus_{k>0} H^k(K; \mathbb{F})$ as the major focus of our study.

The cohomology ring is defined as

$$H^*(K; R) = \bigoplus_i H^i(K; R)$$

with the cup product providing the bilinear maps

$$\smile : H^k(K; R) \times H^l(K; R) \to H^{k+l}(K; R).$$

If we take the scalar multiplication by the coefficient ring R into account, $H^*(K; R)$ becomes a graded algebra, sometimes called the cup product algebra. This ring can in general be very complex and the question of which graded R-algebras arise as cup product algebras has not been answered [7].

Computationally, verifying whether two rings are isomorphic is not a simple task. If we restrict ourselves to coefficients in a field $\mathbb{F}$, $H^*(K; \mathbb{F})$ becomes a vector space with a gradation and a bilinear product. To understand the structure of this product, we consider the ideal

$$I = \bigoplus_{k>0} H^k(K; \mathbb{F}) \text{ of } H^*(K; \mathbb{F}).$$

This ideal gives rise to a nested sequence of vector spaces over $\mathbb{F}$

$$I \supseteq I^2 \supseteq I^3 \supseteq \dots.$$

The sequence of dimensions of the vector spaces I^i is then a topological invariant as an isomorphism of cohomology rings for two spaces would preserve this sequence. Further, we are interested in finding a compatible basis for this nested sequence of vector spaces, that is, we would like to find the basis elements of I^i/I^{i+1} for every $i > 0$. Intuitively, this basis provides us not only with the knowledge of how complex the cup product structure of the cohomology ring becomes, but gives us concrete cocycles that that have non-trivial cup products of a certain length. Since we can attain the dimensions of I^i from knowing the dimensions of I^i/I^{i+1}, we will concentrates on computing the latter.

For a filtration F of a finite simplicial complex, we are further interested in the change in cup product structure as we move up the filtration. We would like to find a homogeneous basis that is compatible with the map $H^*(K_j;\mathbb{F}) \to H^*(K_i;\mathbb{F})$ for all $i < j$ and the nestlings $I^n \supseteq I^m$. If we define

$$I_i = \bigoplus_{k>0} H^k(K_i;\mathbb{F})$$

then we look for a compatible basis across the structure

$$\begin{array}{ccccccccc}
I_0 & \longleftarrow & I_0^2 & \longleftarrow & \cdots & \longleftarrow & I_0^n & \longleftarrow & \cdots \\
{\scriptstyle i^*}\uparrow & & {\scriptstyle i^*}\uparrow & & & & {\scriptstyle i^*}\uparrow & & \\
I_1 & \longleftarrow & I_1^2 & \longleftarrow & \cdots & \longleftarrow & I_1^n & \longleftarrow & \cdots \\
{\scriptstyle i^*}\uparrow & & {\scriptstyle i^*}\uparrow & & & & {\scriptstyle i^*}\uparrow & & \\
\vdots & & \vdots & & \ddots & & \vdots & & \\
{\scriptstyle i^*}\uparrow & & {\scriptstyle i^*}\uparrow & & & & {\scriptstyle i^*}\uparrow & & \\
I_i & \longleftarrow & I_i^2 & \longleftarrow & \cdots & \longleftarrow & I_i^n & \longleftarrow & \cdots \\
{\scriptstyle i^*}\uparrow & & {\scriptstyle i^*}\uparrow & & & & {\scriptstyle i^*}\uparrow & & \\
\vdots & & \vdots & & & & \vdots & &
\end{array}$$

where the horizontal arrows indicate inclusions. We provide this analysis in the next section.

Remark. Notice that our approach to computing cup product structure is much deeper than simply computing the multiplication table for a basis for cohomology. Comparing two multiplication tables is of similar difficulty to assessing the isomorphism class of a ring, and therefore not an acceptable computational invariant. Our approach, that provides weaker information than an equivalence of multiplication tables, allows us to differentiate spaces with much more ease but still intrinsically compute the multiplication table as part of the algorithm if we wish to analyze it later.

5 Computation of the Cup Product

We develop our theory for the cup product in for a filtration in a similar fashion to persistent (co)homology by constructing a graded module that we decompose over a graded principal ideal domain.

For a filtration F of a finite simplicial complex, we know that the induced map $i^* : I_i^n \to I_{i-1}^n$ respects inclusions so we can consider

$$\bar{i}^* : I_i^n / I_i^{n+1} \to I_{i-1}^n / I_{i-1}^{n+1}.$$

Since I_i^n / I_i^{n+1} is a graded $\mathbb{F}$-module, we can take the gradation in $H^k(K_i; \mathbb{F})$. Restricting $\bar{i}^*$ we get

$$f_* : (I_i^n / I_i^{n+1}) \cap H^k(K_i; \mathbb{F}) \to (I_{i-1}^n / I_{i-1}^{n+1}) \cap H^k(K_{i-1}; \mathbb{F}).$$

From this we construct a dual-persistence module of finite type

$$D_k^* = \{(I_i^n / I_i^{n+1}) \cap H^k(K_i; \mathbb{F})\}_{i \geq 0}\}$$

with f_* as the acting homomorphism.

As with persistent cohomology, we can decompose $\beta(D_k^*)$ over $\mathbb{F}[u]$ by Theorem 3.1.2. The decomposition provides us is a collection of intervals and basis elements that are compatible across D_k^*. These basis elements are homogeneous elements of I_i^n / I_i^{n+1} of degree k. Combining across all $k \geq 0$, we have a compatible basis of homogeneous elements across I_i^n / I_i^{n+1} for all $i \geq 0$, which is our desired result.

Theorem 5.0.1. *There exists a basis of homogeneous elements for I_i^n for $i \geq 0$, $n \geq 1$ that is compatible with the homomorphisms $i^* : I_j^n \to I_j^n$ for all $i < j$ and inclusions $I_i^n \supseteq I_i^m$ for all $n < m$. This basis arrises form a homogeneous basis on I_i^n / I_i^{n+1} and each element lies in $I_I^n \setminus I_i^{n+1}$ for some n.*

The homogeneous basis elements lie in I_i^n for some n, i. Since they arise as homogeneous basis elements of I_i^n / I_i^{n+1}, these elements have the form

$$\gamma = \sum_{j=1}^{m} \smile_{l_j=1}^{n} \alpha_{l_j}$$

where α_{l_j} are cohomology classes. Recall that $\smile_{l=1}^{n} \alpha_l$ is shorthand for cup products of length n. Note that γ has the property that each cup product in the expression of γ has length n. We call γ a *cup product expression* to make a distinction between simple cup products of the form $\smile_{l=1}^{n} \alpha_l$.

Given n, the interval (p, q) attained from the decomposition of D_k^* corresponds to birth and death events of cup product expressions in D_k^*. As with persistent cohomology, since i^*, goes against the filtration, birth and death are odd concepts. Let us take an element $\psi = \sum_{i=1}^{m} \smile_{i_j=1}^{n} \phi_{i_j}$ of

$$\left(\bigoplus_{i>0} C^i(K; \mathbb{F}) \right)^n / \left(\bigoplus_{i>0} C^i(K; \mathbb{F}) \right)^{n+1} \cap C^k(K; \mathbb{F})$$

which is a cup product expression, but need not be a cocycle in $C^k(K; \mathbb{F})$, where $K = \cup_{K_i \in F} K_i$. We track ψ as we apply i^*. Note that time for us follows the filtration. The event of "death" for ψ occurs when ψ becomes a cocycle. Note that Lemma 4.0.2 implies that this happens some time before all the cochains ϕ_{i_j} become cocycles. The "birth" event, occurs when ψ is a coboundary, which occurs some time after there is a ϕ_{i_j} in each summand that is a coboundary. Thus, the persistence interval (p, q) corresponds to some cup product expression ψ above for which ϕ_{i_j} are all cocycles before time q and an every summand has a ϕ_{i_j} that is a coboundary before time p.

Computing these cup product expressions in not ideal as they are summations of already complex structures. What we would like is to find a representative of γ of the of the form $\smile_{l=1}^{n} \psi_l$ where ψ_l are non-trivial cocycles. Unfortunately, in the case of persistence this may not be possible. We are able to find there sort of representatives when computing the cup product structure on a simplicial complex without persistence.

6 Algorithms

In this section we present several algorithms for computing cup product structure on simplicial complexes. Our first algorithm introduces the general

technique used for a simplicial complex without computing the intervals for persistence. We do this so the persistence algorithm is simpler to follow. Both algorithms take a sparse filtration of a finite simplicial complex ordered by dimensions. We also discuss a semi-persistense algorithm that omits intervals of short length to improve performance.

6.1 Basic Algorithm of a Simplicial Complex

We start with a sparse filtration F of a simplex K. Given a simplex K we can simply take F to be sparse and ordered by dimension, though the also condition is unnecessary. We will compute the cup product structure of K using Algorithm 2. The subroutine $PersistentCohomology()$ simply runs the persistence algorithm and returns the cohomology basis representatives while setting the $\sigma_i.cocycle$ and $\sigma_i.pivot$ cochains in the filtration. Recall that these values allow us to get a basis for cohomology and for the coboundaries.

In the algorithm below, V will be a set of ordered cup products. These are cup product $\smile_i \alpha_i$ for which $i_{\alpha_i} \leq i_{\alpha_j}$ for $i \leq j$. Also, every element $v \in V$ will have members $v.cocycle$ and $v.coboundary$ which are cochains that we will modify, they are both initially zero. When writing $v(\sigma)$ we mean the values of the cup product on σ. The notation $V \cup \alpha$ means $\{\beta \cup \alpha \mid \beta \in V\}$.

The algorithm may seem a little daunting, but the main goal is to express each viable cup product as an element of $Z^k(K;\mathbb{F}) \simeq H^k(K;\mathbb{F}) \oplus B^k(K;\mathbb{F})$. After the algorithm completes, for ever $v \in V$, we have $v = v.cocycle + \delta(v.coboundary)$. The process performed in the second for-loop is simply the process of solving $Ax = v$ where the columns of A are the basis representatives for cohomology and those for the coboundaries with respect to the basis of $\{\sigma_i^*\}$. Each $\sigma_i.cocycle$ or $\sigma_i.pivot$ is of the form $a\sigma_i^* + \sum_{j>i} a_j\sigma_j^*$, so A is a lower-tringular matrix. Our algorithm then constitutes column reduction to express the cup product v in terms of basis elements of $H^k(K;\mathbb{F}) \oplus B^k(K;\mathbb{F})$.

Theorem 4.0.2, which shows the graded-commutative nature of the cup product, implies that I^n is generated by ordered cup products. Thus, there is a basis of ordered cup products for I^n/I^{n+1}. In our last steps of the algorithms, we take all ordered cup products that are not in the trivial cohomology class and find a basis of the space spanned by them. The basis is constructed such that the longest cup products are considered first, therefore guaranteeing that the number of basis elements of length l that we find is the dimension of I^l/I^{l+1}. These elements are also concrete representatives for a basis of I^l/I^{l+1}. Therefore, we have shown the validity of Algorithm 2.

Algorithm 2 $SimpleCupProduct(F, \mathbb{F})$

Let V be empty
$H = PersistentCohomology(F, \mathbb{F})$
Let $V = H$
for $\alpha \in H$ **do**
  $V = V \cup (V \smile \alpha)$
end for
$V = V \setminus H$ { V now contains all ordered simple cup products}
for $\sigma_i \in F$ **do**
  for For all $v \in V$ such that

$$0 \neq b = v(\sigma_i) - v.cocycle(\sigma_i) - v.coboundary(\partial\sigma_i)$$

  do
    if $\sigma_i.pivot$ **then**
      Let $a = \sigma_i.pivot(\partial\sigma_i)$
      Set $v.coboundary += (b/a)\sigma_i.pivot$
    else if $\sigma_i.cocycle$ **then**
      Let $a = \sigma_i.cocycle(\sigma_i)$ { This is the coefficient of σ_i^* in $\sigma_i.cocycle$}
      Set $v.cocycle += (b/a)\sigma_i.cocycle$
    end if
  end for
end for
Let $\overline{V} = \{v \in V \mid v.cocycle \neq 0\}$.
Order $\overline{V}$ such that the longest products are first.
Let B be a basis for the span of $\overline{V}$ found by iterating through $\overline{V}$ in order.
For each l, the set of elements of length l in B is set of basis representatives for I^l/I^{l+1}.

Theorem 6.1.1. *Algorithm 2 correctly computes the cup product structure on a simplicial complex K, providing us with a basis for $I^l/^{l+1}$ for all $l \geq 0$.*

The algorithm may look overly complex, however many of the steps are essential. With our approach it is necessary to find which cup products are non trivial, this requires identifying a cohomology class to which the cup product belongs, which is done in the main for-loop. The last portion of the algorithm in concerned with the issue of $\alpha \smile \beta$ and $\alpha' \smile \beta' \smile \gamma' + \alpha'' \smile \beta'' \smile \gamma''$ being in the same cohomology class, which implies that $\alpha \smile \beta \in I^3$, not $I^2 \setminus I^3$. Since we were unable to find any indication that such cases did not occur, we take them into account in our algorithm.

The algorithm is unfortunately exponential in the number of simplices n provided because we construct all the ordered cup products. There are at most 2^n ordered cup products since we can over count the basis for cohomology to have n elements. In practice however, the number of cohomology classes is much smaller than the number of simplices. We can also introduce techniques to eliminate cup products as part of the algorithm. For example if the cocycle α lies in the same class as a cup product, then any cup product that as α as a constituent can be dropped from consideration. Other improvements of a similar fashion can be made.

6.2 Discussion on Persistence Algorithms

In this section we discuss the possibly of a general persistent cup product algorithm. As we saw in the previous section, we are interested in decomposing the modules

$$D_k^* = \{(I_i^n / I_i^{n+1}) \cap H^k(K_i; \mathbb{F})\}_{i \geq 0}$$

If we know the generators of the vector spaces in this module, we should be able to compute a compatible basis for the module. In this section we will assume the filtration F is ordered by dimension.

We can find a generating set of each I_i^n by modifying Algorithm 1. We will modify Algorithm 1 to compute the representation of each cup product as part of the computation of persistent cohomology. As we discussed, a cup product is born when it becomes a non trivial cocycle. A simple cup product $\smile_{j=1}^m \alpha_j$ when α_j is no longer a cocycle for some j.

For each cup product, we will keep a representation as $\alpha + \delta\beta$ in $Z^k(K; \mathbb{F}) \simeq H^k(K; \mathbb{F}) \oplus B^k(K; \mathbb{F})$. When we talk about updating the cup product representation over a simplex σ_i, we mean to either update α with $\sigma_i.cocycle$

or β with $\sigma_i.pivot$ as we did in Algorithm 2. When we correct for $\beta \mapsto \beta - (v_\beta/v_\alpha)\alpha$, we assume that the representations are updated as well.

Algorithm 3 *CupProductGenerators*$(F_K, \mathbb{F})$

```
Let lists H, R, V be initially empty
for σ_i ∈ F_K do
  Let α ∈ H be the cocycle of greatest index with v_α = (δα)(σ_i) ≠ 0.
  if α exists then
    Remove α from H
    For all β ∈ H v_β = (δβ)(σ_i) ≠ 0 replace β ↦ β − (v_β/v_α)α
    For cup products v ∈ V that contain β, correct for β ↦ β − (v_β/v_α)α

    Mark all v ∈ V containing α as dead at time i
    Update the coboundary part of the representation of all v ∈ V
    Add α to R
  else
    Add σ_i^* to H
    Update the cocycle part of the representation of all v ∈ V, if the
    cocycle part was updated from 0 to aσ_i*, mark the cup product as
    being born at time i
    Add new ordered cup product to V using σ_i^*
  end if
end for
return V
```

The validity of the above algorithm follows from the fact that a cup product of cocycles is always a cocycle. We consider a cup product dead when we can no longer make a substitution $\beta \mapsto \beta - (v_\beta/v_\alpha)\alpha$. Note that since F is ordered by dimension, any new cup products introduces in the last step of the for-loop are trivial until we start adding simplices of the appropriate dimension.

Given these representations of cup products across the filtration, we now have a tangible approach at decomposing D_k^*. Taking the cup products alive at time i, we can find a generators for I_i^n/I_i^{n+1}. Using these generators, it should then be possible to find a compatible basis across D_k^*. Since the constituents of each cup product at time i are compatible basis elements for cohomology, we can do this using a reduction algorithm.

The complexity of this approach is rather unsatisfactory. This algorithm

is much more costly than Algorithm 2 as cohomology classes are born and die rather quickly in complex data sets. This creates large numbers of extraneous cup products that are born and die very quickly, which would most likely be at the same. To mitigate this issue we can consider what we call a semi-persistent algorithm.

Semi-Persistence Algorithm. Given a filtration F and a set of check-in times T, we would like to know the cup product structure at each time $t \in T$. We can construct such an algorithm by modifying Algorithm 2. We may run Algorithm 2 until a time $t \in T$, then find the cup product structure. At the next check-in time $t' \in T$, we update V to include the necessary cup products and by modifying the old once by correcting for the changes the constituent cocycles. This can be done since each of cohomology basis representative is tied to some σ_i^* by our algorithm. We can update the cup product structure for time t' and continue.

Note that the semi-persistnce algorithm provides us with intermitted cup product structure, but does not give us any sub-decomposition of D_k^* since we do not look for compatible bases.

Discussion Both of the above algorithms are unfortunately complicated and the nature of the cup product does not necessarily lend itself nicely to persistent computation even though the theory gives us a proper decomposition. The semi-persitense algorithm provides us with a much more tangible approach to understanding the transformation of the cup product structure within a filtration.

7 Experiments

In this section we discuss the implemented code and provide a discussion of the computational results.

7.1 Implementations

We have implemented Algorithm 1 and 2 for computing cup product structure on simplicial complex. The implementation works over any finite field $\mathbb{Z}_p$. Implementing over $\mathbb{Z}_p$ allows us to deal with spaces such as the Klein bottle, where the cohomology over $\mathbb{Z}$ is $\mathbb{Z} \times \mathbb{Z}_2$. Parts of Algorithm 3 have also been implemented over $\mathbb{Z}_2$, but this work is incomplete. The code was written in C/C++ using standard libraries. Simplicial complexes are repre-

sented combinatorially as ordered subsets of $\mathbb{Z}^{d+1}$. We have also implemented a subdivision algorithm that we use to construct large simplicial complexes to time our algorithm.

This code is currently standalone, but work is being done to incorporate portions into persistence homology and cohomology software. In their current form, the algorithms are more a demonstration of computation than efficient code.

All tests have been done on a machine running Mac OS X version 10.5.8 on an Intel Core 2 Duo 1.5 GHz processor with 2 GB of DDR2 RAM.

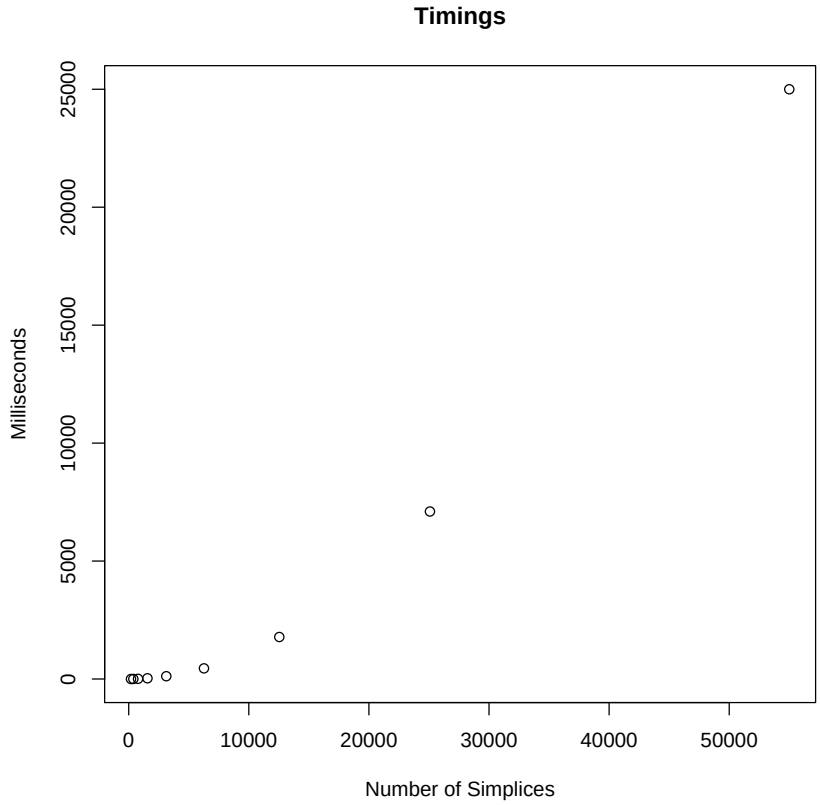
7.2 Data

Due to the nature of our standalone code, we were unable to import large data sets for testing. We have done several tests on the most basic examples available. We ran the torus $T = S^1 \times S^1$ and $M = S^2 \vee S^2 \vee S^1$ as a basic motivating example. For the torus we found two basis elements for I/I^2 and one for I^2/I^3, whereas M had three basis elements for I/I^2, two of dimension 1 and one of dimension 2. This is precisely the desired result.

We also tested more complex 2 dimensional manifolds. We found the correct bases and cup product structure on $T\sharp T, T\sharp T\sharp T$, oriented 2-mainfold of genus 2 and 3 respectively. Projective spaces such as the projective plane and the Klein bottle were also tested, producing proper results. Lens spaces of dimension 3 we also used producing the desired cup product with I/I^2 generated by two elements, one in dimension one and the other in dimension 2, and I^2/I^3 generated by one element in dimension 3. A discussion of the cup product of lens spaces can be found in [7].

One of our constructions of a simplicial complex K by hand had the same cohomology as $T\sharp T$ and the same dimensions for I/I^2 and I^2/I^3. However, of the multiplicative level, only two pairs of the ten ordered cup products of cohomology classes of $H^1(T\sharp T; \mathbb{F})$ are non trivial, while we find four for K that are non trivial. This shows that manifolds are not the only structures to have interesting cup products and that our invariant does not identify the multiplicative structure of the cohomology ring in its entirety.

Using our subdivision algorithm, we were able to get timings for for simplicial complexes of increasing size. We subdivided the 7-simplex representation of the torus. The graph below shows the timings.



Even though our predicted running is exponential, this data can be fitted with a degree 2 polynomial with an $r^2 = 0.998$. However, this may simply be the result of the relatively small cohomology groups of the torus.

8 Conclusions

In this paper we have discussed the theory of persistent cohomology, persistent cup product structure and have provided several algorithms for computing cup product structure. We have been able establish a correspondence that fully describes persistent cup product structure over an arbitrary field. Our algorithm for computing the cup product of a simplicial complex deals directly with the data set computing the cup product using persistent cohomology techniques.

Further research in the subject would be to study persistent cup prod-

uct structure algorithms. In particular, we would be interested in stability results for cup product structure. Also, it would be interesting to answer the question of whether a polynomial time algorithm for computing the cup product of a simplicial complex exits, and if not, are there any conditions on the simplicial complex that allow for a polynomial time algorithm.

Our current implementation of Algorithm 1 and 2 can also be improved by using more robust concurrency libraries and integrating them into other persistence (co)homology frameworks.

9 Acknowledgments

I would like to thank Mikael Vejdemo-Johansson who has helped me through this project this year. He has provided great insight into both the theoretical and computational aspects of the project. I would also like to thank Gunnar Carlsson who first introduced me to this topic and has provided many great articles on the subject. I am also grateful to Dmitriy Morozov for his lectures on persistent homology and computational topology which were very inspiring and influential to my understanding of the subject.

References

[1] Cerri A., Di Fabio B., Ferri M., Frosini P., and Landi C., *Multidimensional persistence homology is stable* (August 2009).

[2] Zomorodian A. and Carlsson G., *Computing persistent homology*, Discrete and Computational Geometry **32** (2005).

[3] de Silva V., *Persistent cohomology*, 2006.

[4] de Silva V., Morozov D., and Vejdemo-Johansson M., *Persistent cohomology and circular coordinates*, Discrete and Computational Geometry (2010).

[5] Carlsson G. and Zomorodian A., *The theory of multidimensional persistence*, Discrete and Computational Geometry **42** (2009).

[6] Carlsson G., Zomorodian A., Collins A., and Guibas L., *Persistence barcodes for shapes*, Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing (2004).

[7] Hatcher, *Algebraic topology*, Cambridge University Press, 2005.