DEPARTMENT OF MATHEMATICS

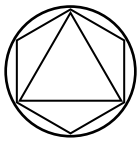
TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters's Thesis in Mathematics

Efficient Topological Data Analysis Using Persistent Relative Homology

Sebastian Kreisel

Supervisor: Prof. Dr. Ulrich Bauer



DEPARTMENT OF MATHEMATICS

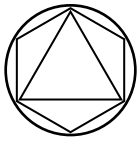
TECHNISCHE UNIVERSITÄT MÜNCHEN

Masters's Thesis in Mathematics

Efficient Topological Data Analysis Using Persistent Relative Homology

Effiziente topologische Datenanalyse mit persistenter relativer Homologie

Author: Sebastian Kreisel
Supervisor: Prof. Dr. Ulrich Bauer
Submission Date: March 01, 2022



I confirm that this masters's thesis in mathematics is my own work and I have documented all sources and material used.

Munich, March 01, 2022 Sebastian Kreisel

Abstract

In recent years, the topological data analysis method persistent homology has found successful application in the field of biology among others. Large data sets surpassing tens and hundreds of thousands of data points, such as the GISAID SARS-CoV-2 virus genome database, remain a challenge even for the highly optimized implementations currently available. This motivates the investigation of alternative methods such as the computation of persistent *relative* homology for which we provide an implementation based on the software Ripser. Moreover, we present and implement an alternative approach to homology representatives based on a dual basis calculation in an effort to reconcile computational efficiency and conceptional ease. Both implementations are evaluated by a detailed performance analysis that also offers general insights into the performance characteristics of Ripser.

Zusammenfassung

In den vergangenen Jahren wurde persistente Homologie, eine Methode der topologische Datenanalyse, erfolgreich angwendet, um zum Beispiel umfangreiche Datensätze in der biologischen Forschung zur analysieren. Die Anzahl an Datenpunkten, wie beispielsweise die des GISAID SARS-CoV-2 Virus Genom Datensatzes, liegt dabei oftmals jenseits von zehn- oder hunderttausend und stellt selbst für kürzlich entwickelte, stark optimierte Implementierungen eine Herausforderung dar. Um dieser gerecht zu werden, wird in der vorliegenden Arbeit persistente *relative* Homologie als Ansatz zur effizienten topologischen Datenanalyse untersucht und basierend auf der Software Ripser implementiert. Zusätzlich wird eine alternative, konzeptionell einfache Methode zur Bestimmung von Homologierepräsentanten auf Grundlage der Berechnung einer Dualbasis präsentiert und implementiert. Beide Implementierungen werden durch eine eingehende Leistungsanalyse evaluiert, welche über die vorgenommenen Anpassungen hinaus auch Einblicke in die Leistungsmerkmale von Ripser selbst bietet.

Contents

1. Introduction

The collection of large amounts of data is common to many fields in modern science. Manual measurements and documentation have largely been replaced by digital data acquisition and automated aggregation, resulting in data sets that often contain thousands or millions of data points. This fundamental change is particularly palpable in biological research. DNA sequencing has revolutionized the field of biology and is nowadays widely available, resulting in a vast amount of raw genome data [35]. The COVID-19 pandemic is an example for the collection of genome data at massive scale. At the time of writing, millions of SARS-CoV-2 virus genomes have been sequenced [23] and made available by GISAID [36], a global initiative established originally in 2008 to aggregate influenza virus genome data. Various institutions and research groups (see [23] for a list examples) have analyzed this data in an effort to understand the evolution of the virus throughout the course of the pandemic and develop countermeasures to its spread and the coronavirus disease. In order to draw insights from such enormous data sets, various data analysis methods have been developed that are themselves a subject of research.
Topological data analysis is one such method, comprising a family of data analysis techniques rooted in algebraic topology. It has been successfully applied to various biological applications in recent years [8, 9, 21], including applications associated to the COVID-19 pandemic [6, 32]. Generally, topological data analysis is suited to gain insights into the global topological structure, the general shape of data that has a geometric nature, at least admitting a notion of distance between data points. More precisely, from the set of pairwise distances within a data set, a combinatorial structure called a simplicial complex is constructed. It encodes the distance relations in the data as a collection of simplices that can be thought of as points, edges (line segments), triangles, tetrahedra and higher-dimensional analogues, glued together along boundaries. As the base input for algebraic tools, simplicial complexes are then characterized by topological features that allow us to draw conclusions about the original data set. The features considered in this thesis are summarized by the term homology, a sequence of vector spaces that describe the number of connected components, holes, voids and higher-dimensional analogues by providing one basis element as a representative for each distinct such feature. Figure 1.1 visualizes eight simplicial complexes in the top-most eight frames based on the same data set but subject to a differing scale parameter. Homology in degree zero (connected components) and one (holes) characterizes them as follows.

scale	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
number of connected components.	45	2	2	1	1	1	1	1
number of holes	0	1	2	3	6	8	2	0

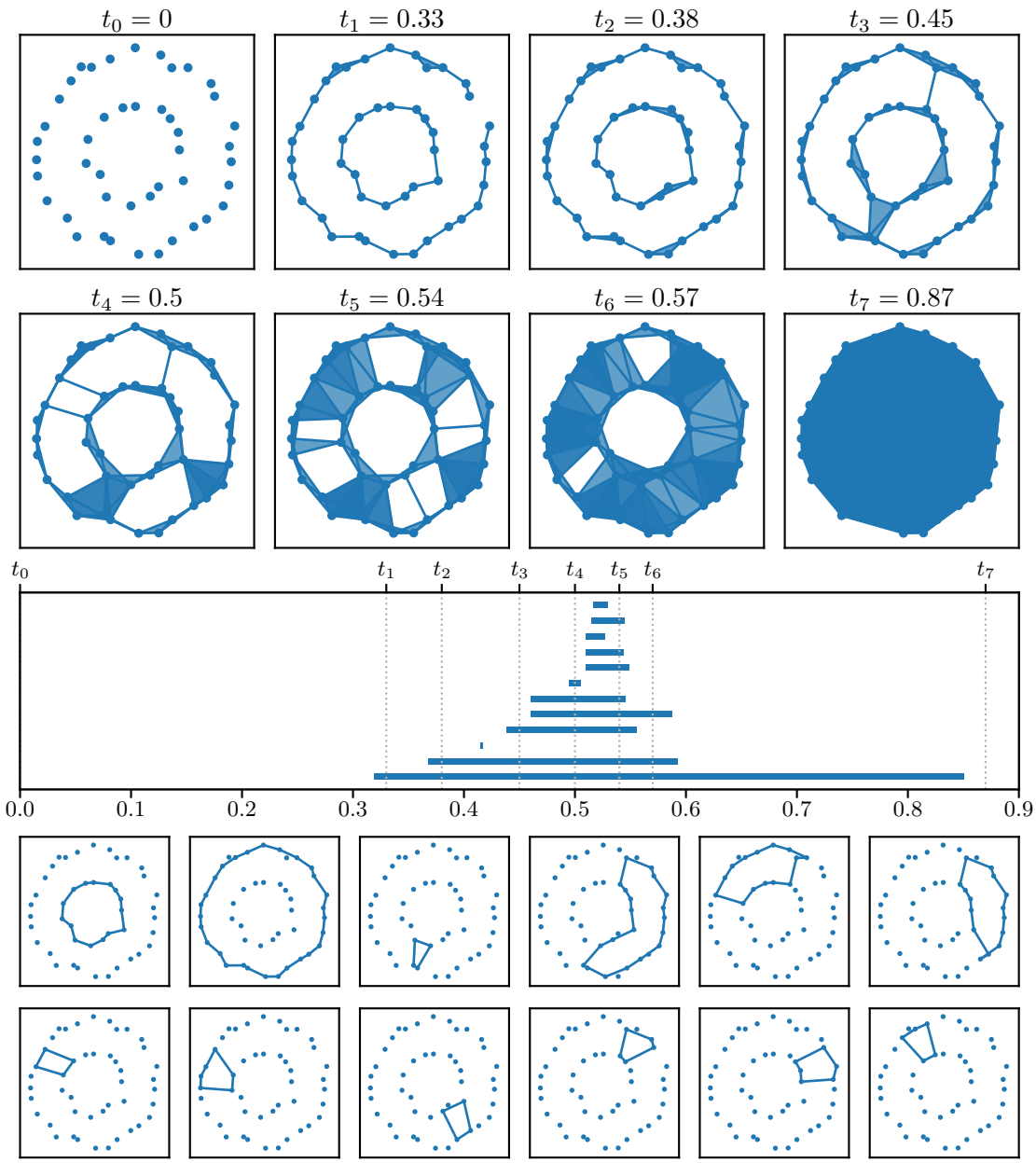


Figure 1.1.: Persistent homology applied to a data set of 45 points in $\mathbb{R}^2$ that is a noisy sample of two circles of radius 0.5 and 1 centered at the origin. The top-most frames show eight parametrized simplicial complexes (the 2-skeleton of Vietoris-Rips complexes) on this set of points at the indicated scale. As the scale increases, line segments and triangles of increasingly large diameter are added, progressively filling in holes. Persistent homology provides a quantitative description of this process: The twelve cycles in the bottom-most frames each represent one distinct occurring hole. Assigned to each such cycle is one bar in the persistence barcode, shown in the middle frame, that specify the persistence (or "lifetime") of its cycle and the hole it represents.

Since all complexes are derived from the same data set, this varying characterization raises the question which one most faithfully reflects the structure in our data set. Since we know that the data was sampled from two circles the answer is the complex at scale t_2 that is characterized by two connected components and two holes (one for each circle). Often however, the underlying structure is unknown, complex and much less explicit. Moreover, we might not find any one single complex that captures all relevant topological features jointly.

Persistent homology is a method in topological data analysis that approaches this question quantitatively. Instead of searching for an optimal simplicial complex, a filtration of (usually parametrized) simplicial complexes is considered. All complexes in this filtration are based on the same underlying data set and include all prior complexes as subcomplexes. Throughout this filtration, topological features appear and vanish as simplices are added progressively. Persistent homology identifies arising features uniquely by assigning it a representative as soon as they appear and tracks how long they persist before vanishing again, if at all. A long interval of persistence indicates a prominent feature that is likely to reflect meaningful global structure in the data set. Short intervals on the other hand correspond to local features that often are the result of noise or artifacts without significance. To summarize, persistent homology identifies topological structure in a data set by considering a filtration of simplicial complexes and determining prominent features by quantifying their persistence throughout the filtration.

Returning to Figure 1.1, the visualized simplicial complexes are eight excerpts of a continuous sequence, called a filtration, parameterized by a scale parameter $t \in \mathbb{R}$ controlling the maximum allowed diameter of simplices. Shown in the middle frame is the persistence barcode in degree one that is obtained by applying persistent homology to the entire filtration (as opposed to only the eight complexes shown). Each of the twelve bars specifies the persistence interval of one distinct homological feature in dimension one, meaning one distinct hole. Each such hole is represented by one of the cycles shown below the barcode plot (bars from bottom to top correspond to cycles from left to right and top to bottom). Note that the first two shown cycles accurately depict the structure in our data set and that the corresponding persistence intervals are indeed the longest among the twelve, whereas all other occurring holes have much shorter associated intervals. This matches our expectation of precisely two predominant features.

This example illustrates how persistent homology is used in practice. Often the persistence barcode alone is used to characterize data sets (see for instance [8, 21, 22, 32]) since it is robust with respect to small perturbations of the data set and the specifics of the machinery required to compute it [3, 11, 13]. The representatives do not satisfy such properties. While the homology classes they represent are topological invariants, the representatives themselves are not uniquely determined and subject to various factors including seemingly arbitrary choices such as the order in which the data points are presented. For certain applications [6, 18, 34], homology representatives are nonetheless of relevance and in part motivate this thesis.

In the following chapter, we provide a relatively self-contained exposition of persistent homology starting with a brief outline of simplicial homology and its variants relative homology and cohomology, as well as filtrations and persistence modules. We then introduce persistent homology, following and expanding upon [2] and [17], with the goal of drawing a clear line between the underlying theoretic framework and algorithmic considerations that are the topic of the next chapter. In the last two sections of Chapter 2, we lift the aforementioned variants of homology into the setting of persistence and reiterate the duality results of [17]. Many key concepts throughout this chapter are illustrated by the means of a recurring example that uses a data set of four points in $\mathbb{R}^2$.
Chapter 3 is concerned with the computational aspects of persistent homology. We first review the required computational setup and the well-known reduction algorithm as well as various optimizations closely following [2]. Motivated by the analysis of the large GISAID SARS-CoV-2 virus genome data set in [6], Section 3.4 investigates persistent relative homology as a tool to tackle data sets that are at the limit or outside the scope of what current implementations can handle within moderate resource constraints. Following a short outline of our implementation based on the software Ripser [2] is a detailed performance analysis and a brief discussion of how the topological features identified by persistent relative homology and persistent (absolute) homology can be related. In the last section of this chapter, we investigate three approaches to the efficient computation of homology representatives with the aim of reconciling conceptual ease with the computational efficiency of the current state-of-the-art method described in [16]. To this end, we present a new variation of the reduction algorithm that derives homology representatives from an optimized cohomology computation facilitated by the duality results of Section 2.9.
The final chapter summarizes our work, discusses a number of topics left untouched and poses open questions.

Part of this thesis project are a number of modified versions of Ripser, implementing a reference homology reduction, several versions that reduce the heavily optimized state-of-the-art implementation to single optimizations each, support for relative (co)homology, additional functionality concerning representatives as well as tooling for parameter specification, improved output generation and the collection of various performance metrics. Besides providing an implementation for the methods of Section 3.4 and 3.5, these versions may be used for performance analysis, for their enhanced functionality with respect to representatives, to visualize barcodes and representatives based on the generated output or for educational purposes. All source code accompanied by configuration files, scripts and data sets is freely available as a fork of Ripser and can be found at `https://github.com/skreisel-tum/ripser/tree/thesis-project`.

2. Persistent Homology

This chapter provides the theoretic framework for the topological data analysis method persistent homology. Broadly speaking, persistent homology extends the notion of simplicial homology to sequences of subcomplexes called filtrations. It tracks how the homology changes throughout the filtration. We develop persistent homology by the means of persistence modules and their barcode decomposition. From this decomposition, the barcode and many other important topological descriptors (such as Betti numbers and homology representatives) derive. The computational aspects of persistent homology are the topic of Chapter 3.

2.1. Simplicial Complexes

We start with a brief introduction to abstract simplicial complexes following [28, §3]. For the purposes of this text it suffices to restrict to finite simplicial complexes of this purely combinatorial nature. We omit the qualifier 'abstract' in the following.

Definition 2.1. Let X be a finite set.

- A *simplex* on X is a non-empty subset $\sigma \subseteq X$.
- The *dimension* of a simplex $\sigma \subseteq X$ is defined to be $\dim(\sigma) = |\sigma| - 1$. We refer to a simplex of dimension k as a k-simplex.
- Given two simplices σ and τ on X we say σ is a *face* of τ and τ is a *coface* of σ if $\sigma \subseteq \tau$. If $\dim(\sigma) + 1 = \dim(\tau)$ then σ is called a *facet* of τ and τ a *cofacet* of σ.
- A set K of simplices on X is called a *simplicial complex* if it is closed with respect to the subset relation. The *dimension* $\dim(K)$ of K is defined to be the maximum dimension among simplices in K.
- A 0-simplex $\{v\} \in K$ is also called a *vertex* of K and is commonly identified with its sole element $v \in X$. We write $\mathrm{vert}(K)$ for the set of all vertices in K.
- A *subcomplex* of a simplicial complex K on X is a subset $F \subseteq K$ that is itself a simplicial complex on X.
- Let K and L be simplicial complexes on sets X and Y respectively. A map of sets $f : K \to L$ is called *simplicial* if it respects the subset relation, that is $\sigma \subseteq \tau$ implies $f(\sigma) \subseteq f(\tau)$ for any simplices $\sigma \subseteq \tau$ in K.
- Simplicial complexes and simplicial maps form a category denoted by **Simp**.

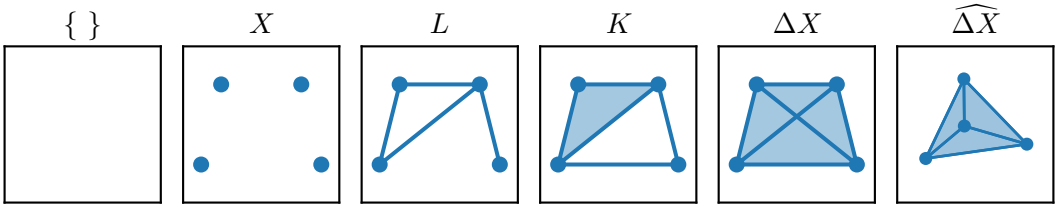


Figure 2.1.: The leftmost five frames visualize five simplicial complexes on the set $X = \{(\pm 2, 2), (\pm 3, -2)\} \subseteq \mathbb{R}^2$. Vertices are depicted as points, 1-simplices as line segments and higher-dimensional simplices as filled polygons. The last frame shows a geometric realization of ΔX in $\mathbb{R}^3$. It is constructed by arbitrarily identifying X with $\{0, e_1, e_2, e_3\}$, the standard basis of $\mathbb{R}^3$ extended by 0.

With the following two definitions we introduce two types of simplicial complexes that are relevant at later points in this thesis.

Definition 2.2. Let X be a finite set. The set $\Delta X = \{\sigma \subseteq X \mid \sigma \neq \emptyset\}$ of all non-empty subsets of X is a simplicial complex called the *full simplex* on X.

Definition 2.3. Let (X, d) be a finite metric space. The *diameter* of a subset $S \subseteq X$ is defined to be $\operatorname{diam}(S) = \max_{x,y \in S} d(x, y)$. For $t \in \mathbb{R}$ the simplicial complex

$$\operatorname{Rips}_t(X) \;=\; \big\{S \subseteq X \mid \operatorname{diam}(S) \leq t,\; S \neq \emptyset\big\}$$

is called the *Vietoris-Rips complex* at scale t.

Note that $\operatorname{Rips}_t(X)$ is the full simplex on X for $t \geq \operatorname{diam}(X)$ whereas for negative t it is empty. We conclude this section with an example of simplicial complexes on a four point set in $\mathbb{R}^2$ to which we return throughout this thesis.

Example 2.4. Let $X = \{(\pm 2, 2), (\pm 3, -2)\} \subseteq \mathbb{R}^2$. Figure 2.1 visualizes the five simplicial complexes $\{\,\}$, X, L, K and ΔX on the set X. The empty complex, X itself and the full simplex ΔX are Vietoris-Rips complexes on (X, d) at respective scales $t < 0$, $t = 0$ and $t \geq \operatorname{diam}(X) = \sqrt{41}$ where d is the Euclidean distance. L and K on the other hand cannot be written as a Vietoris-Rips complex on X. Figure 2.3 shows all possible Vietoris-Rips complexes on X. Note that $\{\,\} \subseteq X \subseteq L \subseteq K \subseteq \Delta X$ are subcomplexes.

Remark 2.5. The notion of a simplicial complex is often used with additional geometric structure: A *geometric k-simplex* is the convex hull of $k + 1$ affinely independent points in an ambient space, usually $\mathbb{R}^d$. The points defining a geometric simplex are called vertices and the convex hull of a subset of vertices is called a face of that simplex. A set of simplices that contains all faces and for which the intersection of any two simplices is a face of both if it is non-empty is called a *geometric simplicial complex*. The union of all simplices of a geometric simplicial complex is a topological space.
Geometric and abstract simplicial complexes are closely related: Reducing all simplices

of a geometric simplicial complex to their respective vertex sets yields an abstract simplicial complex called its *vertex scheme*. Conversely, from any abstract simplicial complex we can construct a geometric simplicial complex called a *geometric realization* by identifying vertices with affinely independent points in $\mathbb{R}^d$ for a sufficiently high dimension d. Figure 2.1 shows a geometric realization of the full simplex on four points. As a topological space it is isomorphic to the topological 3-simplex.

2.2. Simplicial Homology

Simplicial homology is an algebraic descriptor of simplicial complexes that is defined by means of a $\mathbb{Z}$-indexed sequence of vector spaces called a chain complex. In order to define the chain complex on a simplicial complex, in the following denoted by K, it is necessary to place an orientation on its simplices.

Definition 2.6. Let $\sigma = \{v_0, \dots, v_p\} \in K$ be a simplex. By placing a total order $v_{i_0} < \cdots < v_{i_p}$ on the vertices of σ we obtain an *ordered simplex* as the p-tuple $(v_{i_0}, \dots, v_{i_p})$. Identifying two ordered simplices whenever they have identical vertices and their respective orders differ by an even permutation yields an equivalence relation on the set of all ordered simplices of K. An instance of the resulting equivalence classes is called an *oriented simplex*.
We write $[v_0, \dots, v_p]$ for the oriented simplex with vertices $v_0, \dots, v_p$ that lies in the equivalence class of $(v_0, \dots, v_p)$. If σ denotes an oriented simplex we write $-\sigma$ for the oriented simplex of opposite orientation.

Definition 2.7. Let $\mathbb{F}$ be a field and $p \in \mathbb{Z}$. A formal sum of oriented p-simplices

$$\sum_{\substack{\sigma \in K \\ \dim(\sigma)=p}} \lambda_\sigma \, \sigma \qquad (\lambda_\sigma \in \mathbb{F})$$

under the identification $\lambda_\sigma(-\sigma) = (-\lambda_\sigma)\sigma$ is called a *(simplicial) p-chain* on K. The index p is called the *degree* of the chain and the set of all p-chains is denoted by $C_p(K)$.

Remark 2.8. In [28, Section 5] p-chains are defined as a function c from the set of oriented p-simplices in K to $\mathbb{F}$ such that $c(-\sigma) = -c(\sigma)$ and $c(\sigma) = 0$ for all but finitely many oriented simplices σ. This is equivalent to Definition 2.7, the value $c(\sigma)$ corresponding to the coefficient λ_σ.

Pointwise addition defines an abelian group operation on p-chains. Choosing an orientation for all p-simplices results in a set of oriented p-simplices that generates $C_p(K)$ as a free abelian group [28, Lemma 5.1]. If no p-simplex exists $C_p(K)$ contains only the empty sum and is regarded as the trivial group. Since we use field coefficients $C_p(K)$ is in fact a finite-dimensional vector space.

Definition 2.9. Fix a total order on $\mathrm{vert}(K)$. With respect to this order each simplex in K is assigned a unique ordered simplex. Choosing the orientation (equivalence class) of this particular ordered simplex (representative) yields a unique oriented simplex for each simplex. Together they form a basis $\mathcal{C}_p(K)$ of $C_p(K)$ called the *canonical basis* of $C_p(K)$ with respect to the prescribed vertex order.

Any linear map $f : C_p(K) \to V$ may be specified by choosing an image $f(\sigma) \in V$ for all oriented p-simplices σ of K subject to the condition $f(-\sigma) = -f(\sigma)$ [28, Corollary 5.2]. This characterization is not dependent on any choice of basis and we use it in the following definition to introduce the boundary map.

Definition 2.10. The map

$$\partial_p : \quad [v_0, \ldots, v_p] \;\mapsto\; \sum_{k=0}^{p} (-1)^k [v_0, \ldots, v_{k-1}, v_{k+1}, \ldots, v_p]$$

sends an oriented p-simplex to the $(p-1)$-chain of its oriented facets with alternating sign. It respects the orientation, that is $\partial_p(-\sigma) = -\partial_p(\sigma)$ [28, pp. 28–29]) and therefore extends linearly to a linear map $\partial_p : C_p(K) \to C_{p-1}(K)$ called *the boundary map* in degree p.

The sequence of p-chains $C_p(K)$ together with boundary maps ∂_p have the structure of an algebraic object called a chain complex that we now define for the category of finite-dimensional vector spaces $\mathbf{Vect}_{\mathbb{F}}$.

Definition 2.11. For all $p \in \mathbb{Z}$ let C_p be a vector space and $\partial_p : C_p \to C_{p-1}$ a linear map such that

$$\partial_{p-1} \circ \partial_p = 0.$$

The sequence $(C_p, \partial_p)_{p\in\mathbb{Z}}$ is called a *chain complex (of vector spaces)* and ∂_p a *differential*. Let $(C_p, \partial_p)_{p\in\mathbb{Z}}$ and $(D_p, \delta_p)_{p\in\mathbb{Z}}$ be two chain complexes. Then a sequence of maps $(f_p : C_p \to D_p)_{p\in\mathbb{Z}}$ is called *chain map* if it commutes with the differentials, that is

$$f_{p-1} \circ \partial_p \;=\; \delta_p \circ f_p \qquad \text{for all } p \in \mathbb{Z}.$$

Chain complexes of vector spaces and chain maps form a category denoted by $\mathbf{Ch}(\mathbf{Vect}_{\mathbb{F}})$. We also write $(C_\bullet, \partial_\bullet)$ and $f_\bullet$ for an object and a morphism in $\mathbf{Ch}(\mathbf{Vect}_{\mathbb{F}})$ respectively.

Before we continue with the homology of a chain complexes we show that the chain groups $C_p(K)$ together with the boundary map ∂_p of Definition 2.10 are an instance of a chain complex.

Proposition 2.12. The sequence $(C_p(K), \partial_p)_{p\in\mathbb{Z}}$ is a chain complex and the assignment

$$C_\bullet : \mathbf{Simp} \to \mathbf{Ch}(\mathbf{Vect}_{\mathbb{F}}), \qquad \left\{ \begin{array}{ccc} K & \mapsto & C_\bullet(K) \\ f & \mapsto & f_\bullet \end{array} \right\},$$

is a functor that maps a simplicial map $f : K \to L$ to the chain map $f_\bullet : C_\bullet(K) \to C_\bullet(L)$ defined degreewise on the set of oriented simplices by $f_p([v_0, \dots, v_p]) = [f(v_0), \dots, f(v_p)]$ whenever $f(v_0), \dots, f(v_p)$ are distinct and $f_p([v_0, \dots, v_p]) = 0$ otherwise.

Proof. A straightforward calculation suffices to see that $\partial_{p-1} \circ \partial_p = 0$ for all $p \in \mathbb{Z}$ showing that $(C_\bullet(K), \partial_\bullet)$ is a chain complex (see [28, Lemma 5.3]).
It is clear from the construction that f_p respects orientations and thus extends to a linear map $C_p(K) \to C_p(L)$. Verifying that $C_\bullet$ satisfies the properties of a functor is straightforward and we omit it. □

Definition 2.13. Let $(C_\bullet, \partial_\bullet)$ be a chain complex. For $p \in \mathbb{Z}$ we define

$$Z_p \;=\; \ker\big(\partial_p\big), \qquad B_p \;=\; \operatorname{im}\big(\partial_{p+1}\big)$$

called the *group of cycles* and *boundaries* in degree p respectively. Since $\partial_p \circ \partial_{p+1} = 0$ and thus $\partial_p(\operatorname{im}(\partial_{p+1})) = 0$ we have an inclusion $B_p \hookrightarrow Z_p$ of subspaces. We define the *p-th homology group* to be

$$H_p \;=\; \operatorname{coker}(B_p \hookrightarrow Z_p) \;=\; Z_p / B_p.$$

An equivalence class $[z] \in H_p$ is called a *homology class* of $z \in Z_p$ and z is called a *homology representative* of $[z]$.

Simplicial homology refers the homology of $C_\bullet(K)$. Note that the above definitions may be made more generally, replacing $\mathbf{Vect}_\mathbb{F}$ for any abelian category. Common instances are the categories abelian groups and R-modules. For that reason the commonly used terminology is geared towards groups instead of vector spaces. We adopt this terminology, keeping in mind that in our setting all abelian groups are in particular finite-dimensional vector spaces.
The following proposition relates the various groups of Definition 2.13 by assembling them into two short exact sequences that split in $\mathbf{Vect}_\mathbb{F}$.

Proposition 2.14. The short exact sequences

$$0 \longrightarrow Z_p \hookrightarrow C_p \xrightarrow{\partial_p} B_{p-1} \longrightarrow 0$$

$$0 \longrightarrow B_p \hookrightarrow Z_p \twoheadrightarrow H_p \longrightarrow 0$$

split for all $p \in \mathbb{Z}$ and thus $C_p \cong Z_p \oplus B_{p-1}$ as well as $Z_p \cong B_p \oplus H_p$.

Proof. Since $\partial_p(z) = 0$ is satisfied for any p-cycle $z \in Z$, it follows that the former sequence is short exact. The latter is short exact by definition of the quotient $H_p = Z_p / B_p$. In $\mathbf{Vect}_\mathbb{F}$ all short exact sequences split as a consequence of the rank-nullity theorem [25, Theorem 3.2]. □

Simplicial homology in degree zero has a particularly simple structure: The set of homology classes is in bijection with the *connected components* of K that are defined as the connected components of the undirected graph that uses $\mathrm{vert}(K)$ as the set of nodes and the set of 1-simplices as edges.

Proposition 2.15. Let $m \geq 1$ be the number of connected components of K and for $i \in \{1, \ldots, m\}$ let $A_i \subseteq \mathrm{vert}(K)$ denote the set of vertices of the i-th connected component. Then we obtain a basis $\{[v_i] \,|\, v_i \in A_i,\ 1 \leq i \leq m\}$ of $H_0(C_\bullet(K))$ by arbitrarily choosing a vertex $v_i \in A_i$ for each connected component A_i.

Proof. We reproduce the proof of [28, Theorem 7.1] in our setting. First we show that any $v \in A_i$ satisfies $v \in [v_i]$. Since both v and v_i are in the same connected component A_i, there exists a sequence of oriented 1-simplices

$$[v, w_1], [w_1, w_2], [w_2, w_3], \ldots, [w_{n-1}, w_n], [w_n, v_i]$$

such that as a path it connects v and v_1. Applying the boundary map ∂_1 to the chain c given by the sum of these 1-simplices yields $\partial_1 c = v - v_i$ implying $v \in [v_i]$.
It remains to verify that $\{[v_i] \,|\, v_i \in A_i,\ 1 \leq i \leq m\}$ is linearly independent, that is showing that any non-zero combination $c = \sum_{i=1}^m \lambda_i v_i$ ($\lambda_i \in \mathbb{F}$) cannot be a boundary. To do so we assume the contrary: Let $\partial_1 d = c$ for some 1-chain d. Then we may express $d = \sum_{i=1}^m d_i$ for 1-chains d_i such that $\partial d_i \in A_i$ and by linearity of ∂_1 we have $\partial_1 d_i = \lambda_i v_i$. However, the boundary of any 1-chain cannot consist of a single vertex forcing $\lambda_i = 0$ for all i which contradicts our assumption that $c \neq 0$. □

It is often convenient to accumulate all degrees. To do so we take the direct sum over $\mathbb{Z}$ producing *graded vector spaces*.

Definition 2.16. Let $(C_\bullet, \partial_\bullet)$ be a chain complex. We write

$$Z_\bullet = \bigoplus_{p \in \mathbb{Z}} Z_p, \qquad B_\bullet = \bigoplus_{p \in \mathbb{Z}} B_p, \qquad H_\bullet = \bigoplus_{p \in \mathbb{Z}} H_p\,.$$

We may also view $C_\bullet = \bigoplus_{p \in \mathbb{Z}} C_p$ as a graded vector space and $\partial_\bullet$ as a linear map $C_\bullet \to C_\bullet$.
If $C_\bullet = C_\bullet(K)$ then all graded vector spaces are finite-dimensional since all but finitely many chain groups of a finite simplicial complex are trivial. The union of all canonical bases $\mathcal{C}_p(K)$ of $C_\bullet(K)$ is a basis of $C_\bullet(K)$ denoted by $\mathcal{C}(K)$ and is likewise called the *canonical basis* of $C_\bullet(K)$.

Proposition 2.17. The assignment

$$H_p : \mathbf{Ch}(\mathbf{Vect}_\mathbb{F}) \to \mathbf{Vect}_\mathbb{F}, \qquad \left\{ \begin{array}{lcl} C_\bullet & \mapsto & H_p \\ f_\bullet & \mapsto & H_p(f_\bullet) \end{array} \right\}$$

is a functor that maps a chain map $f_\bullet : C_\bullet \to C'_\bullet$ to the linear map $H_p(f_\bullet) : H_p \to H'_p$ given by $[z] \mapsto [f_p(z)]$.

Proof. The chain map $f_\bullet$ commutes with the differential by definition and we have $f_{p-1}(\partial_p(c)) = \partial'_p(f_p(c)) = 0$ if $\partial_p(c) = 0$ for any p-chain $c \in C_p$. Similarly, applying f_p to $c = \partial_{p+1}(d)$ for any $d \in C_{p+1}$ yields $f_p(c) = f_p(\partial_{p+1}(d)) = \partial'_{d+1}(f_{p+1}(d))$. Thus cycles and boundaries are preserved by f_p and $H_p(f_\bullet)$ is well-defined. It is easy to verify that H_p is a functor by checking that it preserves identities and commutes with the composition (see [28, Theorem 12.2]). □

The composite functor $H_p \circ C_\bullet : \mathbf{Simp} \to \mathbf{Vect}_\mathbb{F}$ is often abbreviated to H_p and we write $H_p(K)$ instead of $H_p(C_\bullet(K))$. In this context $\dim(H_p(K))$ is called the *p-the Betti number*.
Proceeding degreewise yields the following corollary to Proposition 2.17.

Corollary 2.18. The assignment $H_\bullet : \mathbf{Ch}(\mathbf{Vect}_\mathbb{F}) \to \mathbf{Vect}_\mathbb{F}$ mapping $C_\bullet \mapsto H_\bullet$ and $f_\bullet \mapsto H_\bullet(f_\bullet)$ is a functor. □

In the following two sections we continue with two topics closely related to the homology of a simplicial complex: Relative homology and cohomology. Similar to homology both are functors taking a simplicial complex to a (graded) vector space. The former is a quotient construction on the level of chain groups, the latter the vector space dual.

2.3. Relative Homology

Throughout this section let K be a finite simplicial complex and $L \subseteq K$ a subcomplex. Note that the canonical basis of $C_p(K)$ restricts to the canonical basis of $C_p(L)$ for any $p \in \mathbb{Z}$ which implies that the inclusion $L \hookrightarrow K$ is preserved by $C_\bullet : \mathbf{Simp} \to \mathbf{Ch}(\mathbf{Vect}_\mathbb{F})$. Consequently we have an inclusion of chain groups $C_p(L) \hookrightarrow C_p(K)$ as well as a quotient $C_p(K)/C_p(L)$ denoted in the following by $C_p(K, L)$. We may summarize these relations as the short exact sequence

$$0 \longrightarrow C_p(L) \hookrightarrow C_p(K) \twoheadrightarrow C_p(K, L) \longrightarrow 0$$

that is an immediate consequence of the first isomorphism theorem (see also [28, p. 136]). In $\mathbf{Vect}_\mathbb{F}$ the sequence splits, that is $C_p(K) \cong C_p(L) \oplus C_p(K, L)$, allowing us to restrict $\mathcal{C}_p(K)$ to a basis of $C_p(K, L)$.

Definition 2.19. The restriction of $\mathcal{C}_p(K)$ (with respect to a fixed order on $\mathrm{vert}(K)$) to oriented simplices in $K \setminus L$ is a basis of $C_p(K, L)$ called the *canonical basis* of $C_p(K, L)$. It is denoted by $\mathcal{C}_p(K, L)$. We omit the subscript p when referring to the canonical basis of $C_\bullet(K, L)$ (that is the union of all $\mathcal{C}_p(K, L)$ for $p \in \mathbb{Z}$).

The above split-exact sequence implies that the boundary map of $C_\bullet(K)$ restricts to the boundary map of $C_\bullet(L)$ and thereby induces a map of quotients

$$\widetilde{\partial}_p : C_p(K, L) \to C_{p-1}(K, L)$$

that satisfies $\widetilde{\partial}_{p-1} \circ \widetilde{\partial}_p = 0$ [28, pp. 47–48]. The sequence $(C_p(K,L), \widetilde{\partial}_p)_{p\in\mathbb{Z}}$ is thus a chain complex in its own right. Its homology is called *(simplicial) relative homology*.

Definition 2.20. The quotient $C_p(K,L) := C_p(K)/C_p(L)$ is called the group of *relative p-chains* (with respect to K and L). Definition 2.13 applied to the chain complex $(C_\bullet(K,L), \widetilde{\partial}_\bullet)$ yields

$$Z_p(K,L) \;=\; \ker(\widetilde{\partial}_p), \qquad B_p(K,L) \;=\; \operatorname{im}(\widetilde{\partial}_{p+1}),$$

the group of *relative cycles* and *relative boundaries* in degree p respectively. The *p-th relative homology group* is defined to be

$$H_p(K,L) \;=\; \operatorname{coker}\big(B_p(K,L) \hookrightarrow Z_p(K,L)\big) \;=\; Z_p(K,L)/B_p(K,L).$$

To avoid ambiguities we sometimes qualify the homology of $C_\bullet(K)$ as *absolute*. As an instance of Definition 2.13, Proposition 2.17 asserts that the assignment $C_\bullet(K,L) \mapsto H_p(K,L)$ satisfies the properties of a functor $\mathbf{Ch}(\mathbf{Vect}_\mathbb{F}) \to \mathbf{Vect}_\mathbb{F}$ albeit it is not defined for all objects in $\mathbf{Ch}(\mathbf{Vect}_\mathbb{F})$. We call such an assignment *functorial*.

The short exact sequence of chain groups above describes how absolute and relative chain groups are related. This relation is generally not preserved by the homology functor H_p as it is neither left nor right exact. Instead absolute and relative homology are related by the long exact sequence

$$\cdots \longrightarrow H_{p+1}(K,L) \xrightarrow{\delta_{p+1}} H_p(L) \longrightarrow H_p(K) \longrightarrow H_p(K,L) \xrightarrow{\delta_p} H_{p-1}(L) \longrightarrow \cdots$$

where the *connecting homomorphism* δ_p sends a relative homology class $[z] \in H_p(K,L)$ to $[\partial_p(d)] \in H_{p-1}(L)$ for any choice of chain $d \in C_p(K)$ that projects onto $z \in C_p(K,L)$. For more details on the definition of δ_p and the construction of the long exact sequence as an instance of the zig-zag lemma we refer to [28, §24].

We conclude this section with the excision theorem that establishes an isomorphism $H_p(K,L) \cong H_p(M,N)$ for subcomplexes $L \subseteq K \subseteq M \supseteq N$ whenever $K \setminus L = M \setminus N$ or equivalently $L = K \cap N$ and $M = K \cup N$. Figure 2.2 shows and example of four complexes that satisfy this condition.

Theorem 2.21. Let $L \subseteq K$, $N \subseteq M$ and $K \subseteq M$ be subcomplexes such that $K \setminus L = M \setminus N$. Then the inclusion $K \hookrightarrow M$ induces isomorphisms $C_p(K,L) \cong C_p(M,N)$ and $H_p(K,L) \cong H_p(M,N)$ for all $p \in \mathbb{Z}$.

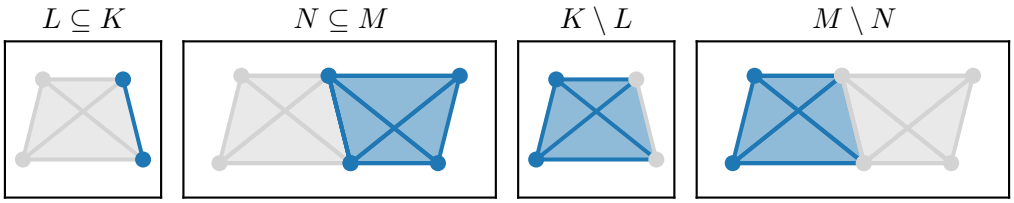


Figure 2.2.: The leftmost two frames show in blue the subcomplexes $L \subseteq K$ and $N \subseteq M$ of $K = \Delta X = \mathrm{Rips}_{\sqrt{41}}(X)$ and $M = \mathrm{Rips}_{\sqrt{41}}(Y)$ respectively where X is the four point space of Example 2.4 and $Y = X \cup \{(8,2),(7,-2)\}$. The rightmost two frames show in blue the identical sets $K \setminus L$ and $M \setminus N$ that are the canonical bases $\mathcal{C}(K,L)$ and $\mathcal{C}(M,N)$ of the relative chain complexes $C_\bullet(K,L)$ and $C_\bullet(M,N)$ respectively. The condition of Theorem 2.21 is satisfied and consequently $H_\bullet(K,L) \cong H_\bullet(M,N)$. Note that this is no longer the case if M is replaced by $\Delta Y \supsetneq \mathrm{Rips}_{\sqrt{41}}(Y)$.

Proof. We reproduce the proof of [28, Theorem 9.1] in our setting. For any $p \in \mathbb{Z}$ we construct a map $\phi_p : C_p(K) \to C_p(M,N)$ by composing the inclusion $C_p(K) \hookrightarrow C_p(M)$ with the projection $C_p(M) \twoheadrightarrow C_p(M,N)$, mapping

$$\sigma \;\mapsto\; \begin{cases} \sigma + C_p(N) & \text{if } \sigma \in K \setminus L = M \setminus N, \\ 0 & \text{if } \sigma \in L = K \cap N. \end{cases}$$

for all $\sigma \in \mathcal{C}_p(K)$. By construction $\ker(\phi_p) = C_p(L)$ and $\mathrm{im}(\phi_p) = C_p(M,N)$ due to $M \setminus N = K \setminus L$. The first isomorphism theorem yields an isomorphism $\widetilde{\phi}_p : C_p(K,L) \to C_p(M,N)$ by mapping $\sigma + C_p(L) \mapsto \sigma + C_p(N)$ for $\sigma \in K \setminus L = M \setminus N$. It is easy to check that $\widetilde{\phi}_p$ commutes with the boundary map and is thus a chain map that the functor H_p takes to an isomorphism $H_p(K,L) \cong H_p(M,N)$. □

2.4. Cohomology

Cohomology is the dual notion to homology and is defined via the contravariant Hom-functor

$$\mathrm{Hom}(-,B) : \mathbf{Vect}_\mathbb{F} \to \mathbf{Vect}_\mathbb{F}, \qquad \left\{ \begin{array}{lcl} A & \mapsto & \mathrm{Hom}(A,B), \\ f & \mapsto & f^* = (\phi \mapsto \phi \circ f) \end{array} \right\},$$

mapping a linear map $f : X \to Y$ to its dual $f^* : \mathrm{Hom}(Y,B) \to \mathrm{Hom}(X,B)$ by pre-composing with f. Note that $\mathrm{Hom}(-,B)$ preserves the zero-morphism and compositions since $0 \mapsto (\phi \mapsto \phi \circ 0) = 0$ and $f \circ g \mapsto g^* \circ f^*$ (see also [28, pp. 245–248]).
As before, we denote by K a finite simplicial complex and restrict to the category of finite-dimensional vector spaces $\mathbf{Vect}_\mathbb{F}$.

Proposition 2.22. For $B = \mathbb{F}$ the functor $\mathrm{Hom}(-, \mathbb{F})$ is also denoted by $(-)^*$ and is an isomorphism of vector spaces, that is

$$A \cong A^* \quad \text{and} \quad A^{**} \cong A$$

for any finite-dimensional vector space A. A^* is the called *dual* and A^{**} (that is the result of $(-)^*$ applied twice) the *double dual* of A. The double dual is a covariant functor $(-)^{**} : \mathbf{Vect}_{\mathbb{F}} \to \mathbf{Vect}_{\mathbb{F}}$ that is naturally isomorphic to the identity functor on $\mathbf{Vect}_{\mathbb{F}}$.

Proof. For $A \cong A^*$ see for instance [25, p. 126] and for the facts about $(-)^{**}$ we refer to [26, pp. 32–33]. □

We proceed in fashion of Section 2.2 and first introduce the construct underlying cohomology, namely cochain complexes. We then define cohomology and exhibit simplicial cohomology as an instance of this general construction.

Definition 2.23. For $p \in \mathbb{Z}$ let C^p be a vector space and $\partial^p : C^p \to C^{p+1}$ a linear map such that

$$\partial_{p+1} \circ \partial_p = 0 \qquad \text{for all } p \in \mathbb{Z}.$$

The sequence (C^p, ∂^p) is called a cochain complex (of vector spaces). Given two cochain complexes $(C^p, \partial^p)_{p\in\mathbb{Z}}$ and $(D^p, \delta^p)_{p\in\mathbb{Z}}$, a sequence of maps $(f^p : C^p \to D^p)_{p\in\mathbb{Z}}$ is called a *cochain map* if

$$f^{p+1} \circ \partial^p = \delta^p \circ f^p.$$

Cochain complexes and cochain maps form the category $\mathbf{CoCh}(\mathbf{Vect}_{\mathbb{F}})$. We write $(C^\bullet, \partial^\bullet)$ and $f^\bullet$ for an object and a morphism in $\mathbf{CoCh}(\mathbf{Vect}_{\mathbb{F}})$ respectively.

Definition 2.24. Let $(C^\bullet, \partial^\bullet)$ be a cochain complex in $\mathbf{Ch}(\mathbf{Vect}_{\mathbb{F}})$. For $p \in \mathbb{Z}$ we define

$$Z^p = \ker(\partial^p), \qquad B^p = \mathrm{im}(\partial^{p-1})$$

to be the group of *cocycles* and the group of *coboundaries* in degree p respectively. The cokernel of the inclusion $B^p \hookrightarrow Z^p$ defines the *p-th cohomology group*

$$H^p = \mathrm{coker}(B^p \hookrightarrow Z^p) = Z^p/B^p.$$

The categories of chain complexes and cochain complexes (indexed over $\mathbb{Z}$) are isomorphic: Reversing the index order by mapping $i \mapsto -i$ bijectively assigns to each chain complex a cochain complex and to each chain map a cochain map. This isomorphism identifies the p-th homology group with the $(-p)$-th cohomology group. The following corollary to Proposition 2.17 is a direct consequence of this isomorphism.

Corollary 2.25. The assignment $C^\bullet \mapsto H^p$ and $f^\bullet \mapsto H^p(f^\bullet)$ is a functor $\mathbf{CoCh}(\mathbf{Vect}_{\mathbb{F}}) \to \mathbf{Vect}_{\mathbb{F}}$ for all $p \in \mathbb{Z}$. □

Proposition 2.26. The assignment

$$\mathrm{Hom}(-,\mathbb{F}) : \mathbf{Ch}(\mathbf{Vect}_{\mathbb{F}}) \to \mathbf{CoCh}(\mathbf{Vect}_{\mathbb{F}}), \qquad \left\{ \begin{array}{lcl} C_\bullet & \mapsto & \mathrm{Hom}(C_\bullet,\mathbb{F}), \\ f_\bullet & \mapsto & f_\bullet^* \end{array} \right\}$$

mapping $C_p \mapsto \mathrm{Hom}(C_p,\mathbb{F})$, $\partial_p \mapsto \partial_p^*$ and $f_p \mapsto f_p^*$ for all $p \in \mathbb{Z}$ is a contravariant functor.

Proof. $\mathrm{Hom}(-,\mathbb{F})$ applied to the equation $\partial_{p-1} \circ \partial_p = 0$ yields $\partial_p^* \circ \partial_{p-1}^* = 0$ for any $p \in \mathbb{Z}$ since $\mathrm{Hom}(-,\mathbb{F})$ preserves compositions and the zero morphism. Thus $\partial^p := \partial_{p+1}^*$ defines a differential $\mathrm{Hom}(C_p,\mathbb{F}) \to \mathrm{Hom}(C_{p-1},\mathbb{F})$ and $(\mathrm{Hom}(C_p,\mathbb{F}),\partial^p)_{p\in\mathbb{Z}}$ is a cochain complex. Let $f_\bullet : (C_\bullet,\partial_\bullet) \to (D_\bullet,\delta_\bullet)$ be a chain map. Then $f_p \circ \partial_{p+1} = \delta_{p+1} \circ f_{p+1}$ is satisfied for all $p \in \mathbb{Z}$. By applying $\mathrm{Hom}(-,\mathbb{F})$ to this equation we obtain $\partial_{p+1}^* \circ f_p^* = f_{p+1}^* \circ \delta_{p+1}^*$ and thus $\partial^p \circ f_p^* = f_{p+1}^* \circ \delta^p$ is satisfied for all $p \in \mathbb{Z}$. The degreewise dual of $f_\bullet$ is thus a cochain map $f^\bullet : \mathrm{Hom}(D_\bullet,\mathbb{F}) \to \mathrm{Hom}(C_\bullet,\mathbb{F})$. Functoriality follows from the degreewise functoriality of $\mathrm{Hom}(-,\mathbb{F})$. □

With the following definition we define the cohomology of a finite simplicial simplex K as the cohomology of $\mathrm{Hom}(C_\bullet(K),\mathbb{F})$.

Definition 2.27. The *simplicial cohomology* of a finite simplicial complex K is the cohomology of the cochain complex

$$\big(C^p(K),\, \partial^p\big)_{p\in\mathbb{Z}} = \big(\mathrm{Hom}(C_p(K),\mathbb{F}),\, \partial_{p+1}^*\big)_{p\in\mathbb{Z}}$$

also denoted by $(C^\bullet(K),\partial^\bullet)$. The differential $\partial^\bullet$ of $C^\bullet(K)$ is also called the *coboundary map*.

Often we simply write H^p for the composite functor $H^p \circ \mathrm{Hom}(-,\mathbb{F}) \circ C_\bullet$ from **Simp** to $\mathbf{Vect}_{\mathbb{F}}$.

Definition 2.28. Let $\mathcal{C}_p(K) = \{\sigma_1,\ldots,\sigma_n\}$ be the canonical basis of $C_p(K)$ with respect to a fixed total order on $\mathrm{vert}(K)$. For $\sigma_i \in \mathcal{C}_p(K)$ we define a linear map

$$\sigma_i^* : \mathcal{C}_p(K) \to \mathbb{F}, \qquad \sigma_i^*(\sigma_j) \;=\; \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

that linearly extends to $C^p(K)$. Together these maps are the dual basis of $\mathcal{C}_p(K)$ denoted by $\mathcal{C}_p^*(K)$ and called the *canonical basis* of $C^p(K)$ (with respect to the prescribed vertex order). We omit the subscript p when referring to the canonical basis of $C^\bullet(K)$ (that is the union of all $\mathcal{C}^p(K)$ for $p \in \mathbb{Z}$).

With the help of Proposition 2.22 we can characterize the coboundary map more concretely: Given a basis element $\sigma^* \in \mathcal{C}_p^*(K)$ the definition of the dual map yields $\partial^p(\sigma^*) = \sigma^* \circ \partial_{p+1}$. Applied to a $(p+1)$-simplex τ the coboundary evaluates to $(\sigma^* \circ \partial_{p+1})(\tau) = \pm 1$ (as opposed to 0) if and only if τ is a cofacet of σ. Neglecting the signs, the coboundary

$\partial^p(\sigma^*)$ is thus the sum of those basis elements $\tau^* : C^{p+1} \to \mathbb{F}$ such that $\tau \supseteq \sigma$ is a cofacet, just as the boundary of a simplex is the sum of its facets. This illustrates the duality of the boundary and coboundary map (see also [28, pp. 252–253]).

Theorem 2.29. Let $C_\bullet$ be a chain complex and $C^\bullet = \mathrm{Hom}(C_\bullet, \mathbb{F})$. Then there exists a natural isomorphism

$$\mathrm{Hom}(H_p(C_\bullet), \mathbb{F}) \;\cong\; H^p(C^\bullet)$$

for any $p \in \mathbb{Z}$. By natural we mean that for any chain map $f_\bullet : C_\bullet \to D_\bullet$ and all $p \in \mathbb{Z}$ the following diagram commutes:

$$\begin{array}{ccc} \mathrm{Hom}(H_p(D_\bullet), \mathbb{F}) & \xrightarrow{\cong} & H^p(D^\bullet) \\ \downarrow{\scriptstyle H_p(f_p)^*} & & \downarrow{\scriptstyle H^p(f_p^*)} \\ \mathrm{Hom}(H_p(C_\bullet), \mathbb{F}) & \xrightarrow{\cong} & H^p(C^\bullet) \end{array}$$

Proof. A proof of this theorem as a consequence of the universal coefficient theorem can be found in [28, Theorem 53.5]. We provide a different proof below that only relies on elementary homological algebra.
First we establish some facts about $\mathrm{Hom}(-, B)$ in $\mathbf{Vect}_\mathbb{F}$. In any abelian category $\mathrm{Hom}(-, \mathbb{F})$ is left-exact [38, Corollary 1.6.9] and therefore maps kernels to cokernels. Due to the fact that any object in $\mathbf{Vect}_\mathbb{F}$ is injective [24, Example 8.4.2] it is also right-exact mapping cokernels to kernels [38, Lemma 2.3.4]. In Definition 2.13 homology is defined via the cokernel $\mathrm{coker}(\mathrm{im}(\partial_{p+1} \hookrightarrow \ker(\partial_p))$. Equivalently we may also express H_p as $\ker(\mathrm{coker}(\partial_{p+1} \twoheadrightarrow \mathrm{im}(\partial_p))$ [24, Definition 8.3.8]. Applying $\mathrm{Hom}(-, \mathbb{F})$ to this alternate expression of H_p yields

$$\ker\big(\mathrm{coker}(\partial_{p+1}) \twoheadrightarrow \mathrm{im}(\partial_p)\big) \quad \xrightarrow{\mathrm{Hom}(-,\mathbb{F})} \quad \mathrm{coker}\big(\mathrm{coim}(\partial^{p-1}) \hookrightarrow \ker(\partial^p)\big)\,.$$

By the natural isomorphism $\mathrm{coim}(\partial^{p-1}) \cong \mathrm{im}(\partial^{p-1})$ [24, Definition 8.3.5] the right-hand side applied to $C^\bullet$ is isomorphic to $H^p(C^\bullet)$ and thus isomorphic the left-hand side under $\mathrm{Hom}(-, \mathbb{F})$ applied to $C_\bullet$, that is $\mathrm{Hom}(H_p(C_\bullet), \mathbb{F})$. Since (co)kernel and (co)image are natural with respect to morphisms, so is this isomorphism. □

By Proposition 2.22 there further exists an isomorphism $\mathrm{Hom}(H_p(C_\bullet), \mathbb{F}) \cong H_p(C_\bullet)$. Together with Theorem 2.29 we thus have an isomorphism of homology and cohomology.

Corollary 2.30. Simplicial homology and cohomology are isomorphic. □

We conclude this section with the definition of simplicial relative cohomology as the dual of relative homology.

Definition 2.31. Let $L \subseteq K$ be a subcomplex. For $p \in \mathbb{Z}$ we define

$$C^p(K, L) \;=\; \mathrm{Hom}(C_p(K, L), \mathbb{F}) \quad \text{and} \quad \widetilde{\partial}^p : C^p(K, L) \to C^{p+1}(K, L), \quad \widetilde{\partial}^p = \widetilde{\partial}^*_{p+1},$$

to be the group of *relative cochains* and the *relative coboundary map* in degree p respectively. Definition 2.24 applied to the cochain complex $(C^\bullet(K,L), \widetilde{\partial}^\bullet)$ yields

$$Z^p(K,L) \;=\; \ker\big(\widetilde{\partial}^p\big), \qquad B^p(K,L) \;=\; \mathrm{im}\big(\widetilde{\partial}^{p-1}\big),$$

the group of *relative cocycles* and the group of *relative coboundaries* (with respect to K and L) in degree p respectively. The cokernel of the inclusion $B^p(K,L) \hookrightarrow Z^p(K,L)$ defines the *p-th relative cohomology group* $H^p(K,L)$.

We can transfer many of the results established for absolute cohomology and relative homology to relative cohomology. For instance, Proposition 2.26 and Theorem 2.29 apply in the relative setting for $C^\bullet = C^\bullet(K,L)$ as summarized in the following corollary.

Corollary 2.32. The assignment $C^p(K,L) \mapsto H^p(K,L)$ is functorial and we have a natural isomorphism $\mathrm{Hom}(H_p(C_\bullet(K,L)), \mathbb{F}) \cong H^p(C^\bullet(K,L))$ for all $p \in \mathbb{Z}$. □

As a further example, the short exact sequence established in the last section relating absolute and relative chain groups is preserved as

$$0 \longleftarrow C^p(L) \twoheadleftarrow C^p(K) \hookleftarrow C^p(K,L) \longleftarrow 0$$

since $\mathrm{Hom}(-,\mathbb{F})$ is exact in $\mathbf{Vect}_\mathbb{F}$. The application of H^p yields an analogous long exact sequence with connecting homomorphisms $\delta^p : H^p(L) \to H^{p+1}(K,L)$ for $p \in \mathbb{Z}$.

2.5. Filtrations

In this section we introduce the notion of a filtration of a finite simplicial complex K. Together with simplicial homology, filtrations sit at the core of persistent homology. Our terminology is largely adapted from [2].

Definition 2.33. Let I be a totally ordered set. A *filtration* of K denoted K_I is a sequence of simplicial subcomplexes $(K_i)_{i\in I}$ such that

$$i \;\leq\; j \quad\Longleftrightarrow\quad K_i \;\subseteq\; K_j$$

and $K_i = K$ for some $i \in I$. A filtration K_I is called *essential* if $K_i \neq K_j$ holds for all indices $i \neq j$. It is called *simplexwise* if for any $j \in I$

$$K_j \;=\; \begin{cases} \{\,\} \;\text{ or} \\ K_i \cup \{\sigma_j\} \text{ for some } \sigma_j \in K \text{ and } i < j, \end{cases}$$

that is simplices are added one by one. A simplexwise filtration induces a total order on K given by

$$\sigma < \tau \quad\Longleftrightarrow\quad \text{there exists } i \in I \text{ such that } \sigma \in K_i \text{ and } \tau \notin K_i.$$

In the following we refer to this order as the *filtration order* with respect to K_I.

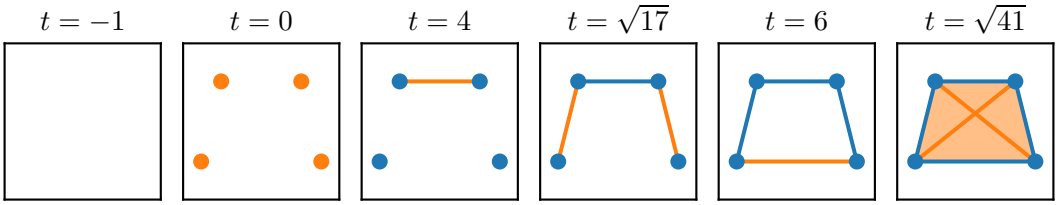


Figure 2.3.: All distinct simplicial complexes that appear in the Vietoris-Rips filtration of the full simplex on the four point space introduced in Example 2.4 at the indicated scale. Simplices added at the respective scale are highlighted in orange. As a filtration these six complexes condense the Vietoris-Rips filtration.

A simplexwise filtration starts with the empty set and adds precisely one simplex at each filtration index such that the resulting set of simplices is again a simplicial complex. Our definition of a filtration K_I requires it to end in K. This comes at no loss of generality since any filtration of K that would end in a subcomplex $L \subseteq K$ may just as well be considered a filtration of L.
Note that an essential filtration of a finite simplicial complex necessarily has a finite index set. Such filtrations can be thought of as a time series where simplices are added in discrete steps to ultimately obtain K. This motivates the following terminology.

Definition 2.34. Let K_I be an essential filtration and σ_i, $\sigma_j \in K$ two simplices entering the filtration at indices i and j respectively. If $i < j$ and thus $\sigma_i < \sigma_j$ in the filtration order we say σ_i is *older than* σ_j and σ_j is *younger than* σ_i.

With the following definition we assemble the Vietoris-Rips complexes $\mathrm{Rips}_t(X)$ into a filtration by allowing the parameter $t \in \mathbb{R}$ to vary.

Definition 2.35. Let (X, d) be a finite metric space. The sequence of Vietoris-Rips complexes $\mathrm{Rips}_t(X)$ for $t \in \mathbb{R}$ is a filtration of the full simplex ΔX with index set $\mathbb{R}$ called the *Vietoris-Rips filtration* of ΔX.

Equivalently, the Vietoris-Rips filtration may be defined as the sublevel set of the diameter function $\mathrm{diam} : K \to \mathbb{R}$.
In later sections we require filtrations to be essential and simplexwise. We now introduce the necessary tools to obtain such a filtration from an arbitrary one.

Definition 2.36. Let I and J be two totally ordered sets, K_I and K_J two filtrations and $r : I \to J$ a monotonically increasing map (satisfying $r(a) \leq r(b)$ if $a \leq b$ with respect to the orders on I and J).
Then K_J is called a *reindexing* of K_I with *reindexing map* r if $K_i = K_{r(i)}$ for all $i \in I$. If r is surjective we say K_J is a *condensation* of K_I. If r is injective we call K_J a *refinement* of K_I.

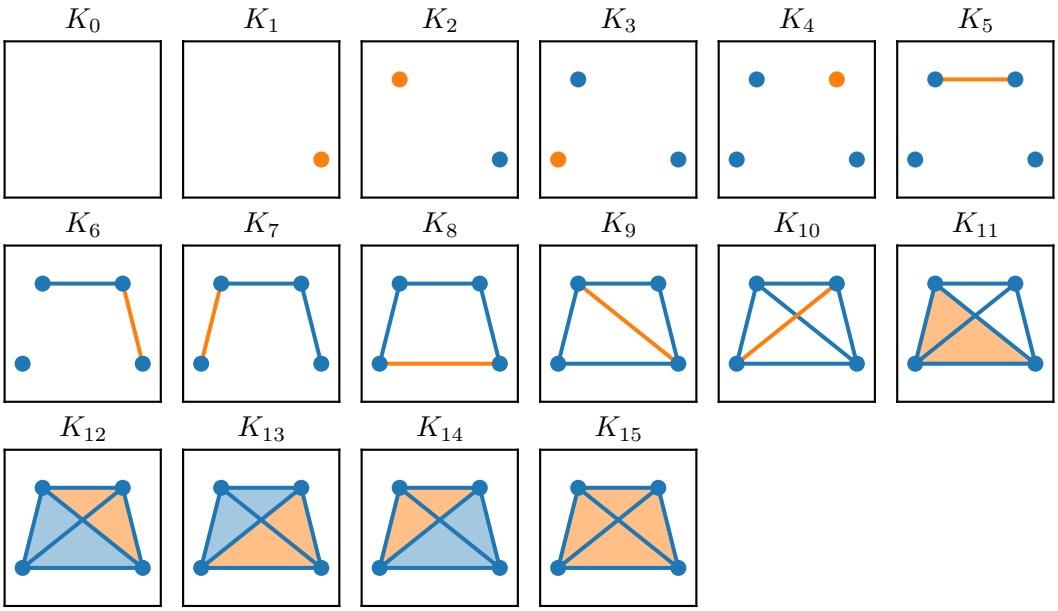


Figure 2.4.: Subcomplexes $K_0, \dots, K_{15}$ of $K = \Delta X$ as introduced in Example 2.4. The single simplex added at each respective index $i > 0$ is highlighted in orange. The sequence $K_0 \subseteq \cdots \subseteq K_{15}$ is a simplexwise filtration of K that refines the essential filtration shown in Figure 2.3.

With Example 2.37 we show how a condensation followed by a refinement produces a simplexwise filtration from the Vietoris-Rips filtration.

Example 2.37. Recall from Example 2.4 the simplicial complex ΔX. Figure 2.3 shows the six distinct complexes that appear in the Vietoris-Rips filtration of ΔX. As a filtration, they are a condensation with respect to the reindexing map

$$\mathbb{R} \;\to\; \left\{ -1, 0, 4, \sqrt{17}, 6, \sqrt{41} \right\}, \qquad t \;\mapsto\; \begin{cases} -1 & \text{if } t < 0, \\ 0 & \text{if } t \in [0, 4), \\ 4 & \text{if } t \in [4, \sqrt{17}), \\ \sqrt{17} & \text{if } t \in [\sqrt{17}, 6), \\ 6 & \text{if } t \in [6, \sqrt{41}), \\ \sqrt{41} & \text{if } t \geq \sqrt{41}. \end{cases}$$

This condensation may then be refined to the simplexwise filtration shown in Figure 2.4 by the reindexing map

$$\left\{ -1, 0, 4, \sqrt{17}, 6, \sqrt{41} \right\} \;\to\; \{0, \dots, 15\}, \qquad \left\{ \begin{matrix} -1 \mapsto 0, & 0 \mapsto 4, & 4 \mapsto 5 \\ \sqrt{17} \mapsto 7, & 6 \mapsto 8, & \sqrt{41} \mapsto 15 \end{matrix} \right\}.$$

Any partially ordered set I may be considered a category with the elements $i \in I$ as

objects and the relations $i \leq j$ as morphisms $i \to j$. This category is called the *poset category* of I and we denote it by $\mathbf{I}$. Any filtration K_I may be viewed as a functor $\mathbf{I} \to \mathbf{Simp}$ as we show with the following proposition.

Proposition 2.38. Let I be a totally ordered set. The assignment

$$K_I : \mathbf{I} \to \mathbf{Simp}, \qquad \begin{Bmatrix} i & \mapsto & K_i, \\ i \leq j & \mapsto & K_i \hookrightarrow K_j \end{Bmatrix}$$

is a functor.

Proof. Let $i \leq j$ and $j \leq k$ be morphisms in $\mathbf{I}$ with composite $i \leq k = (j \leq k) \circ (i \leq j)$. Clearly, $K_i \hookrightarrow K_k = (K_j \hookrightarrow K_k) \circ (K_i \hookrightarrow K_j)$. Moreover, $\mathrm{id}_i = (i \leq i) \mapsto K_i \hookrightarrow K_i = \mathrm{id}_{K_i}$ for any $i \in I$. This shows that that K_I respects compositions and the identity as desired. □

Note that a reindexing map $r : I \to J$ may similarly be considered a functor $\mathbf{I} \to \mathbf{J}$ and expresses K_I as the composition of functors $K_I = K_J \circ r$.

2.6. Persistence Modules

The next step towards persistent homology is taking the homology of a filtration K_I. This not only produces a collection of homology groups $H_\bullet(K_i)$ for $i \in I$ (from the pointwise application of $H_\bullet$) but also a map $H_\bullet(K_i) \to H_\bullet(K_j)$ induced by $K_i \hookrightarrow K_j$ whenever $i \leq j$. Since $H_\bullet$ does not necessarily preserve inclusions the resulting structure is no longer a filtration but rather a diagram of vector spaces indexed over I, called a persistence module. First introduced under this name in [39], persistence modules are the centerpiece of persistence homology and we refer to [11] for an extensive treatment of the topic as well as [7] for a categorical point of view.
Throughout this section K denotes a finite simplicial complex, I an index set, $\mathbf{I}$ its poset category and K_I a filtration of K.

Definition 2.39. A functor

$$V : \mathbf{I} \to \mathbf{Vect}_{\mathbb{F}}, \qquad \begin{Bmatrix} i & \mapsto & V_i, \\ i \leq j & \mapsto & v_{ij} : V_i \to V_j \end{Bmatrix}$$

is called a *persistence module* with respect to I. A persistence module is called *pointwise finite-dimensional* if V_i is finite-dimensional for all $i \in I$. A *submodule* of V is a persistence module W such that W_i is a subspace of V_i for all $i \in I$ and w_{ij} is the restriction of v_{ij} for all $i \leq j$.

Proposition 2.40. The composition of functors

$$H_p \circ K_I : \mathbf{I} \to \mathbf{Vect}_{\mathbb{F}}, \qquad \left\{ \begin{array}{lcl} i & \mapsto & H_\bullet(K_i), \\ i \leq j & \mapsto & \phi_p^{ij} : H_\bullet(K_i) \to H_\bullet(K_j) \end{array} \right\}$$

is a pointwise finite-dimensional persistence module for all $p \in \mathbb{Z}$. Likewise, $H_\bullet \circ K_I$ is a pointwise finite-dimensional persistence module.

Proof. $H_p \circ K_I$ is a functor since by Proposition 2.17 and 2.38 both H_p and K_I are functors. The homology groups of a finite simplicial complex are finite-dimensional hence $H_p \circ K_I$ is pointwise finite-dimensional. By Corollary 2.18 we may replace H_p with $H_\bullet$. □

The next section reviews persistent homology by the means of the persistence module $H_\bullet \circ K_I$. Beforehand we present a few general results about persistent modules, most importantly a structure theorem that asserts the existence of a direct sum decomposition of pointwise finite-dimensional persistence modules. This result is derived by interpreting persistence modules as graded R-modules over $R = \mathbb{F}[t]$ and applying tools and results from commutative algebra [39, 11, 17]. This is beyond the scope of this thesis and Theorem 2.44 (and 2.45) are stated without proof.
We may therefore stay in the categorical setting and work with persistence modules as objects in $\mathbf{Fun}(\mathbf{I}, \mathbf{Vect}_{\mathbb{F}})$. This functor category is abelian since $\mathbf{Vect}_{\mathbb{F}}$ is abelian and thus contains all direct sum (biproducts). Such a direct sum $A \oplus B$ of persistence modules A and B is taken pointwise in $\mathbf{Vect}_{\mathbb{F}}$, that is

$$(A \oplus B)_i \;=\; A_i \oplus B_i \quad \text{for all indices } i \in I.$$

The summands in the aforementioned direct sum decomposition are particularly simple persistence modules called interval persistence modules.

Definition 2.41. A non-empty subset $J \subseteq I$ is called an *interval* if for all $i, k \in J$ $i \leq j \leq k$ implies $j \in J$. We write $\inf(J)$ for the greatest lower bound and $\sup(J)$ for the least upper bound of J in I.
Let $J \subseteq I$ be an interval. The persistence module

$$V(J) : \mathbf{I} \to \mathbf{Vect}_{\mathbb{F}}, \qquad \left\{ \begin{array}{lcll} i & \mapsto & \mathbb{F} & \text{if } i \in J, \\ i & \mapsto & 0 & \text{if } i \notin J, \\ i \leq j & \mapsto & \mathrm{id} & \text{if } i, j \in J, \\ i \leq j & \mapsto & 0 & \text{if } i \notin J \text{ or } j \notin J. \end{array} \right\}$$

is called *interval (persistence) module* with respect to J.

Remark 2.42. In the following, I frequently is a finite ordered subset of $\mathbb{Z}$. Despite this, we use the notation of half-open intervals, writing $[a, b)$ instead of $[a, b-1]$, $[a, \infty)$ instead of $[a, \sup(I)]$ and $(-\infty, b)$ instead of $[\inf(I), b-1]$ because this often simplifies the notation.

Interval modules have a particularly simple structure. For a discrete index set they are of the form

$$V(J) \;=\; \big(\cdots \longrightarrow 0 \longrightarrow 0 \longrightarrow \underbrace{\mathbb{F} \xrightarrow{\mathrm{id}} \mathbb{F} \xrightarrow{\mathrm{id}} \cdots \xrightarrow{\mathrm{id}} \mathbb{F}}_{i \,\in\, J} \longrightarrow 0 \longrightarrow 0 \longrightarrow \cdots\big).$$

This simplicity is characterized by the property of being indecomposable that we introduce with the following proposition.

Proposition 2.43. Let $J \subseteq I$ be an interval and $V(J)$ the corresponding interval persistence module. Then $V(J)$ cannot be expressed as the direct sum of two non-trivial submodules of $V(J)$. It is called *indecomposable*.

Proof. $\mathbb{F}$ as a $\mathbb{F}$-vector space cannot be written as a non-trivial direct sum of $\mathbb{F}$-vector spaces. This extends to persistence modules in a pointwise fashion. We refer to [11, Proposition 2.6] and [10, Lemma 1.3] for more details. □

The following two theorems allow us to decompose any pointwise finite-dimensional persistence module into interval modules and guarantee uniqueness.

Theorem 2.44. Let V be a pointwise finite-dimensional persistence module. Then there exists a multiset $\mathcal{J}$ of intervals of I called the *barcode* of $\mathcal{J}$ such that

$$V \;\cong\; \bigoplus_{J \in \mathcal{J}} V(J).$$

This direct sum decomposition is called the *barcode decomposition* of V.

Proof. We refer to [15] that presents a self-contained proof of this theorem and the proof of [11, Theorem 2.8] that lists a number of useful references. □

The notion of a multiset is necessary here since an interval may identically appear more than once.

Theorem 2.45. Let V be a pointwise finite-dimensional persistence module. Then the barcode $\mathcal{J}$ of V is uniquely determined. It is an isomorphism invariant of V.

Proof. We refer to [11, Theorem 2.7] that proves this theorem using Krull-Remak-Schmidt's theorem [1]. □

Remark 2.46. The assumption that V is pointwise finite-dimensional can be replaced by the assumption that the index set I is finite. If neither is assumed such a decomposition may not exist [11, Theorem 2.8]. Either condition is sufficient for our purposes.

2.7. Persistent Homology

In order to understand how the homology of a simplicial complex K develops over the course of a filtration K_I, not only the groups $H_p(K_i)$ for $i \in I$ but also the maps $\phi_p^{ij} : H_p(K_i) \to H_p(K_j)$ induced by $K_i \hookrightarrow K_j$ for $i \leq j$ need to be considered. Historically, these maps are used to define the *p-th persistent homology group* $H_p^{ij}(K) = \mathrm{im}(\phi_p^{ij})$ and the *p-th persistent Betti number* $\beta_p^{ij} = \mathrm{rank}(\phi_p^{ij})$ that capture to which extent the homology present at index i persists until index j. Tracking specific homology classes using the persistent homology groups is not straightforward however:
Suppose one has computed bases for $H_p(K_i)$ and $H_p(K_j)$ and wishes to track a homology class $[z] \in H_p(K_i)$ (corresponding to a distinct topological feature of K_i) that is known to persist from i to j. Unless these bases are subject to specific constraints, $\phi_p^{ij}([z]) \in H_p(K_j)$ is not itself a basis element but a linear combination. We can correct for this by performing a change of basis transformation but as soon as more than two indices need to taken into account it becomes laborious to keep all bases compatible. Moreover, it is not clear from the persistent homology groups at which index $[z]$ is introduced as a new homology class unless all indices smaller than i are considered.
In the following subsection, we develop the notion of a compatible basis of $C_\bullet(K)$ that alleviates these issues for all indices at once. As it turns out, by constructing such a basis we also produce the barcode decomposition of the persistence module $H_\bullet \circ K_I$. From this decomposition the persistent homology groups and Betti numbers are easily derived. For this reason the computation of the barcode decomposition has become a popular method to analyze the topological features of filtrations of simplicial complexes and the rather general term *persistent homology* is often used in this narrower sense, denoting the barcode decomposition of $H_\bullet \circ K_I$. In order to compute persistent homology, we introduce a tool called matrix reduction in Subsection 2.7.2 and apply it in Subsection 2.7.3 to derive a compatible basis from the canonical basis $\mathcal{C}(K)$ of $C_\bullet(K)$ which in turn produces the barcode decomposition of $H_\bullet \circ K_I$. The computational aspects of this approach are the topic of Chapter 3.

Two notes on our assumptions. In this section, we restrict to essential simplexwise filtrations K_I of a finite simplicial complex K in order to concisely work towards the barcode decomposition. As part of the computational setup in Section 3.2, we provide the means to generalize to arbitrary filtrations. We further assume without loss of generality that the index set is given by $I = \{0, \ldots, |K|\}$ with the usual order.

2.7.1. Compatible Bases

We start with definition of three constraints on bases of $C_\bullet(K)$ that vaguely reflect the structural properties of the barcode decomposition.

Definition 2.47. Let $\mathcal{S}$ be a basis of $C_\bullet(K)$. We call $\mathcal{S}$

1. *compatible with the grading* if $\mathcal{S}$ restricts to a basis of $C_p(K)$ for all $p \in \mathbb{Z}$,
2. *compatible with the filtration* K_I if $\mathcal{S}$ restricts to a basis of $C_\bullet(K_i)$ for any $i \in I$,
3. *compatible with the boundary map* $\partial_\bullet$ if $\mathcal{S}$ restricts to a basis $\mathcal{Z}$ of $Z_\bullet(K)$ that itself restricts to a basis $\mathcal{B}$ of $B_\bullet(K)$ such that for any $s_\ell \in \mathcal{B}$ there exists an element $s_i \in \mathcal{S}$ that satisfies $\partial_\bullet(s_i) = s_\ell$.

If $\mathcal{S}$ is compatible in all three respects we call it *compatible*.

Proposition 2.48. The canonical basis $\mathcal{C}(K)$ of $C_\bullet(K)$ is compatible with the grading and the filtration K_I.

Proof. Since $\mathcal{C}(K)$ is defined as union of $\mathcal{C}_p(K)$ for all $p \in \mathbb{Z}$ it is compatible with the grading. By definition, $\mathcal{C}(K)$ is the set of simplices in K (oriented in a prescribed way) which restricts to the set of simplices in K_i (oriented analogously). Thus $\mathcal{C}(K_i) \subseteq \mathcal{C}(K)$ is satisfied for $i \in I$ showing compatibility with the filtration. □

Note that if K contains at least one simplex σ of dimension 1, $\mathcal{C}(K)$ is not compatible with $\partial_\bullet$ since $\partial_1(\sigma)$ is the sum of two 0-simplices (which is neither 0 nor a simplex).
In the following let $\mathcal{S} = \mathcal{S}(K)$ be a compatible basis of K and let $\mathcal{B} \subseteq \mathcal{Z} \subseteq \mathcal{S}$ be bases of $B_\bullet(K)$ and $Z_\bullet(K)$ respectively. We write $\mathcal{S}(K_0)$ for the empty set and $\mathcal{S}(K_i)$ for the restriction of $\mathcal{S}$ to K_i. We further write $\mathcal{S}_p$ for the restriction to $C_p(K)$ for any $p \in \mathbb{Z}$. Going forward we use this notation not only for $\mathcal{S}$ but any basis occurring in this context. By the fact that the various restrictions of $\mathcal{S}$ are bases of subspaces of $C_\bullet(K)$ it follows that the constraints of Definition 2.47 commute with each other. We show this for one particularly relevant instance in the following lemma.

Lemma 2.49. The basis $\mathcal{Z} \subseteq \mathcal{S}$ restricts to a basis of $Z_\bullet(K_i)$ and $\mathcal{B} \subseteq \mathcal{Z}$ restricts to a basis of $\mathcal{B}_\bullet(K_i)$ for any $i \in I$. Consequently the dashed arrows exist and the diagrams

$$\begin{array}{ccc} \mathcal{Z}(K_i) & \dashrightarrow & \mathcal{Z} \\ \downarrow & & \downarrow \\ \mathcal{S}(K_i) & \hookrightarrow & \mathcal{S} \end{array} \qquad \begin{array}{ccc} \mathcal{B}(K_i) & \dashrightarrow & \mathcal{B} \\ \downarrow & & \downarrow \\ \mathcal{Z}(K_i) & \dashrightarrow & \mathcal{Z} \end{array}$$

commute.

Proof. First note that $Z_\bullet(K_i) \subseteq Z_\bullet(K)$ is a subspace since it is the kernel of $\partial_\bullet$ restricted to $C_\bullet(K_i)$. If we assume that there exists $z \in \mathcal{Z}(K_i) \setminus \mathcal{Z}$ then $\mathcal{Z} \cup \{z\}$ is linearly independent as a subset of the basis $\mathcal{S}$ that lies in $Z_\bullet(K)$. This is in contradiction with $\mathcal{Z}$ being a basis of $Z_\bullet(K)$. The existence of $\mathcal{B}(K_i) \hookrightarrow \mathcal{B}$ is shown analogously. □

Not only is the compatible basis $\mathcal{S}$ subject to the constraints of Definition 2.47 but also to the isomorphisms

$$C_p(K_i) \;\cong\; Z_p(K_i) \oplus B_{p-1}(K_i), \qquad Z_p(K_i) \;\cong\; B_p(K_i) \oplus H_p(K_i)$$

of Proposition 2.14 (for all $p \in \mathbb{Z}$ and $i \in I$). The following corollary to Lemma 2.49 is a direct consequence of the latter isomorphism and the compatibility of $\mathcal{S}$.

Corollary 2.50. The basis $\mathcal{Z}(K_i)$ restricts to a basis

$$\mathcal{E}(K_i) \;=\; \mathcal{Z}(K_i) \setminus \mathcal{B}(K_i)$$

of $H_\bullet(K_i)$ for all $i \in I$. □

The following lemma points out another consequence of the above isomorphisms and examines the manner in which $\mathcal{Z}(K_{i-1})$, $\mathcal{B}(K_{i-1})$ and $\mathcal{E}(K_{i-1})$ change as $\mathcal{S}(K_{i-1})$ is extended to $\mathcal{S}(K_i)$.

Lemma 2.51. Let $\sigma_i \in K$ be the p-simplex such that $K_i = K_{i-1} \cup \{\sigma_i\}$ and let $s_i \in \mathcal{S}$ be a basis element such that $\mathcal{S}(K_i) = \mathcal{S}(K_{i-1}) \cup \{s_i\}$. Then s_i is a p-chain containing σ_i (as a non-zero scalar multiple) and precisely one of the two holds:

1. $\mathcal{E}(K_i) = \mathcal{E}(K_{i-1}) \cup \{s_i\}$.
2. There exists a unique cycle $s_\ell \in \mathcal{E}(K_{i-1})$ such that $\mathcal{B}(K_i) = \mathcal{B}(K_{i-1}) \cup \{s_\ell\}$ and $\mathcal{E}(K_i) = \mathcal{E}(K_{i-1}) \setminus \{s_\ell\}$.

In particular there exists a unique pairing (s_ℓ, s_i) of all elements in $\mathcal{S} \setminus \mathcal{E}$ that is determined by case (2). We call ℓ the *pivot index* of s_i and denote it by $\mathrm{pivot}(s_i)$.

Proof. Since $\mathcal{S}$ is compatible with the grading, any element therein lies in $C_p(K)$ for a unique $p \in \mathbb{Z}$ and because the addition of σ_i extends the basis of p-chains, s_i extends $\mathcal{S}_p$ as a p-chain. Furthermore σ_i must be contained in s_i since σ_i is the linear combination of elements in $\mathcal{S}(K_i)$ but no s_j for $j < i$ may contain σ_i due to compatibility with the filtration.
Since $C_p(K_j) \cong Z_p(K_j) \oplus B_{p-1}(K_j)$ for all $j \in I$, the compatibility of $\mathcal{S}$ forces exactly one of $\mathcal{Z}_p$ or $\mathcal{B}_{p-1}$ to extend by precisely one element in order to reflect the extension of $\mathcal{S}(K_{i-1})$ by s_i.

1. If the former is the case, $\mathcal{Z}(K_i) = \mathcal{Z}(K_{i-1}) \cup \{s_i\}$ follows by compatibility with the filtration. Moreover, s_i can not be a boundary in $C_\bullet(K_i)$ since it contains σ_i which has no coface in K_i by virtue of the filtration. Thus $\mathcal{E}(K_{i-1})$ is extended by s_i to $\mathcal{E}(K_i)$ (as subsets of $\mathcal{Z}(K_{i-1})$ and $\mathcal{Z}(K_i)$ respectively).
2. In the latter case the splitting isomorphism necessitates the extension of $\mathcal{B}(K_{i-1})$ by precisely one basis element. This element however cannot be the p-chain s_i since $\mathcal{S}$ is assumed to be compatible with the grading. By compatibility with the filtration,

the only other possibility to account for the extension is that a linear combination of elements in $\mathcal{S}(K_{i-1}) \cup \{s_i\}$ bounds a in $C_\bullet(K_{i-1})$ unbounded cycle $s_\ell \in \mathcal{E}(K_{i-1})$ which now extends $\mathcal{B}(K_{i-1})$ to $\mathcal{B}(K_i)$. Moreover, s_ℓ is uniquely determined: A second distinct candidate s_k would likewise need to be bounded in $C_\bullet(K_i)$ as a result of the addition of s_i and thus both s_ℓ and s_k would lie in $B_{p-1}(K_i)$. But then $\dim(B_{p-1}(K_i)) = \dim(B_{p-1}(K_{i-1})) + 2$ since $\{s_\ell, s_k\} \cup \mathcal{B}(K_{i-1}) \subseteq \mathcal{S}$ which is in contradiction with the assumption that $\dim(B_{p-1}(K_{i-1}))$ increases by one.

In the second case, the element s_ℓ is the image of s_i under $C_p(K_i) \xrightarrow{\cong} Z_p(K_i) \oplus B_{p-1}(K_i)$ given by ∂_p on $\mathcal{S}_p(K_i)$. Since $\mathcal{S}$ is compatible with the filtration this is also the case when K_i is replaced by K resulting in a uniquely determined pair (s_ℓ, s_i) in $\mathcal{S} \setminus \mathcal{E}$. Since the case distinction is applicable for any $i \in I$, those elements that do not remain in $\mathcal{E}$ must be paired at some index, yielding the desired pairing of all elements in $\mathcal{S} \setminus \mathcal{E}$. □

Remark 2.52. In case two of the proof of Lemma 2.51 the unbounded $(p-1)$-cycle s_ℓ becomes bounded by a chain $c \in C_p(K_i)$ as the consequence of adding s_i to $\mathcal{S}(K_{i-1})$. Since $\mathcal{S}$ is compatible with the boundary map there exists $s_{i'}$ such that $\partial_\bullet(s_{i'}) = s_\ell$. From this it follows that $c = s_{i'} = s_i$. The proof of Lemma 2.51 does not require this condition however and Theorem 2.53 holds without it.

Starting at $\mathcal{Z}(K_0) = \emptyset$, Lemma 2.51 informs on how $\mathcal{Z}$ develops as a filtered basis of $Z_\bullet(K)$: Either an unbounded cycle (a new homology class) is added or such a cycle becomes bounded, paired and moved to extend the basis of boundaries (removing an existing homology class). Put differently, from the index of its addition to $\mathcal{E} \subseteq \mathcal{S}$ an unbounded cycle persists identically as a basis element of $H_\bullet(K_i)$ (under the identification of Corollary 2.50) until (excluding) the index at which a bounding chain becomes available and it is moved from $\mathcal{E}$ to $\mathcal{B}$ or the filtration ends and it remains in $\mathcal{E}$. It is now apparent how the barcode decomposition arises from $\mathcal{Z}$.

Theorem 2.53. Let $\mathcal{S} = \{s_1, \dots, s_{|K|}\}$ be a compatible basis of $C_\bullet(K)$ such that $\mathcal{S}(K_i) = \mathcal{S}(K_{i-1}) \cup \{s_i\}$ for all $i \in I$. Using the interval notation of Remark 2.42, the barcode decomposition of $H_\bullet \circ K_I$ is then given by

$$H_\bullet \circ K_I \cong \bigoplus_{s_j \in \mathcal{E}} V\big[j, \infty\big) \oplus \bigoplus_{\substack{s_\ell \in \mathcal{B} \\ \ell = \text{pivot}(s_i)}} V\big[\ell, i\,\big).$$

The interval modules are generated by $s_i \in \mathcal{E}$ and $s_\ell \in \mathcal{B}$ respectively. □

Each index in I occurs exactly once in the barcode decomposition, either as the start or endpoint of a unique interval in the barcode. With the addition of each simplex σ_i to the filtration one homology class is either created or destroyed.

Definition 2.54. Let $\mathcal{S}$ be a compatible basis of $C_\bullet(K)$ and $J \in \mathcal{J}$ be an interval in the barcode of $H_\bullet \circ K_I$.

1. We call $i = \inf(J)$ a *birth index* and σ_i a *birth simplex*. If $j = \sup(J)$ exists then we call it a *death index* and σ_j a *death simplex*.
2. If J is of the form $[j, \infty)$ it is called *essential*. The index i, the corresponding simplex σ_j and the chain $s_j \in \mathcal{S}$ are analogously called *essential*.
3. If J is of the form $[\ell, i)$ it is called *non-essential*. Similarly, the indices ℓ and i, the simplices σ_ℓ and σ_i as well as the chains s_ℓ and s_i are called a *non-essential pair*.

So far the condition $\partial_\bullet(s_i) = s_\ell$ of Definition 2.47 was not necessary. With the following corollary to Lemma 2.51 we show that if it is satisfied for all non-essential pairs the chain complex $C_\bullet(K)$ itself decomposes, justifying its inclusion in Definition 2.47.

Corollary 2.55. Let $\mathcal{S}$ be a compatible basis of $C_\bullet(K)$. Then the chain complex $C_\bullet(K)$ is the direct sum of chains

$$\cdots \longrightarrow 0 \longrightarrow s_j\,\mathbb{F} \longrightarrow 0 \longrightarrow 0 \longrightarrow \cdots$$

$$\cdots \longrightarrow 0 \longrightarrow s_i\,\mathbb{F} \xrightarrow{\partial_\bullet} s_\ell\,\mathbb{F} \longrightarrow 0 \longrightarrow \cdots$$

taken over all essential chains s_j and non-essential pairs (s_ℓ, s_i).

Proof. Since $\mathcal{S}$ is assumed to be compatible with the grading, $\mathcal{S}_p$ is a basis of the chain group $C_p(K)$ for all $p \in \mathbb{Z}$ and $C_p(K) \cong \bigoplus_{s_k \in \mathcal{S}_p} s_k\,\mathbb{F}$. Since the direct sum in $\mathbf{Ch}(\mathbf{Vect}_\mathbb{F})$ is defined degreewise the claim now follows from the compatibility of $\mathcal{S}$ (asserting $\partial_\bullet(s_i) = s_\ell$ for all non-essential pairs (s_ℓ, s_i)) and Lemma 2.51 (implying that the union of all essential and non-essential chains is precisely $\mathcal{S}$). □

Remark 2.56. The barcode of $H_\bullet \circ K_I$ always contains the interval $[1, \infty)$ (in degree 0) if K_I contains at least one non-empty complex. This can be seen as follows: As a consequence of Proposition 2.15, the vertex added at $i = 1$ to K_I represents a homology class in degree zero. By compatibility with the filtration, a scalar multiple of this first vertex must be an element of any compatible basis $\mathcal{S}$ and since (a scalar multiple of) a single vertex cannot be the boundary of any 1-chain it must be essential.

The remainder of this section is concerned with the construction of a compatible basis of $C_\bullet(K)$ from the canonical basis $\mathcal{C}(K)$. We achieve this by performing a sequence of transformations that establish compatibility with the boundary map while retaining the compatibility with the grading and filtration that $\mathcal{C}(K)$ already has (see Proposition 2.48). These basis transformations are steered by decomposing the matrix that represents the boundary map $\partial_\bullet$ with respect to $\mathcal{C}(K)$ in both domain and codomain.

2.7.2. Matrix Reduction

The general objective of matrix reduction is to produce a basis of the kernel and image of its underlying linear map subject to an order constraint. When applied to the matrix representing the boundary map $\partial_\bullet$ this constraint ensures compatibility with the filtration. For the remainder of this section however let $M \in \mathbb{F}^{m\times n}$ be any matrix.

Definition 2.57. We define the *pivot index* $\mathrm{pivot}(M_j)$ of a column M_j to be the largest row index $i \in \{1,\dots,m\}$ such that $M_{ij} \neq 0$. If M_j is the zero column we set $\mathrm{pivot}(M_j) = 0$. The pivot index is thus a map

$$\mathrm{pivot}\big(M_{(-)}\big) : \{1,\dots,n\} \to \{0,1,\dots,m\}, \qquad j \mapsto \mathrm{pivot}(M_j).$$

The *set of pivots of* M denoted $\mathrm{pivots}(M)$ is defined to be the set of all non-zero pivot indices of M.
We call a column M_j *reduced* if the pivot index $\mathrm{pivot}(M)$ cannot be reduced by column additions of scalar multiples of columns M_i for $i < j$. If all columns of M are reduced, we call M *reduced*.

It is no coincidence that we reuse the term 'pivot index' here (first appearing in Lemma 2.51): In the next subsection the pivot index of certain columns specifies the death index of a non-essential pair in the barcode decomposition of $H_\bullet \circ K_I$ which is in agreement with its previous meaning.

Lemma 2.58. Let M_i and M_j be two distinct non-zero columns of M that are reduced. Then $\mathrm{pivot}(M_i) \neq \mathrm{pivot}(M_j)$.

Proof. Without loss of generality we assume $i < j$. If $\ell = \mathrm{pivot}(M_i) = \mathrm{pivot}(M_j)$ then $M_j - (M_{\ell j}/M_{\ell i})M_i$ has a strictly smaller pivot index than ℓ and M_j is not reduced. This yields the claim by contraposition. □

Lemma 2.59. A non-zero column M_j is reduced if for all $i < j$ the column M_i is reduced and $\mathrm{pivot}(M_i) \neq \mathrm{pivot}(M_j)$.

Proof. We show that $\ell = \mathrm{pivot}(M_j)$ cannot be reduced further by adding any linear combination $L = \sum_{i<j} \lambda_i M_i$ to M_j. If $L \neq 0$ then by Lemma 2.58 (using that M_i is assumed to be reduced for any $i < j$) there exists a unique $k < j$ such that $m = \mathrm{pivot}(M_k)$ is maximal among columns $M_i \neq 0$ with $\lambda_i \neq 0$ and $\mathrm{pivot}(L) = m$. Since $m \neq \ell$ by assumption, adding L to M_j may only increase its pivot index. □

Proposition 2.60. For any matrix $M \in \mathbb{F}^{m\times n}$ there exists an upper unitriangular matrix $V \in \mathbb{F}^{n\times n}$ (upper triangular such that $V_{ii} = 1$ for all $i \in \{1,\dots,n\}$) such that $R = MV$ is reduced.

Proof. We construct V and thereby R iteratively. Initially set $V := \mathrm{Id}$ and $R := M$. The column $R_1 = M_1$ is reduced by definition, $V = \mathrm{Id}$ is unitriangular and $R_1 = MV_1$. Assume now that the column R_i is reduced for $i < j$ whereas R_j is not reduced, that is $\ell = \mathrm{pivot}(R_j) = \mathrm{pivot}(R_k) \neq 0$ for some $k < j$. We then update

$$R_j \;:=\; R_j + \lambda R_k \qquad \text{and} \qquad V_j \;:=\; V_j + \lambda V_k \qquad \text{where} \quad \lambda = -\frac{R_{\ell j}}{R_{\ell k}}.$$

Now the ℓ-th entry of R_j is zero and the pivot index has been reduced. If V was unitriangular beforehand then it still is afterwards due to $k = \mathrm{pivot}(V_k) < \mathrm{pivot}(V_j) = j$. We now repeat the above update until the pivot index of R_j is either zero and R_j thus trivially reduced or it is unique among those of R_i for $i < j$. In this case R_j is reduced as a consequence of Lemma 2.59. $R_j = MV_j$ is ensured by the structure of the update (V_j recording the columns added to R_j).
By reducing all columns of R in this fashion with increasing column index we obtain the desired decomposition $R = MV$. □

The iterative manner in which the decomposition $R = MV$ is constructed makes it extensible.

Corollary 2.61. In the notation of Proposition 2.60 let V be an upper unitriangular matrix such that $R = MV$ is reduced. Let M' denote the extension of M by a column M_{n+1}. Then we can likewise extend R by a column R_{n+1} and V by a column V_{n+1} to R' and V' respectively such that V' is unitriangular and $R' = M'V'$ reduced. □

Proposition 2.62. Let V and V' be two distinct upper unitriangular matrices such that $R = MV$ and $R' = MV'$ are reduced. Then $\mathrm{pivot}(R_{(-)}) = \mathrm{pivot}(R'_{(-)})$.

Proof. Let $j \in \{1, \dots, n\}$ be any column index. It is enough to show that $\mathrm{pivot}(R_j) = \mathrm{pivot}(R'_j)$. The columns V_j and V'_j encode linear combinations

$$L \;=\; \sum_{i<j} V_{ij} M_i, \qquad L' \;=\; \sum_{i<j} V'_{ij} M_i$$

such that $R_j = L + M_j$ and $R'_j = L' + M_j$ (note that $V_{jj} = V'_{jj} = 1$ by unitriangularity). If we assume (without loss of generality) that $\mathrm{pivot}(R_j) < \mathrm{pivot}(R'_j)$ it follows that R'_j cannot be reduced: We have

$$R_j \;=\; L + M_j \;=\; L + (R'_j - L') \;=\; R'_j + L - L'$$

which means that $\mathrm{pivot}(R'_j)$ can be reduced further to $\mathrm{pivot}(R_j)$ by the column additions encoded in $L - L'$. By applying $(V')^{-1}$ we can express $L - L'$ as a linear combination of columns R'_i for $i < j$ thereby showing that $\mathrm{pivot}(R'_j)$ can be further reduced by adding scalar multiples of those columns R'_i.
By contraposition it follows that $\mathrm{pivot}(R_j) = \mathrm{pivot}(R'_j)$. A different proof of this proposition can be found in [14]. □

2.7.3. A Compatible Basis by Matrix Reduction

We return to persistent homology and apply matrix reduction to the matrix representing the boundary map. Recall from Definition 2.33 that a simplexwise filtration K_I induces a total order on the set of all (oriented) simplices, the filtration order.

Definition 2.63. With respect to the canonical basis $\mathcal{C}(K)$ of $C_\bullet(K)$ in filtration order, we represent the boundary map $\partial_\bullet$ as the matrix

$$D = \begin{pmatrix} | & | & & | \\ \partial_\bullet(\sigma_1) & \partial_\bullet(\sigma_2) & \cdots & \partial_\bullet(\sigma_{|K|}) \\ | & | & & | \end{pmatrix}, \qquad \sigma_i \in \mathcal{C}(K)$$

where the i-th column is given by the coordinate vector of $\partial_\bullet(\sigma_i)$ relative to $\mathcal{C}(K)$. We call D the *boundary matrix* with respect to K_I.

Example 2.64. Recall from Example 2.4 the full simplex ΔX on the four point space X and from Figure 2.4 the simplexwise refinement of the Vietoris-Rips filtration of ΔX. The boundary matrix with respect to this filtration is given by

$$D \;=\; \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & 1 & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 1 & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}$$

representing the boundary map $\partial_\bullet$ of the chain complex on ΔX over $\mathbb{F}_2$ (dots representing zeros).

Proposition 2.65. There exists an ordered basis $\mathcal{V}$ of $C_\bullet(K)$ and change of basis transformation from $\mathcal{V}$ to $\mathcal{C}(K)$ in filtration order that is represented by an upper unitriangular matrix V such that $R = DV$ is reduced and $\mathcal{V}$ is compatible with both the grading and the filtration.

Proof. By applying Proposition 2.60 to the boundary matrix D we obtain $R = DV$ such that R is reduced and V is an upper unitriangular matrix. V is the change of basis transformation of the basis $\mathcal{V}$ given by the ordered set of columns V_j to $\mathcal{C}(K)$ under identification of coordinate vector (relative to $\mathcal{C}(K)$) and vector space element of $C_\bullet(K)$. The sets of row indices $\{i \in I \mid D_{ij} \neq 0, \dim(\sigma_j) = p\}$ are disjoint for all $p \in \mathbb{Z}$ since $\mathcal{C}(K)$ is compatible with the grading. The reduction of a column D_j therefore requires only those columns that corresponding to simplices of the same dimension as σ_j, preserving

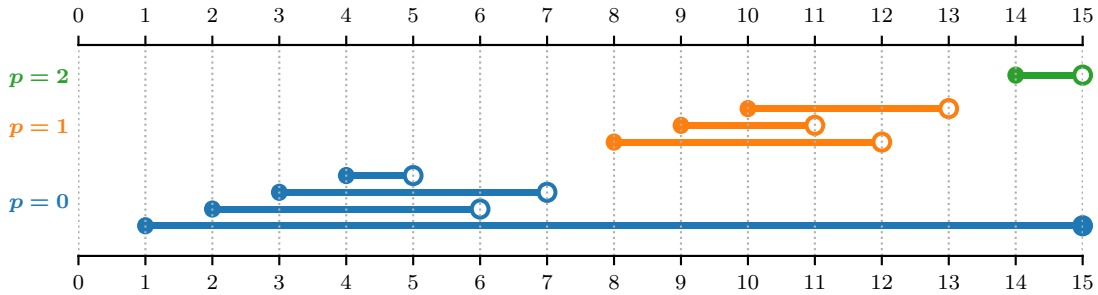


Figure 2.5.: The barcode of $H_\bullet \circ K_I$, where K_I is the simplexwise filtration shown in Figure 2.4 and chain groups are taken over $\mathbb{F}_2$. Intervals corresponding to non-essential pairs are visualized as half-open intervals (see Remark 2.42) and the interval corresponding to the single essential index is visualized as a closed interval. Endpoints are marked by filled circles if they are included and empty circles if they are excluded. The persistent homology for $p > 2$ is trivial. The generators of the corresponding interval modules are visualized in Figure 2.6.

compatibility with the grading. The compatibility with the filtration of $\mathcal{C}(K)$ is preserved since only columns D_i for $i < j$ are candidates for the reduction of a column D_j. A column $V_i \in \mathcal{V}$ is thus the linear combination of elements in $\mathcal{C}(K_i)$ and the first i elements of $\mathcal{V}$ form a basis of $C_\bullet(K_i)$ for all non-zero $i \in I$. □

Remark 2.66. The identification of a coordinate vector $v \in \mathbb{F}^n$ relative to $\mathcal{C}(K)$ with the thereby determined vector in $C_\bullet(K)$ is given by

$$(v_1, \dots, v_n)^T \;\mapsto\; v_1 c_1 + \cdots + v_n c_n, \qquad v_1, \dots, v_n \in \mathbb{F}, \;\; c_1, \dots, c_n \in \mathcal{C}(K)$$

that is an isomorphism $\mathbb{F}^n \to C_\bullet(K)$ (where $n = |K|$). In the following we use this identification without further comment, in particular to identify the columns of R or V with bases of subspaces of $C_\bullet(K)$.

With the following Proposition we show that the matrices V and R obtained by reducing the boundary matrix D contain a basis of $Z_\bullet(K)$ and $B_\bullet(K)$ respectively.

Proposition 2.67. Let V be an upper unitriangular matrix such that $R = DV$ is reduced. Then the ordered sets of columns

$$\mathcal{Z} = \{\, V_j \mid R_j = 0 \,\} \qquad \text{and} \qquad \mathcal{B} = \{\, R_j \mid R_j \neq 0 \,\}$$

are bases of the group of cycles $Z_\bullet(K)$ and boundaries $B_\bullet(K)$ respectively. $\mathcal{Z}$ and $\mathcal{B}$ are furthermore compatible with the grading and the filtration.

Proof. The columns in $\mathcal{Z}$ are linearly independent as a subset of the column space $\mathcal{V}$ of V that is a basis of $C_\bullet(K)$. The columns $R_j \in \mathcal{B}$ are linearly independent as a consequence

of Lemma 2.58 which asserts that the pivot index of any R_j is unique. From the definition it immediately follows that $\mathcal{Z}$ lies in the kernel and $\mathcal{B}$ in the image of $\partial_\bullet$. The rank-nullity theorem [25, Theorem 3.12] then establishes $\mathcal{Z}$ and $\mathcal{B}$ as a basis of $Z_\bullet(K)$ and $B_\bullet(K)$ respectively.
Compatibility of $\mathcal{Z}$ (with the grading and filtration) follows directly from Proposition 2.65. Since the matrix reduction operates on V and R in a similar manner, its proof is applicable to R without conceptual change. □

$\mathcal{B}$ is not necessarily a subset of $\mathcal{Z}$ however and these two bases are thus not suitable candidates to be extended to a compatible basis of $C_\bullet(K)$ yet. The following proposition resolves this issue by replacing elements in $\mathcal{Z}$ with elements from $\mathcal{B}$.

Proposition 2.68. In the notation of Proposition 2.67 we define the ordered sets of columns

$$\mathcal{E} \;=\; \big\{\, V_j \mid R_j = 0,\; j \notin \text{pivots}\big(R\big) \,\big\}, \qquad \widetilde{\mathcal{Z}} \;=\; \mathcal{E} \cup \mathcal{B}, \qquad \mathcal{G} \;=\; \big\{\, V_j \mid R_j \neq 0 \,\big\}.$$

Then $\widetilde{\mathcal{Z}}$ a basis of $Z_\bullet(K)$ that restricts to $\mathcal{E}$, a basis of $H_\bullet(K)$, and $\mathcal{G} \cup \mathcal{E} \cup \mathcal{B}$ is a compatible basis of $C_\bullet(K)$.

Proof. Since R is reduced Lemma 2.58 ensures that any pivot index $\text{pivot}(R_j)$ is unique if it is non-zero. Any column $V_\ell \in \mathcal{Z} \setminus \mathcal{E}$ can thus be exchanged with the unique boundary $R_j \in \mathcal{B}$ such that $\text{pivot}(R_j) = \ell = \text{pivot}(V_\ell)$. Exchanging all columns of V in $\mathcal{Z} \setminus \mathcal{E}$ in this manner preserves the triangular structure and the resulting column set $\mathcal{G} \cup \mathcal{E} \cup \mathcal{B}$ is thus a basis of $C_\bullet(K)$ that retains compatibility with the grading and the filtration.
$\widetilde{\mathcal{Z}}$ lies in $Z_\bullet(K)$ since both $\mathcal{B}$ as a basis of $B_\bullet(K)$ and $\mathcal{E}$ as a subset of $\mathcal{Z}$ do. It is a basis of $Z_\bullet(K)$ since it is a linearly independent set (as subset of $\mathcal{G} \cup \mathcal{E} \cup \mathcal{B}$) that has the same cardinality as $\mathcal{Z}$ (since the exchange of elements is one-to-one). Finally, from $R = DV$ it follows that $R_j = DV_j$ is satisfied for all $V_j \in \mathcal{G}$. In other words, $\partial_\bullet(s_i) = s_\ell$ is satisfied for all non-essential pairs (s_i, s_ℓ) in $\mathcal{G} \cup \mathcal{E} \cup \mathcal{B}$ showing compatibility with the boundary map. □

We conclude this section by returning to the full simplex on the four point space of Example 2.4, in the following K, and its simplexwise filtration shown in Figure 2.4.

Example 2.69. By reducing the boundary matrix of Example 2.64 and applying Proposition 2.67 we obtain bases $\mathcal{Z}$ of $Z_\bullet(K)$ and $\mathcal{B}$ of $B_\bullet(K)$. Figure 2.6 visualizes these bases in blue and orange respectively. Following Proposition 2.68 we may exchange cycles for boundaries to obtain a compatible basis of $C_\bullet(K)$. In this instance the exchange is necessary whenever the visualized basis element of $\mathcal{B}$ and $\mathcal{Z}$ differ at any one index in Figure 2.6, that is for $i \in \{2, 3, 4, 9, 10\}$. Figure 2.5 visualizes the barcode of the barcode decomposition that arises from this compatible basis as an application of Theorem 2.53.

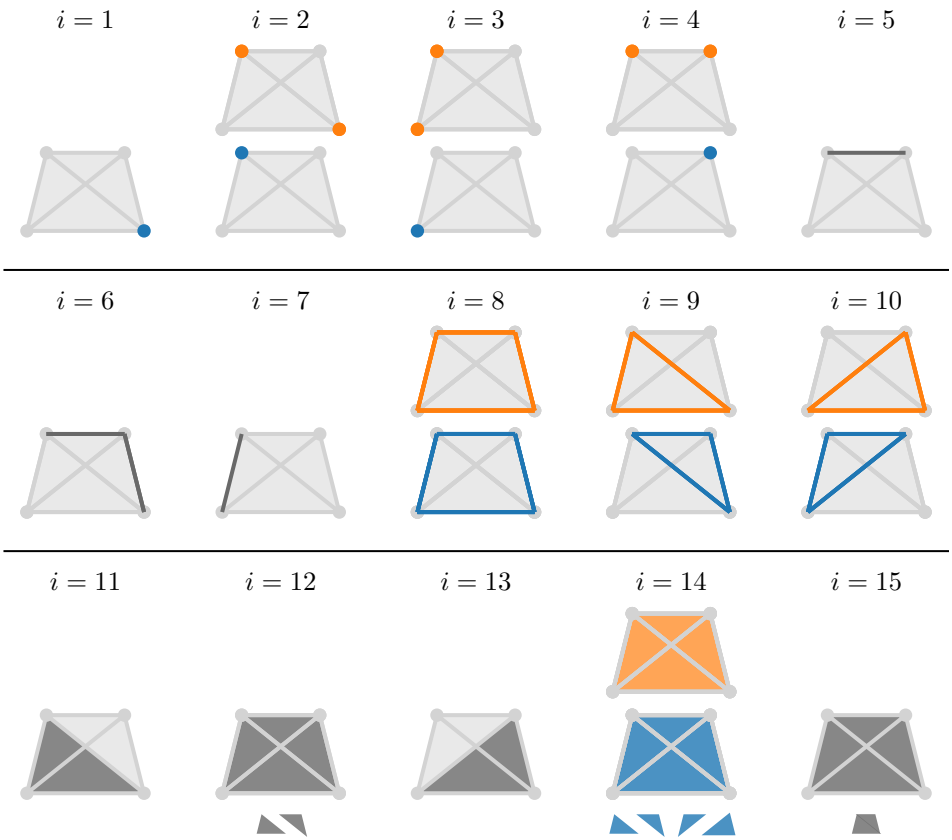


Figure 2.6.: Proposition 2.67 and 2.68 applied to the simplexwise filtration Figure 2.4. Visualized are the bases $\mathcal{B}$ (top rows, orange) as well as $\mathcal{Z}$ and $\mathcal{G}$ (bottom rows, blue and gray respectively) that are the result of reducing the boundary matrix of Example 2.64. The minimized simplices at indices $i \in \{12, 14, 15\}$ clarify which chain is shown.
Exchanging cycles of $\mathcal{Z}$ for the boundaries above them yields $\widetilde{\mathcal{Z}}$ and the compatible basis $\widetilde{\mathcal{Z}} \cup \mathcal{G}$ of $C_\bullet(K)$. The compatibility constraints can be verified visually: Since any basis element contains only simplices of a single degree $\widetilde{\mathcal{Z}} \cup \mathcal{G}$ is compatible with the grading. By comparison with Figure 2.4 it is clear that it is also compatible with the filtration. Lastly, the boundaries at indices $2, 3, 4, 8, 9, 10$ and 14 are bounded by the chains at indices $6, 7, 5, 12, 11, 13$ and 15 respectively. This demonstrates compatibility with the boundary map and yields all non-essential index-pairs of the barcode shown in Figure 2.5.

2.8. Persistent Relative Homology

The relative homology $H_\bullet(K, L)$ of a finite simplicial complex K and a subcomplex $L \subseteq K$ may be placed into the persistence settings in two different ways. Either K is filtered while the relative subcomplex L is held constant or vice versa.

1. In the former case, $i \leq j$ induces an inclusion $C_\bullet(K_i, L \cap K_i) \hookrightarrow C_\bullet(K_j, L \cap K_j)$ and we may apply the methods of Section 2.7 for the analysis of the corresponding persistence module via a compatible basis of $C_\bullet(K, L)$.
2. In the latter case, $i \leq j$ induces a projection $C_\bullet(K, L_i) \twoheadrightarrow C_\bullet(K, L_j)$. While this still induces a map in homology and thereby a persistence module, the projections of relative chain groups are in conflict with the notion of a compatible basis and matrix reduction which are both based on inclusions. Instead of adapting Section 2.7 we follow [17] and describe how its barcode decomposition is determined by the persistent (absolute) homology of the filtration L_I of L.

As in Section 2.7 we restrict to essential simplexwise filtrations K_I of a finite simplicial complex K with subcomplex L and assume $I = \{0, \ldots, |K|\}$. We start with the first type of relative persistent homology.

Proposition 2.70. The inclusion $C_\bullet(K_i) \hookrightarrow C_\bullet(K_j)$ induces an inclusion of chain complexes

$$C_\bullet(K_i, L \cap K_i) \hookrightarrow C_\bullet(K_j, L \cap K_j), \quad c + C_\bullet(L \cap K_i) \mapsto c + C_\bullet(L \cap K_j)$$

for all $i \leq j$. The homology groups of $C_\bullet(K_i, L \cap K_i)$ for $i \in I$ assemble into a pointwise finite-dimensional persistence module denoted by $H_\bullet(-, L) \circ K_I$.

Proof. We first restrict to index pairs of the form $(i-1) \leq i$ and distinguish two cases contingent on σ_i extending K_{i-1} to K_i. If σ_i lies in L then conditions of Theorem 2.21 are met yielding isomorphisms

$$C_\bullet(K_{i-1}, L \cap K_{i-1}) \cong C_\bullet(K_i, L \cap K_i) \quad \text{and} \quad H_\bullet(K_{i-1}, L \cap K_{i-1}) \cong H_\bullet(K_i, L \cap K_i).$$

If $\sigma_i \notin L$ then $L \cap K_{i-1} = L \cap K_i$ and the inclusion is the subspace relation. Since the inclusion corresponding to indices $i \leq j$ is the composition of inclusions corresponding to pairs of the form $(i-1) \leq i$ this argument suffices. It is straightforward to check that the proposed inclusions are chain maps and therefore induce maps in homology that are the structure maps of $H_\bullet(-, L) \circ K_I$. □

To simplify the notation we proceed to write $C_\bullet(K_i, L)$ instead of $C_\bullet(K_i, L \cap K_i)$ even if $L \nsubseteq K_i$. Taking into account the following considerations we may apply the results and methods of Section 2.7 to obtain the barcode decomposition of $H_\bullet(-, L) \circ K_I$.

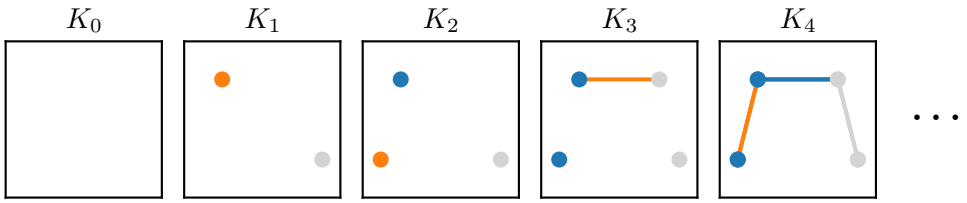


Figure 2.7.: The first five complexes of a condensation of the simplexwise filtration K_I shown in Figure 2.4. Taking K and L from Figure 2.2, simplices newly added to $K \setminus L$ are colored orange while simplices in L are colored gray. After K_4 the filtration continues as in Figure 2.4 with K_8 to K_{15} (here relative to L). As a whole, this condensation is suitable for persistent relative homology since the canonical basis $\mathcal{C}(K, L)$ is filtered in an essential and elementwise manner.

- The filtration K_I needs to be condensed to a filtration such that all isomorphisms of the form $C_\bullet(K_{i-1}, L) \cong C_\bullet(K_i, L)$ that are the result of adding a simplex to L (the first case in the proof of Proposition 2.70) vanish.
- This condensed filtration may no longer be simplexwise. However, at each index of this condensation exactly one basis element is added to $C_\bullet(K_{i-1}, L)$ to form $C_\bullet(K_i, L)$. This is a sufficient replacement for the assumption that the filtration is simplexwise made in the beginning of Section 2.7.
- The conditions for a compatible basis stated in Definition 2.47 can be formulated without conceptual change for $(C_\bullet(K, L), \widetilde{\partial}_\bullet)$ and the results of Subsection 2.7.1 can be transferred to this chain complex its homology.
- The canonical basis $\mathcal{C}(K, L)$ of $C_\bullet(K, L)$ introduced in Definition 2.19 satisfies compatibility with the grading and the condensed filtration. Matrix reduction can thus be applied to the matrix representing the relative boundary map $\widetilde{\partial}_\bullet$ with respect to $\mathcal{C}(K, L)$ in filtration order to obtain a compatible basis of $C_\bullet(K, L)$ as described in Subsection 2.7.3.

Example 2.71. Recall from Example 2.4 the simplicial complex ΔX, in this example denoted by K, and from Figure 2.2 the subcomplex $L \subseteq K$. Figure 2.7 visualizes the first five complexes of a condensation of the filtration shown in Figure 2.4 which filters the canonical basis $\mathcal{C}(K, L)$ in an essential and elementwise manner. By means of this condensation we are able to derive a compatible basis using the methods of Section 2.7.

The rest of this section is concerned with second type of persistent relative homology. Since our main motivation is Theorem 2.73 that establishes a correspondence between the barcode decompositions of $H_\bullet \circ K_I$ and $H_\bullet(K, -) \circ K_I$ we restrict to $L = K$.

Proposition 2.72. The inclusion $C_\bullet(K_i) \hookrightarrow C_\bullet(K_j)$ induces a projection of chain

complexes

$$C_\bullet(K, K_i) \twoheadrightarrow C_\bullet(K, K_j), \qquad c + C_\bullet(K_i) \mapsto c + C_\bullet(K_j)$$

for all $i \leq j$. The homology groups of $C_\bullet(K, K_i)$ for $i \in I$ assemble into a pointwise finite-dimensional persistence module denoted by $H_\bullet(K, -) \circ K_I$.

Proof. Consider the following diagram:

$$\begin{array}{ccccccccc} 0 & \longrightarrow & C_\bullet(K_i) & \xhookrightarrow{u} & C_\bullet(K) & \xrightarrow{p} & C_\bullet(K, K_i) & \longrightarrow & 0 \\ & & \alpha\downarrow & & \| & & \vdots\varphi & & \\ 0 & \longrightarrow & C_\bullet(K_j) & \xhookrightarrow{v} & C_\bullet(K) & \xrightarrow{q} & C_\bullet(K, K_j) & \longrightarrow & 0 \end{array}$$

Note that the left square commutes and $p \circ u$ and $q \circ v$ are the zero morphism from which it follows that $q \circ u = q \circ (v \circ \alpha)$ is likewise zero. Both p and q are thus cokernels of u. By the universal property of the cokernel the dashed map φ exists and the right square commutes, that is $\varphi \circ p = q$. Since q is a epimorphism, so is φ. □

The following theorem shows that the barcode decomposition of $H_\bullet(K, -) \circ K_I$ is determined by the barcode decomposition of $H_\bullet \circ K_I$:

- Any essential interval $[j, \infty)$ in the barcode of $H_\bullet \circ K_I$ corresponds to $[0, j)$ in the barcode of $H_\bullet(K, -) \circ K_I$ with identical generator $s_j \in \mathcal{E}$.
- Any non-essential interval $[\ell, i)$ in the barcode of $H_\bullet \circ K_I$ is identically an interval in the barcode of $H_\bullet(K, -) \circ K_I$. Its generator however is the bounding chain $s_i \in \mathcal{S} \setminus \mathcal{Z}$ of $s_\ell \in \mathcal{B}$ representing a homology class in degree $\dim(s_\ell) + 1$.

Theorem 2.73. Let $\mathcal{S} = \{s_1, \ldots, s_{|K|}\}$ be a compatible basis of $C_\bullet(K)$ such that $\mathcal{S}(K_i) = \mathcal{S}(K_{i-1}) \cup \{s_i\}$ for all non-zero $i \in I$. Let $\mathcal{B} \subseteq \mathcal{Z} \subseteq \mathcal{S}$ denote bases of $B_\bullet(K)$ and $Z_\bullet(K)$ respectively such that $\mathcal{E} = \mathcal{Z} \setminus \mathcal{B}$ is a basis of $H_\bullet(K)$. Then the barcode decomposition of $H_\bullet(K, -) \circ K_I$ is given by

$$H_\bullet(K, -) \circ K_I \cong \bigoplus_{s_j \in \mathcal{E}} V\big[0, j\big) \oplus \bigoplus_{\substack{s_i \in \mathcal{S} \setminus \mathcal{Z} \\ \ell = \mathrm{pivot}(s_i)}} V\big[\ell, i\big).$$

Proof. We refer to [17] that presents a slight variation of this theorem as Proposition 2.4 and provides a proof based on the chain complex decomposition of Corollary 2.55. □

Remark 2.74. Remark 2.56 points out that the barcode of $H_\bullet \circ K_I$ contains the interval $[1, \infty)$ for any simplexwise filtration K_I containing at least one non-empty complex. By the above correspondence, the barcode of $H_\bullet(K, -) \circ K_I$ therefore always contains the interval $[0, 1)$.

2.9. Persistent Cohomology

The contravariant functor $\mathrm{Hom}(-,\mathbb{F})$ maps an inclusion $C_\bullet(K_i) \hookrightarrow C_\bullet(K_j)$ of chain complexes to a projection of cochain complexes $C^\bullet(K_i) \twoheadleftarrow C^\bullet(K_j)$ which in turn induces a map $H^\bullet(K_i) \leftarrow H^\bullet(K_j)$ in cohomology for all indices $i \leq j$. Taking the latter as the structure maps of a persistence module, the order of the index set I needs to be reversed in order to obtain a covariant functor. Such a persistence module is structurally similar to $H_\bullet(K,-) \circ K_I$ as defined in Proposition 2.72 to which the results of Section 2.7 are not easily transferable (as noted in the beginning of Section 2.8). To alleviate this we proceed in fashion of Section 2.8, this time moving to persistent relative cohomology to which the notion of a compatible basis and matrix reduction are applicable without conceptual change.
Thereafter we point out consequences of the vector space duality of homology and cohomology that allow us to derive homology representatives from a computation in the cohomological setting in Subsection 3.5.3. This duality is then used again to prove a dual version of Theorem 2.73, reiterated as a correspondence of persistent relative and absolute cohomology. We thereby gain indirect access to persistent (absolute) cohomology which is central to the current state-of-the-art implementations of persistent homology (reviewed in Section 3.3).
Throughout this section let K denote a finite simplicial complex and K_I an essential simplexwise filtration with index set $I = \{0,\dots,|K|\}$. In the following proposition we define persistent cohomology by means of a persistence module indexed by I with reversed order. This is modeled by passing to the opposite category $\mathbf{I}^{\mathrm{op}}$ of $\mathbf{I}$ that contains the same objects but replaces any morphism $i \to j$ in $\mathbf{I}$ with a morphism $j \to i$ (also denoted by $j \leq^{\mathrm{op}} i$) by formally reversing the arrow (see [26, Construction 1.1.9]). Since $\mathrm{Hom}(-,\mathbb{F})$ can be interpreted as a covariant functor $\mathbf{Vect}_{\mathbb{F}}^{\mathrm{op}} \to \mathbf{Vect}_{\mathbb{F}}$ we get a covariant functor $\mathbf{I}^{\mathrm{op}} \to \mathbf{Vect}_{\mathbb{F}}$ as required for a persistence module.

Proposition 2.75. The assignment

$$H^\bullet \circ K_I :\ \mathbf{I}^{\mathrm{op}} \to \mathbf{Vect}_{\mathbb{F}}, \qquad \left\{ \begin{array}{lcl} i & \mapsto & H^\bullet(K_i), \\ i \leq^{\mathrm{op}} j & \mapsto & H^\bullet(K_i) \to H^\bullet(K_j) \end{array} \right\}$$

is a (covariant) functor and moreover a pointwise finite-dimensional persistence module.

Proof. We define $H^\bullet \circ K_I$ as the composition of covariant functors

$$H^\bullet \ \circ\ \mathrm{Hom}(-,\mathbb{F}) \ \circ\ C_\bullet^{\mathrm{op}} \ \circ\ K_I^{\mathrm{op}} :\ \mathbf{I}^{\mathrm{op}} \to \mathbf{Vect}_{\mathbb{F}}$$

where the superscript 'op' denotes the opposite functor (see [26, Definition 1.2.10]). Since K is assumed to be finite, all occurring vector spaces are finite. □

Definition 2.39 asks for $\mathbf{I}$ instead of $\mathbf{I}^{\mathrm{op}}$ but this is of no practical relevance, especially since $\mathbf{I}^{\mathrm{op}}$ is isomorphic to $\mathbf{I}$ for the index sets relevant in this thesis, namely finite sets

and $\mathbb{R}$. For this reason and in a slight abuse of notation we omit the superscript 'op' from K_I in the composite $H^\bullet \circ K_I$.
With the following Proposition we introduce another $\mathbf{I}^{\mathrm{op}}$-indexed persistence module that is the dual of $H_\bullet(K, -) \circ K_I$.

Proposition 2.76. The inclusion $C_\bullet(K_i) \hookrightarrow C_\bullet(K_j)$ induces an inclusion of relative cochain complexes

$$C^\bullet(K, K_i) \hookrightarrow C^\bullet(K, K_j)$$

for all $i \leq^{\mathrm{op}} j$. The cohomology groups of $C^\bullet(K, K_i)$ assemble into a pointwise finite-dimensional persistence module indexed over $\mathbf{I}^{\mathrm{op}}$ that we denote $H^\bullet(K, -) \circ K_I$.

Proof. By applying $\mathrm{Hom}(-, \mathbb{F})$ to the projection $C_\bullet(K, K_i) \twoheadrightarrow C_\bullet(K, K_j)$ of Proposition 2.72 we obtain the inclusion of relative cochain complexes that induces the structure map of $H^\bullet(K, -) \circ K_I$ for $i \leq^{\mathrm{op}} j$. □

Since the structure maps of $H^\bullet(K, -) \circ K_I$ are induced by inclusions of cochain complexes, we may reformulate the constraints imposed on a compatible basis, making it suitable for persistent relative cohomology.

Definition 2.77. Let $\mathcal{T}$ be a basis of $C^\bullet(K)$. We call $\mathcal{T}$ *compatible with the filtration (of relative cochain complexes derived from K_I)* if $\mathcal{T}$ restricts to a basis $\mathcal{T}(K_i)$ of $C^\bullet(K, K_i)$ for all $i \in I$. From Definition 2.47 we transfer the notion of compatibility with the boundary map to the cohomological setting and without change the compatibility with the grading. If $\mathcal{T}$ is compatible in all three respects it is called *compatible*.

With this notion of a compatible basis for $C^\bullet(K)$, the results of Section 2.7 can be transferred to the cohomological setting taking into account the following considerations.

- Since $H^\bullet(K, -) \circ K_I$ is indexed by $\mathbf{I}^{\mathrm{op}}$, inclusions of bases are reversed with respect to the indexing. An example is $\mathcal{T}(K_i) \subseteq \mathcal{T}(K_{i-1})$ for all non-zero $i \in I$. In other words, $\mathcal{T}$ as a filtered basis extends as the filtration index decreases.
- Another consequence of the reversed indexing is that essential intervals persist until $i = 0$ (the maximum element of I with reversed order). We continue to write intervals as usual however, endpoints ordered with respect to the usual order on I.
- The coboundary map $\partial^\bullet$ of $C^\bullet(K) = C^\bullet(K, K_0)$ restricts to the coboundary map of $C^\bullet(K, K_i)$ for all $i \in I$. We represent it as the matrix $D^\perp$, the *coboundary matrix*, with respect to the canonical basis $\mathcal{C}^*(K)$ in reverse filtration order. Since $\mathcal{C}^*(K)$ is compatible with the grading and the filtration, the results of Subsection 2.7.3 applied to $D^\perp$ yield a compatible basis of $H^\bullet(K, -) \circ K_I$ and thereby its barcode decomposition.

Corollary 2.78. Let $\mathcal{T} = \{t_1, \ldots, t_{|K|}\}$ be a compatible basis of $C^\bullet(K)$ such that $\mathcal{T}(K_{i-1}) = \mathcal{T}(K_i) \cup \{t_i\}$ for all non-zero $i \in I$. Let $\mathcal{B} \subseteq \mathcal{Z} \subseteq \mathcal{S}$ denote bases of $B^\bullet(K)$

and $Z^\bullet(K)$ respectively such that $\mathcal{E} = \mathcal{Z} \setminus \mathcal{B}$ is a basis of $H^\bullet(K)$. Then the barcode decomposition of $H^\bullet(K,-) \circ K_I$ is given by

$$H^\bullet(K,-) \circ K_I \cong \bigoplus_{t_j \in \mathcal{E}} V\big[0,j\big) \oplus \bigoplus_{\substack{t_\ell \in \mathcal{B} \\ \ell = \mathrm{pivot}(t_i)}} V\big[i,\ell\big). \qquad \square$$

We return to $H^\bullet \circ K_I$. Recall Theorem 2.29 which asserts that cohomology is naturally isomorphic to the dual of homology, that is $\mathrm{Hom}(H_\bullet(K), \mathbb{F}) \cong H^\bullet(K)$. In this instance, naturality means that for any $i \leq j$ the following diagram commutes:

$$\begin{array}{ccc} \mathrm{Hom}(H_\bullet(C_\bullet(K_j)), \mathbb{F}) & \xrightarrow{\cong} & H^\bullet(C^\bullet(K_j)) \\ \downarrow{\scriptstyle \phi_{ij}^*} & & \downarrow{\scriptstyle \phi^{ji}} \\ \mathrm{Hom}(H_\bullet(C_\bullet(K_i)), \mathbb{F}) & \xrightarrow{\cong} & H^\bullet(C^\bullet(K_i)) \end{array}$$

Here ϕ_{ij}^* is the dual map of $\phi_{ij} : H_\bullet(K_i) \to H_\bullet(K_j)$ that is the structure map of $H_\bullet \circ K_I$ induced by $i \leq j$. ϕ^{ji} on the other hand denotes the structure map of $H^\bullet \circ K_I$ that is the image of $j \leq^{\mathrm{op}} i$. The underlying chain map ($f_\bullet$ in Theorem 2.29) is the inclusion of chain complexes $C_\bullet(K_i) \hookrightarrow C_\bullet(K_j)$.
A second isomorphism is given by vector space duality, that is $\mathrm{Hom}(H_\bullet(K), \mathbb{F}) \cong H_\bullet(K)$ (see Proposition 2.22). Together these two isomorphisms imply that the rank of ϕ_{ij} and ϕ^{ji} is equal for all $i \leq j$. This further implies that $H_\bullet \circ K_I$ and $H^\bullet \circ K_I$ have the same barcode since the barcode is an isomorphism invariant (see Theorem 2.45).

Corollary 2.79. The barcodes of $H_\bullet \circ K_I$ and $H^\bullet \circ K_I$ are identical. Similarly, the barcodes of $H_\bullet(K,-) \circ K_I$ and $H^\bullet(K,-) \circ K_I$ are identical. $\square$

Vector space duality also yields an isomorphism of chain groups $C_p(K) \cong C^p(K)$ for all $p \in \mathbb{Z}$ which is well-behaved with respect to the chain complex decomposition of Corollary 2.55 as the following proposition shows.

Proposition 2.80. Let $\mathcal{S} = \{s_1, \ldots, s_{|K|}\}$ be a compatible basis of $C_\bullet(K)$ and $\mathcal{S}^* = \{s_1^*, \ldots, s_{|K|}^*\}$ the dual basis. Then the cochain complex $C^\bullet(K)$ is the direct sum of cochains

$$\cdots \longleftarrow 0 \longleftarrow s_j^* \mathbb{F} \longleftarrow 0 \longleftarrow 0 \longleftarrow \cdots$$

$$\cdots \longleftarrow 0 \longleftarrow s_i^* \mathbb{F} \xleftarrow{\partial_\bullet^*} s_\ell^* \mathbb{F} \longleftarrow 0 \longleftarrow \cdots$$

taken over all essential chains s_j and non-essential pairs (s_ℓ, s_i).

Proof. This follows straightforwardly from the definition of $\partial_\bullet^*$ and the properties of a compatible basis. We also refer to Proposition 2.10 in [17] and the proof thereof. $\square$

Proposition 2.80 suggests that the dual basis $\mathcal{S}^*$ of a compatible basis $\mathcal{S}$ is compatible with the coboundary map. Indeed, the dual of a compatible basis is itself compatible as the following proposition shows.

Proposition 2.81. Let $\mathcal{S}$ be a compatile basis of $C_\bullet(K)$. Then the dual basis $\mathcal{S}^*$ is a compatible basis of $C^\bullet(K)$.

Proof. Since $\mathrm{Hom}(-,\mathbb{F})$ preserves direct sums, compatibility with the grading is preserved for the dual basis. By the assumption that $\mathcal{S}$ is compatible with respect to the filtration, it follows that $\mathcal{S}^*$ likewise is: For any $i \in I$ the complement of the restriction $\mathcal{S}(K_i)$ is a basis of relative chains $C_\bullet(K, K_i)$ that define a restriction of $\mathcal{S}^*$ to $C^\bullet(K, K_i)$ by dualization. Compatibility with the coboundary map follows from Proposition 2.80. □

Corollary 2.82. Let $\mathcal{T}$ be a compatible basis of $C^\bullet(K)$. Then $\mathcal{T}^*$ is a compatible basis of $C_\bullet(K)$ under the identification of $C_\bullet(K)$ and its double dual. □

Proposition 2.83. In the notation of Corollary 2.78, the barcode decomposition of $H^\bullet \circ K_I$ is given by

$$H^\bullet \circ K_I \;\cong\; \bigoplus_{t_j \in \mathcal{E}} V\big[j, \infty\big) \;\oplus \bigoplus_{\substack{t_i \in \mathcal{T} \setminus \mathcal{Z} \\ \ell = \mathrm{pivot}(t_i)}} V\big[i, \ell\,\big).$$

Proof. This is a consequence of Theorem 2.73 and the vector space duality of persistent (absolute) homology and cohomology. For more details we refer to [17, Section 2.7]. □

We conclude this section with an application of Proposition 2.83 to our recurring example of the full simplex ΔX, in the following Example denoted by K, on the four point space X that is introduced in Example 2.4.

Example 2.84. Figure 2.8 visualizes a set of generators of the barcode decomposition of $H^\bullet \circ K_I$ where K_I is the reverse of the simplexwise filtration shown in Figure 2.4. Using Proposition 2.83 these generators are derived from the barcode decomposition of $H^\bullet(K, -) \circ K_I$ that was computed by reducing the coboundary matrix $D^\perp$.

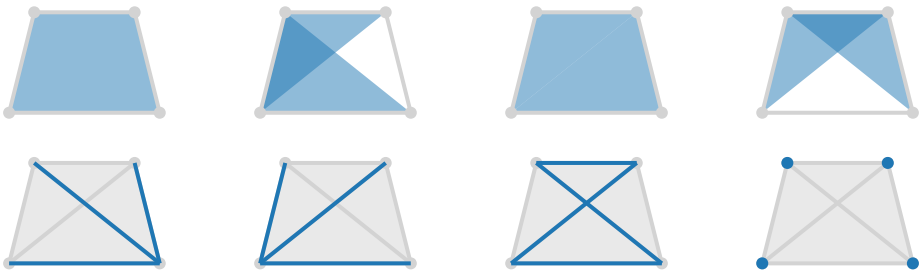


Figure 2.8.: A set of generators for the interval modules of the barcode decomposition of $H^\bullet \circ K_I$ where K_I is the filtration shown in Figure 2.4. The visualization as chains in $C_\bullet(K)$ uses the isomorphism $\mathrm{Hom}(C_\bullet(K), \mathbb{F}) \cong C_\bullet(K)$ defined by means of the respective the canonical basis, mapping by $\sigma^* \mapsto \sigma$.

3. Computing Persistent Homology

The previous chapter introduced the persistent (relative) (co)homology of a filtration of a finite simplicial complex as the barcode decomposition of the corresponding persistence module. This chapter examines various practical aspects of the computation of this decomposition. We start with a precise problem statement that defines what we mean by 'computing persistent homology'.

Problem 3.1. Let K_I be a filtration of a finite simplicial complex K. Furthermore, let M be one of the following persistence modules

$$H_\bullet \circ K_I, \qquad H_\bullet(K,-) \circ K_I, \qquad H^\bullet \circ K_I, \qquad H^\bullet(K,-) \circ K_I,$$

and let $\mathbb{F}$ be a field. Parametrized by M and $\mathbb{F}$, the *persistent homology* of K_I is the computational problem specified as

Input:	K,	a finite simplicial complex,
	K_I,	a filtration of K with index set I.
Output:	$\mathcal{J}$,	the barcode of the persistence module M,
	$\mathcal{E}$,	the interval module generators for all essential indices,
	$\mathcal{B}$,	the interval module generators for all non-essential indices.

Remark 3.2. In Problem 3.1 we write 'parameterized' to convey that one particular choice for M and $\mathbb{F}$ constitutes one instance of Problem 3.1. M determines which type of homology is to be computed and is neither dependent on the input nor part of it.

We briefly summarize the results of Sections 2.7, 2.8 and 2.9 that instruct on how to solve this problem by matrix reduction if the filtration K_I is essential and simplexwise.
First the boundary matrix is constructed by representing the boundary map $\partial_\bullet$ with respect to the canonical basis $\mathcal{C}(K)$ in filtration order. Then this matrix is reduced to obtain a matrix decomposition $R = DV$. From the columns of R and V the barcode and interval module generators can be read off. In the cohomological setting we instead work with the coboundary matrix $D^\perp$ representing the coboundary map $\partial^\bullet$ with respect to $\mathcal{C}^*(K)$ in reverse filtration order.
Asking for an essential simplexwise filtration is too restrictive however, in particular since many applications work with sublevel sets of real-valued functions, such as the Vietoris-Rips filtration introduced in Definition 2.35. This is addressed in Section 3.1 where we describe how to construct an essential simplexwise refinement of the input filtration such that our desired output can be derived from the matrix reduction in the simplexwise setting. We consider this process to be an input-output transformation that lies on top of the actual reduction algorithm, formulated in Section 3.2, which

operates on the (co)boundary matrix directly. In Section 3.3 we then proceed to review a number of optimizations that improve the computational efficiency of the reduction algorithm, in large parts following [2]. Going beyond algorithmic optimizations, Section 3.4 investigates the computation of persistent relative homology as an alternative to persistent (absolute) homology promising better computational performance at the cost of topological insights. We conclude this chapter with Section 3.5 that compares three approaches to the computation of homology representatives including a new approach that utilizes the duality results of Section 2.9 in an attempt to streamline the current state-of-the-art approach.

3.1. Computational Setup

The first part of this section describes the input-output transformation that constructs an essential simplexwise filtration based on the input filtration to which we can apply matrix reduction. This filtration is constructed in a way that allows us to retranslate the output of the matrix reduction to meet the output specification of Problem 3.1 with respect to the input filtration.
The input-output transformation is constructed in two steps. First all duplicate complexes in the filtration have to be removed by condensing it to an essential filtration with a finite index set. Then follows the refinement to a simplexwise filtration. For both steps we analyze how the barcode decomposition changes in order to recover our desired output starting with the following proposition.

Proposition 3.3. Let K_I be a filtration and $c : I \to I$ be a map that for all $i \in I$ maps the set $A_i = \{j \in I \mid K_i = K_j\}$ to $\inf(A_i)$ if it is attained and an arbitrary index within A_i otherwise. Then c is a reindexing map that condenses K_I to an essential filtration K_J with index set $J = c(I)$.
The barcodes for K_I and K_J are identical under the following identifications:

- If I has no upper bound, intervals of the form $[i, \sup(I)) \in \mathcal{I}$ are identified with a corresponding interval $[i, \sup(J)] \in \mathcal{J}$ each.
- If I has no lower bound, intervals of the form $(\inf(I), j) \in \mathcal{I}$ are identified with a corresponding interval $[\inf(J), j) \in \mathcal{J}$ each and intervals of the form $(\inf(I), \sup(I)) \subseteq \mathcal{I}$ with a corresponding interval $[\inf(I), \sup(J)] \subseteq \mathcal{J}$.

Under this identification of barcodes, the corresponding interval module generators for K_I and K_J are identical.

Proof. By definition the two sets A_i and A_j are either equal or disjoint and c is thus a well-defined monotonously increasing map. It is a condensation since it maps surjectively onto its image J. K_J is essential since $c(i) = c(j)$ if and only if $K_i = K_j$.
If $K_i = K_j$ then $C_\bullet(K_i) \to C_\bullet(K_j)$ is the identity map which is preserved by any of the

four homology functors. From this it is clear that the intervals of the two barcodes must be identical except possibly at the bounds where I gets truncated by c. If there exist an $i > \sup(J)$ it is mapped to $\sup(J) = \inf(A_{\sup(I)})$ and if I has no lower bound $\inf(I)$ is mapped to some index in $A_{\inf(I)}$. □

Remark 3.4. In the notation of Remark 2.42 the edge cases of Proposition 3.3 correspond to an identification of $[a, \infty)$ with $[a, \infty)$, $(-\infty, a)$ with $(-\infty, a)$ and $(-\infty, \infty)$ with $(-\infty, \infty)$, which justifies calling the barcodes identical.

In Example 2.37 the condensation of Proposition 3.3 is constructed explicitly to condense a Vietoris-Rips filtration to an essential filtration with finite index set, choosing -1 as image of the unbounded interval $(-\infty, 0) \subseteq \mathbb{R}$.
Once an essential filtration is obtained from the input filtration, we need to construct a simplexwise refinement of it. In the following we outline an approach adapted from [2] that is based on the colexicographic order using $\mathrm{vert}(K)$ with a prescribed total order as alphabet and the thereby determined ordered simplices as words.

Definition 3.5. Let $v_1 < \cdots < v_n$ be a total order on the set of vertices $\mathrm{vert}(K)$ of a finite simplicial complex K. This order induces a lexicographic order on the set of all simplices in K as follows.
Let σ and τ be two distinct p-simplices with vertices $v_{i_0} < \ldots < v_{i_p}$ and $v_{j_0} < \ldots <_{j_p}$ respectively and let $k \in \{0, \ldots, p\}$ be the largest index such that $v_{i_k} \neq v_{j_k}$. Then we set $\sigma < \tau$ if and only if $v_{i_k} < v_{j_k}$. The resulting total order on the set of p-simplices is referred to as the *colexicographic* order.

The use of the colexicographic order instead of more intuitive lexicographic order (that uses the smallest index k such that $v_{i_k} < v_{j_k}$) is motivated by the following remark.

Remark 3.6. For the colexicographic order on the set of p-simplices of the full simplex ΔX on a finite set X there exists an order-preserving bijection

$$\{\sigma \in K \mid \dim(\sigma) = p\} \;\to\; \left\{0, \ldots, \binom{n}{p+1} - 1\right\} \quad \text{by} \quad (v_{i_p}, \ldots, v_{i_0}) \;\mapsto\; \sum_{k=0}^{p} \binom{i_k}{k+1}.$$

This bijection is used in the software Ripser to represent simplices and enumerate (co)facets of a given simplex. For more details and examples see [2].

The following proposition defines on the set of all simplices in K a total order that respects the order imposed by the input filtration and refines an essential condensation thereof to a simplexwise filtration.

Proposition 3.7. Let K_I be an essential filtration and fix a total order on $\mathrm{vert}(K)$. For two distinct simplices σ and τ we define a total order on all simplices by setting $\sigma < \tau$ if

1. (filtration parameter) there exists $i \in I$ such that $\sigma \in K_i$ and $\tau \notin K_i$ or otherwise

2. (dimension) $\dim(\sigma) < \dim(\tau)$ or if $p = \dim(\sigma) = \dim(\tau)$
3. (reverse colexicographic order) $\sigma > \tau$ with respect to the colexicographic order.

For $j \in \{0, \dots, |K|\}$ a simplexwise filtration K_J is defined by setting $K_0 = \emptyset$ and $K_j = K_{j-1} \cup \{\sigma_j\}$ for all non-zero $j \in J$ such that σ_j denotes the j-th simplex with respect to the above order. Mapping $i \in I$ to the largest index $j \in J$ such that $\sigma_j \in K_i$ and $\sigma_{j+1} \notin K_i$ is a reindexing $r : I \to J$ refining K_I to K_J, called its *lexicographic refinement*.

Proof. It is clear that the three conditions define a total order on the set of simplices in K. Condition two ensures that all faces of a simplex σ are added before σ and K_J is thus a simplexwise filtration by construction. The first condition implies that r is monotonic and thus a reindexing map. To verify that r is injective and thus a refinement, note that i is mapped to the index j of the youngest simplex in K_J such that σ_j still lies in K_i. Since K_I is assumed to be essential this simplex is unique for every $i \in I$. □

Note that the reverse colexicographic order is generally not equal to the lexicographic order. The reversal of the colexicographic order in condition three of Proposition 3.7 is motivated by certain algorithmic advantages compared to the colexicographic order in connection with the enumeration of facets and cofacets using the combinatorial numbering system of Remark 3.6 as pointed out in [2, Section 4]. Disregarding such considerations, any order on the set of p-simplices may be used instead of the reverse colexicographic order.
Applied to a Vietoris-Rips filtration the above order on the set of simplices in K translates to an order first by diameter, then by dimension and then by the reverse colexicographic order.
We want to emphasize that the lexicographic refinement depends on a choice of order for $\mathrm{vert}(K)$. Although Proposition 3.9 shows that this choice does not affect the specified output of the computational problem, the particular cycle representing the interval module generator as basis element in $\widetilde{\mathcal{Z}}$ does depend on it.

Example 3.8. We return to the full simplex ΔX of Example 2.4 and define a total order on $\mathrm{vert}(\Delta X)$ by

$$(2,2) \;<\; (-3,-2) \;<\; (-2,2) \;<\; (3,-2).$$

in the following denoted by $v_1 < \cdots < v_4$. With respect to this vertex order, the simplexwise filtration that is defined in Example 2.37 and visualized in Figure 2.4 is the lexicographic refinement of the condensation of the Vietoris-Rips filtration of ΔX that is defined in the same example and shown in Figure 2.3. The reverse colexicographic order determines the filtration whenever two simplices of the same dimension and diameter are added. Notably, with respect to the reverse colexicographic refinement the vertices are ordered $v_4 < \cdots < v_1$. The other instances where condition three of Proposition 3.7 applies are $\{v_1, v_4\} < \{v_2, v_3\}$ and $\{v_2, v_3, v_4\} < \{v_1, v_3, v_4\} < \{v_1, v_2, v_4\} < \{v_1, v_2, v_3\}$ at indices $i \in \{6, 7\}$ and $i \in \{11, 12, 13, 14\}$, respectively.

Figure 3.1.: The barcode of $H_\bullet \circ K_I$ where K_I is the Vietoris-Rips filtration of ΔX, the full simplex on the four point space introduced in Example 2.4 and chain groups are taken over $\mathbb{F}_2$. The bottom-most bar visualizes the unbounded interval $[0, \infty) \subseteq \mathbb{R}$. The persistent homology for $p \geq 2$ is trivial. This barcode is derived from the barcode shown in Figure 2.5 by using Proposition 3.3 and 3.9.

Similar to Proposition 3.3 the following proposition examines how the barcode decomposition for an essential filtration can be recovered from the barcode decomposition for its lexicographic refinement following [2, Proposition 2.1].

Proposition 3.9. Let K_I be an essential filtration and K_J the lexicographic refinement of K_I with corresponding reindexing map $r : I \to J$. Denote by $\mathcal{I}$ and $\mathcal{J}$ the barcodes of K_I and K_J respectively. Then $\mathcal{I}$ is equal to the (multi)set of those non-empty preimages $r^{-1}[i, j)$ for $[i, j) \in \mathcal{J}$ that have positive (non-zero) length.
Intervals in $\mathcal{J}$ that have zero length under the preimage r^{-1} are called *zero persistence* intervals. Those interval module generators with respect to K_J that have positive length under r^{-1} identically generate those with respect to K_I.

Proof. We refer to Proposition 2.1 in [2] and the proof thereof. □

Remark 3.10. The zero persistence intervals of a Vietoris-Rips filtration are precisely those intervals for which birth and death simplex have equal diameter.

In summary, we have shown that the lexicographic refinement (after essential condensation) as an input transformation is well-behaved with respect to the original filtration as the output of Problem 3.1 may be derived from the output corresponding to the simplexwise refinement using Proposition 3.3 and 3.9. With the following example we illustrate this by relating the barcode decomposition for the Vietoris-Rips filtration of ΔX as defined in Example 2.4 with that for its simplexwise refinement.

Example 3.11. Recall from Example 2.4 the full simplex ΔX and from Example 2.37 the essential condensation of the Vietoris-Rips filtration of ΔX as well as its subsequent lexicographic refinement (visualized in Figure 2.4). Computing the barcode decomposition for this lexicographic refinement yields the barcode shown in Figure 2.5. In order to obtain the barcode for the initial Vietoris-Rips filtration, shown in Figure 3.1 we apply Proposition 3.3 and 3.9. The key points to this derivation are the following.

In degree zero there are no intervals of zero persistence. This is due to the fact that all 0-simplices have diameter zero whereas all 1-simplex have non-zero diameter. In particular, all persistence intervals associated to the lexicographic refinement are a superset of the interval $[4,5) \subseteq \{0,\dots,15\}$ with preimage $[0,4) \subseteq \{-1,0,4,\sqrt{17},6,\sqrt{41}\}$ under the lexicographic reindexing map (which is spelled out in Example 2.37). The single essential interval $[1,\infty)$ has preimage $[0,\infty)$ where ∞ denotes the supremum of $\{0,\dots,15\}$ (using the shorthand of Remark 2.42) and the index set of the above condensation respectively. In the latter case it is attained with value $\sqrt{41}$ and is identified with the supremum of $\mathbb{R}$ yielding the sole essential interval $[0,\infty) \subseteq \mathbb{R}$ in the barcode corresponding to the Vietoris-Rips filtration of ΔX.
In degree one there are two zero persistence intervals, that is $[9,11)$ and $[10,13)$. For both intervals birth and death simplex have identical diameter and their length under the preimage of the lexicographic reindexing map is therefore zero. The same applies for the sole interval $[14,15)$ in degree two and the persistent homology of the Vietoris-Rips filtration of ΔX is thus trivial for degrees $p \geq 2$.

3.2. The Reduction Algorithm

The *reduction algorithm* formalizes the results of Sections 2.7 to 2.9 solving the following problem.

Problem 3.12. Let M be one of the four persistence module listed in Problem 3.1 and let K_I be an essential and simplexwise filtration. Parametrized by M and a field $\mathbb{F}$ we formulate the problem *persistent homology with respect to a simplexwise filtration* K_I as

Input:	D, the (co)boundary matrix with respect to the simplexwise K_I.
Output:	$\mathcal{J}$, the barcode of the persistence module M,
	$\mathcal{E}$, the interval module generators for all essential indices,
	$\mathcal{B}$, the interval module generators for all non-essential indices.

Whether the expected input is the boundary or coboundary matrix dependens on M.

The previous section describes how Problem 3.1 is solved by solving the above computational problem. Problem 3.12 for the parameter instance $M = H_\bullet \circ K_I$ (and any choice of field) is solved by Algorithm 3.1 that implements the results of Section 2.7 and is commonly called the *reduction algorithm*. The sole input is the boundary matrix D as specified in the above problem statement. In Lines 1 and 2 the output sets are assigned the empty set and the matrices R and V are initialized as in the proof of Proposition 2.60. Line 3 specifies the iteration over the columns of D from left to right. Note that although $|K|$ is not specified as an input of Problem 3.12 we use it here to concisely denote the number of columns of D. Lines 4–8 carry out one iteration of the matrix reduction, reducing the j-th column as described in the aforementioned proof. Lines 9–16 implement

the two cases of Lemma 2.51. If the current column R_j was reduced to the zero column and the condition in Line 10 is satisfied, a new generator V_j and its associated interval $[j, \infty)$ are added as a preliminary essential interval summand, corresponding to the first case of Lemma 2.51. Otherwise the second case applies and the previously unbounded V_ℓ becomes bounded by V_j. Consequently it is replaced by the boundary B_j and its interval is adjusted to end at index j.
After Algorithm 3.1 terminates $\mathcal{E}$ and $\mathcal{B}$ contain precisely those columns that Proposition 2.67 and 2.68 specify proving it to be correct.

Algorithm 3.1: The reduction algorithm

```
1  J := ∅;  E := ∅;  B := ∅
2  R := D;  V := id
3  for j := 1 to |K| do
4      R_j := D_j
5      while there exists k < j such that ℓ := pivot(R_k) = pivot(R_j) do
6          λ := R_{ℓj} / R_{ℓk}
7          R_j := R_j − λR_k
8          V_j := V_j − λV_k
9      ℓ := pivot(R_j)
10     if ℓ = 0 then
11         J := J ∪ [j, ∞)
12         E := E ∪ V_j
13     else
14         J := J ∪ [ℓ, j) \ [ℓ, ∞)
15         B := B ∪ R_j
16         E := E \ V_ℓ
```

Remark 3.13. It is possible to restructure Algorithm 3.1 in order to the separate matrix reduction from the output assembly, moving the latter out of the reduction loop and avoiding the update of Lines 14–16: Once $R = DV$ has been computed the compatible basis and thereby the barcode can simply be read off as shown in Proposition 2.67 and 2.68. This alternate option may be slightly more concise but precludes the optimization of Subsection 3.3.1 which omits the assembly of R, discarding the column R_j after the j-th iteration.

The rest of this section is concerned with the adaptation of Algorithm 3.1 to solve Problem 3.12 for the other three instances (depending on M). Recall from Section 2.9 we specify any interval $J \subseteq I = \{0, \dots, |K|\}$ with respect to the usual order on I even if the persistence module is $\mathbf{I}^{\mathrm{op}}$-indexed.

- (Relative homology) Theorem 2.73 instructs on how Algorithm 3.1 has to be changed to output the barcode decomposition of $H_\bullet(K, -) \circ K_I$ when provided with the

boundary matrix D. Essential intervals are now of the form $[0, j)$ and Line 11 has to be changed to $\mathcal{J} := \mathcal{J} \cup [0, j)$. The corresponding generators remain unchanged however. On the other hand, the generators for non-essential intervals are now given by the bounding chain V_j, replacing the boundary $R_j = DV_j$ while the associated interval update stays identical. We therefore adjust Line 15 to be $\mathcal{B} := \mathcal{B} \cup V_j$.

- (Relative cohomology) Algorithm 3.1 applied to the coboundary matrix $D^\perp$ computes the barcode decomposition of $H^\bullet(K, -) \circ K_I$ as described in Section 2.9. Since persistent relative cohomology develops in reverse filtration order and the usual order on I is inverted we have to adjust the interval bounds. In particular supremum of I is now 0 instead of $|K|$ (in Algorithm 3.1 denoted by ∞ as per Remark 2.42). Moreover, the iteration index j of Line 3 increases as the filtration index of K_I decreases since j indexes columns in $D^\perp$ that are ordered in reverse filtration order. Consequently, the iteration index j corresponds to the filtration index $m - j$ where $m = |K| + 1$. We therefore adjust Line 11 to read $\mathcal{J} := \mathcal{J} \cup [0, m - j)$ and similarly translate the addition of non-essential intervals in Line 14 to be $\mathcal{J} := \mathcal{J} \cup [m - j, m - \ell) \setminus [0, j)$.
- (Absolute cohomology) In order to adapt Algorithm 3.1 to compute the barcode decomposition of $H^\bullet \circ K_I$ the changes of the previous two instances need to be combined: Bounding chains are the generators of non-essential intervals (as opposed to boundaries) and all indices need to be translated to accommodate the reversed filtration order. We thus replace Line 11 by $\mathcal{J} := \mathcal{J} \cup [m - j, \infty)$, Line 14 with $\mathcal{J} := \mathcal{J} \cup [m - j, m - \ell)$ and Line 15 with $\mathcal{B} := \mathcal{B} \cup V_j$ (where $m = |K| + 1$). This precisely matches Proposition 2.83 after taking into account the translation of the iteration index to the filtration index.

3.3. Efficient Computation and Optimizations

With the goal of improving the computational performance of Algorithm 3.1 we review five optimizations in this section. Together they contribute to the current state-of-the-art implementation of the reduction algorithm in the software Ripser [2] and giotto-ph [31]. First however, we introduce a variation of the reduction algorithm that makes possible a degreewise computation. Instead of operating on the boundary matrix D representing $\partial_\bullet : C_\bullet(K) \to C_\bullet(K)$ the matrices representing $\partial_p : C_p(K) \to C_{p-1}(K)$ are reduced independently of one another. This reduces the amount of memory required at any one time since the data structures associated with the reduction in degree p can mostly be discarded when moving on to the next degree. Another important practical advantage of a degreewise reduction is the increased level of control over the output. For most applications only small number of degrees, often $0 \leq p \ll \dim(K)$, are of interest. Performing a degreewise computation allows us to straightforwardly select the degrees for which the barcode decomposition is to be computed.

The following proposition shows that this approach solves Problem 3.12 equally well if a permutation of rows and columns is taken into account. As before, we restrict to essential simplexwise filtrations. Recall that this comes at no loss of generality due to the input-output transformation described in Section 3.1.

Proposition 3.14. Let K_I be an essential simplexwise filtration. We denote by $\mathcal{C}^{\downarrow}(K)$ the canonical basis of $C_\bullet(K)$ ordered first by decreasing dimension and within a dimension by the filtration order of K_I. With respect to $\mathcal{C}^{\downarrow}(K)$ we represent the $\partial_\bullet$ as the matrix

$$D^{\downarrow} = \begin{pmatrix} 0 & & & \\ D^p & 0 & & \\ & \ddots & \ddots & \\ & & D^1 & 0 \end{pmatrix}$$

where D^p is the *p-th boundary matrix* representing ∂_p with respect to $\mathcal{C}_p(K)$ in filtration order. Then the decompositions $D = RV$ and $D^{\downarrow} = R^{\downarrow}V^{\downarrow}$ as computed by matrix reduction (Lines 3–8 of Algorithm 3.1) for the respective input D and $D^{\downarrow}$ are related by

$$R = PR^{\downarrow}P, \qquad V = PV^{\downarrow}P$$

where the permutation matrix P is the change of basis transformation $\mathcal{C}^{\downarrow}(K) \to \mathcal{C}(K)$.

Proof. The sets $\operatorname{pivots}(D^p)\setminus\{0\}$ are pairwise disjoint for all $p \in \mathbb{Z}$ since $\mathcal{C}(K)$ is compatible with respect to the grading by Proposition 2.48 and the boundary of any basis element in $\mathcal{C}_p(K)$ thus lies entirely in $C_{p-1}(K)$. Any column associated with a p-simplex is therefore reduced by using columns corresponding to other p-simplices exclusively. Matrix reduction is thus invariant under a reordering of columns and rows as long as the order within one dimension remains unchanged and the corresponding change of basis transformation P is applied. □

Corollary 3.15. The application of Algorithm 3.1 to $D^{\downarrow}$ instead of D produces the same output after all elements in $\mathcal{B}$ and $\mathcal{E}$ are multiplied by the permutation matrix P as defined in Proposition 3.14 and the endpoints of all intervals in the barcode $\mathcal{J}$ are permuted according to P. □

Applying the (transposed) permutation matrix P to the input and output of Algorithm 3.1 can be seen as another input-output transformation layered on top of the reduction algorithm, similar to the input-output transformation that connects Problem 3.1 with Problem 3.12.

Remark 3.16. Proposition 3.14 and Corollary 3.15 apply likewise in the cohomological setting. We define $\mathcal{C}^{\uparrow}(K)$ to be the canonical basis $\mathcal{C}^*(K)$ of $C^\bullet(K)$ ordered first by increasing dimension and then by reverse filtration order. With respect to $\mathcal{C}^{\uparrow}(K)$ we represent $\partial^\bullet$ as the matrix $D^{\uparrow}$ replacing $D^{\perp}$ in order to compute the degreewise reduction.

Remark 3.17. The choice of ordering by decreasing and increasing dimension defining $\mathcal{C}^{\downarrow}(K)$ and $\mathcal{C}^{\uparrow}(K)$, respectively, is motivated by the optimization of Subsection 3.3.2 where we omit certain columns in a dimension-sensitive fashion.

Ripser implements the reduction algorithm in a degreewise fashion to solve Problem 3.1 for the instance $M = H^\bullet \circ K_I$ and a prime field $\mathbb{F}_p$. It restricts the input of Problem 3.1 to Vietoris-Rips filtrations that may be cut off at specified diameter threshold $t > 0$ and outputs only the barcode $\mathcal{J}$ of $H^\bullet \circ K_I$ (equivalently that of $H_\bullet \circ K_I$ by Corollary 2.79), discarding any representatives. The input filtration is refined by colexicographic refinement and matrix reduction is implemented for the coboundary matrices $(D^\perp)^p$ as described above, starting in degree zero and proceeding up to a maximum specified degree.
In each of the next five subsections we review one optimization of Algorithm 3.1. The first four are described in [2] and are implemented as part of Ripser. For each we give a short summary of the specifics concerning Ripser's implementation. The fifth optimization comprises a parallelization of the reduction algorithm that was first introduced in [27] and is implemented as part of giotto-ph. In the final two subsections we briefly summarize all reviewed optimizations in the context of Ripser, highlight examples of recent and ongoing research and end on a short description of the data structures used in Ripser in preparation for the performance analysis in later sections.

3.3.1. Implicit Matrix Representation

The amount of memory required to maintain the matrices that are part of the iteratively constructed decomposition $R = DV$ starts to become problematic for moderately large complexes even if a degreewise reduction is used and only non-zero entries are stored. In this subsection we review a method presented in [2] that involves implicitly representing the matrices R and D thereby reducing the memory footprint of the reduction algorithm at the cost of additional computational effort. The same techniques can be applied in the cohomological setting without conceptual change.

Implicit Representation of R. At iteration index j during the reduction loop the column R_j (initialized as D_j) is to be reduced which requires access to the previously reduced columns R_i for $i < j$. Instead of retrieving any such R_i from the matrix R stored in memory, the column may be recomputed by evaluating the matrix-vector product $R_i = DV_i$. For the purpose of matrix reduction, this recomputation may replace all accesses to a stored R rendering the storage of R superfluous. R_j may thus be discarded immediately after the j-th iteration finishes. However, columns of R are also used as part of the output of Algorithm 3.1. Only by removing $\mathcal{B}$ from the output specification is it possible to eliminate R from the memory footprint entirely. For many applications the omission of of interval module generators has no downside because only the barcode $\mathcal{J}$ is of interest. Even if $\mathcal{B}$ is needed we may identify and omit all representatives corresponding to zero-persistence intervals that often are a substantial fraction of all non-essential pairs [2].

Algorithm 3.2 implements the implicit matrix representation of R together with clearing, an optimization introduced in the following subsection (a version without clearing can be found in the appendix). This modified reduction algorithm introduces the iteratively updated map pivot-column-index that assigns to a pivot index ℓ the column index j of R such that $\text{pivot}(R_j) = \ell$ or indicates that no such column exists by returning the value -1. In Line 3 this map is initialized for all potential pivot indices with value -1. It is updated in Line 16 whenever the reduction of a column has finished with non-zero pivot index. Since the non-zero pivot indices of a reduced matrix are unique, all encountered non-zero pivot indices have a unique assigned column index at any one point during the reduction. Lines 9–17 replace the reduction loop of Lines 5–8 of Algorithm 3.1. By keeping track of the encountered pivot indices via pivot-column-index, the index of the next column required to further reduce the current column R_j (if possible) is given by $i = \text{pivot-column-index}(\ell)$ (if it is not -1) where ℓ denotes the current pivot index of R_j. This makes explicit Line 5 of Algorithm 3.1. In Line 10 of Algorithm 3.2, the column R_i is then reconstructed from D and V_i and subsequently is added to R_j in Line 12 after which the current pivot index ℓ of R_j is updated and the reduction of R_j continues.

Implicit Representation of *D*. Let $L = \Delta(\text{vert}(K))$ be the full simplex on the vertex set of a finite simplicial complex K. For any fixed dimension $p \in \mathbb{N}_0$ we may represent a p-simplex $\sigma \in L$ as a natural number $s \in \mathbb{N}_0$ by utilizing a combinatorial numbering system as outlined in Remark 3.6 and described in more detail in [2]. As a basis element of the canonical basis $\mathcal{C}(K)$, σ is thereby represented as a tuple $(s, p) \in \mathbb{N}_0 \times \mathbb{N}_0$. Solely from this representation, the combinatorial numbering system allows for an efficient enumeration of all facets (and cofacets) of σ by evaluating sums of binomial coefficients (see [2, pp. 413, 414]). Since the sum of all facets (with alternating sign) is the boundary of σ, the tuple (s, p) implicitly represents the column of D corresponding to σ.
This system can be transferred to the subcomplex $K \subseteq L$ by providing a function that decides if (s, p) represents a simplex L that also lies in K. For a Vietoris-Rips complex at scale $t < \text{diam}(\text{vert}(K))$ the diameter d of a simplex may be added to the tuple representing it, resulting in a triple $(d, s, p) \in \mathbb{R} \times \mathbb{N}_0 \times \mathbb{N}_0$. The aforementioned function is then given by simply comparing d with the scale parameter t. For complexes that do not have such a straightforward condition this approach of representing D may be cumbersome and inefficient however.

Ripser implements the implicit representation of both R and the coboundary matrix $D^\uparrow$, assembled and processed one degree at a time as mentioned in the beginning of the section. The basis elements of $\mathcal{C}^\uparrow(K)$ are represented as tuples of the form (d, s) (in the above notation), omitting the dimension p that is instead determined by the reduction context.

3.3.2. Clearing

This and the following subsection are concerned with the omission of columns that need not be reduced. First, we examine columns that are reduced to the zero column. This is the case whenever the current iteration index j is a birth index. If j is moreover the birth index of a non-essential interval (as opposed to an essential interval) then no information from the j-th iteration is relevant for the output or subsequent iterations:
The basis element V_j added to $\mathcal{E}$ in Line 12 of Algorithm 3.1 will be replaced by R_i for some index $i > j$ and the interval $[j, \infty) \in \mathcal{J}$ will be replaced by $[j, i)$. After the column R_i is reduced, the output corresponding to the non-essential interval $[j, i)$ is thus solely determined by R_i and its pivot index $\text{pivot}(R_i) = j$. Since R_j was reduced to the zero column it is not used for the reduction of any other column either. In summary, at iteration index i it is clear that the past reduction of column j was superfluous. Avoiding such reductions is called *clearing* in [2] and was originally introduced in [12].
During the reduction of D the birth index of a non-essential pair is encountered before its death index which appears later in the filtration order. The reduction of non-essential birth columns is thus unavoidable. Degreewise reduction solves this issue for all but a single degree: Note that due to the face relation, birth and death simplex are separated by one dimension.
In persistent homology the dimension of any birth simplex is decreased by one compared to the corresponding death simplex. This property can be used in conjunction with the reduction of $D^{\downarrow}$ to guarantee that columns corresponding to death simplices are encountered before the corresponding birth columns which are thereby identified for omission in the future.
In the cohomological setting the relationship of death and birth simplex with respect to the dimension is reversed. In this case clearing necessitates a degreewise reduction in increasing dimension.
As Remark 3.17 alludes to, the order chosen for the definition of $D^{\downarrow}$ and $D^{\uparrow}$ is motivated by clearing. These two matrices are the designated input options for Algorithm 3.2 that implements clearing in addition to the implicit matrix representation of R (a version without implicit matrix representation can be found in the appendix). To keep track of identified non-essential birth indices, that is non-zero pivot indices, the map pivot-column-index introduced in the previous subsection is reused. Line 5 of Algorithm 3.2 checks if the current iteration index has previously been a pivot index in which case clearing applies and the iteration is skipped. In addition, the update of the barcode decomposition (Lines 14–16 of Algorithm 3.1) is removed. Instead, non-essential pairs are added when their death-simplex is discovered, that is a column R_j retains a non-zero pivot. In this case the condition of Line 18 is satisfied and we add interval and generator to the output. If R_j is instead reduced to the zero column and thus $\ell = 0$, clearing implies that we found an essential interval (since all columns corresponding to non-essential birth indices are skipped by means of Line 5) and the output is extended accordingly in Lines 22–23.

Algorithm 3.2: The reduction algorithm modified to use implicit matrix representation and clearing

```
1  J := ∅;  E := ∅;  B := ∅
2  V := id
3  pivot-column-index(−) := −1
4  for j := 1 to |K| do
5      if pivot-column-index(j) = −1 then
6          R_j := D_j^↓
7          ℓ = pivot(R_j)
8          while ℓ ≠ 0 do
9              if i := pivot-column-index(ℓ) ≠ −1 then
10                 R_i = D^↓ · V_i
11                 λ := R_ℓj / R_ℓi
12                 R_j := R_j − λR_i
13                 V_j := V_j − λV_i
14                 ℓ = pivot(R_j)
15             else
16                 pivot-column-index(ℓ) := j
17                 break
18         if ℓ ≠ 0 then
19             J := J ∪ [ℓ, j)
20             B := B ∪ R_j
21         else
22             J := J ∪ [j, ∞)
23             E := E ∪ V_j
```

Clearing does not apply in either largest (homological setting) or the lowest desired degree (cohomogical setting) since no birth-simplices can be identified in the absent degree above or below. Note that if this degree is $\dim(K)$ or 0 (in homology and cohomology respectively), no non-essential birth-indices exist anyway. But for most applications the top degree is chosen to be much smaller than $\dim(K)$ which precludes clearing in the top degree in the homological setting. This pinpoints the main advantage the cohomological setting has over the homological setting with respect to computational performance: For many types of filtrations the number of simplices increases substantially with increasing dimension in which case the absence of clearing is most severe in the top degree. This impairs the efficiency of the reduction algorithm in the homological setting severely. The Vietoris-Rips filtration is an example of such a filtration and for this reason Ripser implements clearing as part of the reduction algorithm in the cohomological setting. Ripser implements Line 5 of Algorithm 3.2 indirectly, interjecting a preliminary assembly of all

iteration index that have not been identified to be cleared in the previous degree before the reduction starts. It is important to keep in mind however that computing persistent cohomology instead of persistent homology yields a basis of cohomology representatives instead of homology representatives. This is not desirable for certain applications and the topic of Section 3.5.

3.3.3. Emergent and Apparent Pairs

Clearing addresses columns that would needlessly be reduced to the zero column. In this subsection we review an optimization that concerns columns of D that are already reduced. At the beginning of the j-th iteration of the main reduction loop the column D_j is retrieved and its pivot is checked for uniqueness among pivots of previously reduced columns. If the pivot is unique, D_j is already reduced and Lines 5–8 of Algorithm 3.1 are skipped. This seems to come at a minimum computational cost but the access of D_j does have a non-zero cost associated with it, especially if D is represented implicitly as described in Subsection 3.3.1. In addition, a small implementation-specific overhead related to initializations is incurred at the beginning of every iteration. If the number of iterations is large, these costs accumulate and noticeably impact the overall runtime of the reduction algorithm. However, in many cases it is possible to decide whether D_j is already reduced before it is made available in full or better yet omitting the associated iteration altogether. In this subsection we review two computational shortcuts that are introduced in [2], making use of the following terminology for certain simplex pairs.

Definition 3.18. Let K_I be an essential simplexwise filtration of a finite simplicial complex K and $\sigma, \tau \in K$. With respect to the filtration order, (σ, τ) is called

1. an *emergent facet pair* if $\sigma \subseteq \tau$ is the youngest facet of τ,
2. an *emergent cofacet pair* if $\tau \subseteq \sigma$ is the oldest cofacet of σ,
3. an *apparent pair* if its both an emergent facet and cofacet pair.

The two shortcuts described below identify already reduced columns by finding emergent and apparent pairs. For both we provide a short practical explanation and explain why the check for an emergent or apparent pair outperforms Lines 4–5 of Algorithm 3.1 if its implementation is close to that of Ripser. We refer to [2] for a thorough explanation, connections to discrete Morse theory and further references.

Emergent Pairs Shortcut. Since the boundary matrix D represents the boundary map with respect to the canonical basis in filtration order, a simplex pair (σ_ℓ, σ_j) is an emergent facet pair if and only if $\ell = \text{pivot}(D_j)$. By Lemma 2.59 and Propositions 2.67 and 2.68 it is an emergent (non-essential) persistence pair if and only if D_j is already reduced. The retrieval of D_j and the subsequent pivot check can thus be replaced by the enumeration of only the youngest facet σ_ℓ of σ_j followed by a check to determine if σ_ℓ is

already part of a previously determined persistence pair. If D is represented implicitly as outlined in Subsection 3.3.1, the computational effort to assemble D_j is thus cut short.
The baseline implementation of the reduction algorithm in Ripser facilitates the identification of emergent pairs. Line 4 of Algorithm 3.1 is implemented by enumerating all facets of the simplex σ_j (corresponding to the column D_j) in colexicographic order (as opposed to reverse colexicographic order) disregarding diameters. If $\dim(\sigma_j) = 1$ then all facets are vertices of zero diameter and the facet enumerated first must be in an emergent pair with σ_j. Even if $\dim(\sigma_j) > 1$, the set of simplices of the equal diameter are nevertheless enumerated in reverse filtration order to the effect that once the youngest facet of diameter $\operatorname{diam}(\sigma_j)$ is encountered, the emergent pair including σ_j is determined. The restriction to facets of diameter equal to $\operatorname{diam}(\sigma_j)$ is permissible since the colexicographic refinement of a Vietoris-Rips filtrations guarantees the existence of a facet with diameter $\operatorname{diam}(\sigma_j)$ if $\dim(\sigma_j) > 1$ and any potential emergent persistence pairs must have zero persistence. Once the required facet has been identified, the negation of the condition in Line 5 of Algorithm 3.1 is evaluated to determine if the found emergent pair is moreover a persistence pair in which case the rest of the j-th reduction may be skipped after recording the new non-zero pivot.

Apparent Pairs Shortcut. To decide whether an emergent pair is also a persistence pair requires the check of all previously determined persistence pairs, or equivalently the check of all non-zero pivot entries. This means in particular that the emergent pairs shortcut is dependent on previous iterations and therefore cannot be separated from the reduction loop.
The condition for an apparent pair is more restrictive but guarantees that any apparent pair is also a persistence pair, as [2, Lemma 3.3] shows. Since any apparent pair is also an emergent pair, we may reuse the above argument to conclude that for degree $p > 1$ any apparent pairs is a zero-persistence pair. By determining all apparent pairs before entering the reduction loop allows us to skip the corresponding iterations altogether thereby avoiding any computational costs related them. If zero-persistence pairs are required to be part of the output, they may be added as soon as they are identified, circumventing the matrix reduction regardless. Note, that while columns corresponding to simplices in apparent pairs are already reduced, they might still be required for the reduction of other columns and must be included in the check of Line 5 of Algorithm 3.1. For the adapted Algorithm 3.2 this means that the map pivot-column-index needs to be updated for apparent pairs in advance of the reduction.
Regarding the identification of apparent pairs before entering the reduction loop in degree p, every p-simplex σ_j is checked for being part of an apparent pair of the form (σ_i, σ_j) and (σ_j, σ_k) (if $p = 0$ or $p = \dim(K)$ then the former or latter check do not apply, respectively). If either pair is apparent, the column with index j may be omitted from the reduction as described above. By definition either check may be expressed as a test for both an emergent facet and cofacet pair and Ripser implements them as such in a similar manner as described for the emergent pairs shortcut. The apparent pairs shortcut is compatible

with the clearing (introduced in the previous subsection) and should be secondary to it since testing for apparent pairs is substantially more expensive.

3.3.4. Degree Zero and Connected Components

Recall from Proposition 2.15 that the homology of a finite simplicial complex K in degree zero is determined by the connected components of K. This provides an alternative approach to the computation of the barcode decomposition in degree zero that was first introduced in [20] and may be implemented as follows for an essential simplexwise filtration K_I.
Let A_i denote the singleton set containing the i-th vertex in $\mathrm{vert}(K) = \{v_1, \dots, v_n\}$. Iterating through the set of 1-simplices viewed as edges between vertices and merging two sets A_i and A_j whenever the current edge has an end point in either set computes the connected components of K (a form of Kruskal's Algorithm). This straightforward approach also yields the barcode decomposition in degree zero if the sets are merged to retain the older set, that is to say if the edge $\sigma_k = \{v_i, v_j\}$ connects A_i and A_j in the k-th iteration, the set containing the older vertex is retained. If for instance $v_i < v_j$, this implies that $A_i := A_i \cup A_j$ is retained. This rule is sometimes referred to as the *elder rule* [19, Chapter VII]. In our terminology this means that the addition of σ_k bounds the previously unbounded younger v_j, producing a non-essential pair (v_j, σ_k) with persistence interval $[j, k)$. Essential indices are determined after all 1-simplices have been processed, taking the oldest vertex v_i in each remaining connected component A_i as an essential simplex with corresponding persistent interval $[i, \infty)$. Note that edges that do not connect two previously disjoint set A_i and A_j connect two vertices within a connected component. In the context of persistent homology this means that it is the birth simplex of a homology class in degree 1.
For a degreewise implementation of the reduction algorithm, this approach may replace the reduction of D^1. If only the barcode is desired it is also applicable in the cohomological setting, replacing the reduction of $(D^\perp)^0$. In the latter case, the edges identified as homological death simplices are identically birth simplices in persistent cohomology by duality (see Proposition 2.80) and determine those indices that clearing omits for the reduction of $(D^\perp)^1$.
Ripser implements this method to replace the reduction of the coboundary matrix in degree zero using a union-find data structure that maintains the sets A_i and provides the means to merge two components. For large data sets this has substantially improved performance over the reduction of $(D^\perp)^0$ since the number of operations required for each iteration is small [20, p. 523].

3.3.5. Parallelization

The reduction algorithm is well suited for a parallel implementation. In this subsection we provide a brief summary of the algorithmic changes to the reduction algorithm introduced in [27] that make a parallel reduction possible.
In order to reduce a column D_j, Algorithm 3.1 adds a scalar multiple of a previously reduced column R_i satisfying $\text{pivot}(R_i) = \text{pivot}(D_j)$ to reduce the pivot index of D_j. The particular choice of column is arbitrary as long as the constraint $i < j$ is satisfied. In fact the column chosen to reduce the pivot index of D_j need not itself be reduced. While Proposition 2.60 and Algorithm 3.1 suggests a sequential reduction of D with increasing column index we may just as well reduce the columns of D out of order. To avoid ambiguities we only refer to columns of R going forward, initialized as $R_j := D_j$. Described in [27] is an algorithm for such an out-of-order reduction that rests on pivot-column-index introduced in Subsection 3.3.1 mapping a pivot index ℓ to the smallest column index i such that $\text{pivot}(R_i) = \ell$ if such a column exists and -1 otherwise (in [27] this map is denoted by `pivots`). In other words, pivot-column-index(ℓ) stores the index of left-most candidate to reduce a column with pivot index ℓ or indicates that none exists. Note that in contrast to a sequential reduction, multiple candidates may exist since several partially reduced column may have the same pivot index. To understand the reduction of a column consider a column R_j with pivot index ℓ that is to be reduced. We distinguish three cases contingent on $i = \text{pivot-column-index}(\ell)$:

1. $(i = -1)$ R_j cannot be reduced further and pivot-column-index(ℓ) is set to j.
2. $(i < j)$ R_j is reduced by a adding a scalar multiple of R_i.
3. $(i > j)$ R_j cannot be reduced by R_i since only columns of smaller index are permissible. Instead R_i is reduced by adding a scalar multiple of R_j after which pivot-column-index(ℓ) is set to j.

The reduction of R_i instead of the current column R_j in the third case is critical to ensure that R is reduced after each column has been processed. This out-of-order reduction may now be parallelized by dividing $R := D$ into several disjoint sets of columns called chunks, each of which is assigned to a separate thread that together reduce the matrix concurrently.
To implement such a parallel reduction, several race conditions and memory management challenges need to be addressed. For this we refer to [27] aside from highlighting one such instance: The update of pivot-column-index. It is possible that two threads simultaneously need to update pivot-column-index(ℓ) after both reduced their respective current column to have the same pivot index ℓ. In this case the implementation must ensure that one thread updates pivot-column-index(ℓ) while the other is notified and continues with the reduction to accommodate this update.
We conclude this subsection with a remark on one additional opportunity for parallelism arising from the specific implementation of the apparent pairs shortcut in Ripser. As

mentioned in Subsection 3.3.2 and 3.3.3, Ripser determines a subset of column indices that need to be consider for reduction, eliminating superfluous indices by clearing and the application of the apparent pairs shortcut. The latter requires to check all indices for an apparent pair which Ripser implements sequentially, checking one index at a time. This is easily parallelized since the check for an apparent pair depends only on the index at hand. The recently released software giotto-ph provides such a parallel implementation of the apparent pairs check in addition to a concurrent matrix reduction.

3.3.6. Summary and Further Directions

The various optimizations reviewed in the previous subsections all contribute to the efficient implementations of persistent homology that in recent times have enabled the use of persistent homology for a variety of applications [6, 8, 21, 32]. Ripser implements all but the parallelization outlined in the previous subsection and may be considered the reference implementation of persistent homology at the time of writing.
At the core of Ripser's efficient implementation lies the implicit matrix reduction and representation of simplices and chains, reducing the memory footprint enough to allow for the analysis of large data sets. Utilizing this efficient organization of the internal data, a degreewise reduction in the cohomological setting is implemented, effectively computing persistent relative cohomology. This facilitates clearing in all non-zero degrees which is implemented alongside the emergent and apparent pairs shortcut, drastically reducing the amount of columns required to be reduced. In degree zero, the matrix reduction is replaced by the much more efficient computation of connected components.
This extensively optimized implementation of the reduction algorithm only solves a restriction of Problem 3.1 however. Concerning the input, Ripser restrictrs to Vietoris-Rips filtrations (that my be cut off at a diameter threshold) and prime fields $\mathbb{F}_p$. Ripser's output on the other hand solely consists of the barcode, discarding all interval module generators, albeit cohomology representatives may be stored and added to the output without drastic changes. Homology representative however are not easily obtained since the cohomological setting is crucial to the substantial performance benefit of clearing. This motivates Section 3.5 that investigates how homology representatives may be computed in an efficient manner.

Further improvements of the computational efficiency are an active subject of research. We refer for instance to [4] that investigates performance improvements connected to the chosen vertex order (that in Ripser is determined by order of the input data) for tree-like data sets. Another example is the recent release of giotto-ph [31] that improves slightly upon the implementation of Ripser in addition to the use of parallelization as described in Subsection 3.3.5. Further improvements in this regard might be achieved by addressing the questions raised in [27] and the use of GPUs.

3.3.7. Performance Metrics

In preparation for the performance analysis that is part of Section 3.4 and 3.5 we conclude this section with a short outline of the internal data structures and elementary operations used for the matrix reduction in Ripser, expanding on the short remarks made in Subsection 3.3.1. For a detailed description of the implementation of Ripser we refer to [2, Section 4].

- Simplices and identically the basis elements of the canonical basis $\mathcal{C}^*(K)$ are represented as tuples $(d, s) \in \mathbb{R} \times \mathbb{N}_0$, encoded in the type `diameter_index_t` that is a `std::pair<float, int64_t>`. The first component caches the diameter of the simplex, the second is the number assigned by the combinatorial numbering system (see Remark 3.6).

- To represent the scalar multiple of a basis element, the above tuples are extended by a coefficient to triples $(d, s, c) \in \mathbb{R} \times \mathbb{N}_0 \times \mathbb{F}_p$. Ripser implements this by packing the coefficient into the 64 bit word that encodes the index s if $p > 2$. Otherwise the above encoding suffices since a non-zero scalar multiple is unique in $\mathbb{F}_2$. In either case the resulting type is denoted by `diameter_entry_t`.

- Whenever a column of $D_j^\perp$ is required, it is made available by enumerating all cofacets of the corresponding simplex σ_j. This enumeration relies only on the representation (d, s) of σ_j (as well as the dimension p) and is implemented by the class `simplex_coboundary_enumerator`.

- The columns of R and V that are coordinate vectors with respect to $\mathcal{C}^*(K)$ are represented by the underlying linear combination of basis elements as a chain in $C^\bullet(K)$ (see Remark 2.66). Ripser encodes such a linear combination as a `std::priority_queue` maintaining a `std::vector` of `diameter_entry_t` entries in reverse filtration order (that is first by decreasing diameter d and then by increasing index s). Since the diagonal entries of V are always 1 they are not stored explicitly.

- In this setting a column addition corresponds to the addition of linear combinations which is carried out by inserting the `diameter_entry_t` values to be added into the receiving priority queue using the function `push`. Importantly, this has the consequence that columns in Ripser are evaluated lazily, allowing more than one scalar multiples of the same basis element. These may in particular add up to zero.

- The function `top` provides constant-time access to the largest basis element (with respect to the reverse filtration order) occurring in the linear combination representing a column. This allows for a straight forward retrieval of the pivot entry, the value at the pivot index of a column. Due to lazy evaluation it is however required to `pop` elements off the queue until the second largest basis element is encountered (if it exists). If it turns out that the sum of all encountered scalar multiples of the

largest basis element is equal to zero the search for the pivot continues.

- Due to the implicit matrix representation of R the addition of a column R_i to the currently processed column R_j requires the recomputation of R_i as the product DV_i. This is implemented by the function `add_coboundary` which calls `add_simplex_coboundary` for each element in V_i and its implicit diagonal entry. This latter function inserts σ_i into the priority queue encoding V_j then enumerates all cofacets of σ_i which in turn are inserted into the queue encoding R_j.

In the following two sections we compare the performance characteristics of a number of modified reduction loops. For this purpose, we consider the following performance metrics for each degree separately:

1. The time elapsed between the start of the first and end of the last iteration of the reduction loop (in a single degree) as measured with `std::chrono::steady_clock` and `std::chrono::duration`.

2. The maximum reported virtual memory size (vmSize) in KiB (1024 bytes) during the reduction as reported by the Linux kernel interface `/proc/self/statm`. This file is read after every column addition during the reduction and at once again at the end of each iteration, updating the maximum encountered value if necessary.

3. The number of columns to be reduced (excluding columns identified to be omitted by clearing and the apparent pairs short cut but including those subject to the emergent pairs shortcut). This is a rather coarse performance gauge that also indicates the number of persistence pairs if clearing is employed (excluding zero-persistence apparent pairs).

4. The number of calls to `add_coboundary` and `add_simplex_coboundary`. While these are a finer measure of performance, the number of required column additions may vary considerably between columns. Considering only the total number of calls misses performance characteristics hidden by this variance.

5. The number of calls to `push` and `pop`. These two functions lie at the core of Ripser and the total number of calls to them faithfully reflects how many elementary operations have been carried out. It is important to keep in mind however that the cost of either function is logarithmic in the number of stored entries and therefore varies from as little as one to about 30 required heap operations (plus a constant overhead).

All measurements for (1) and (2) are taken on a computer with an Intel Core i7-3520M processor at 2.9GHz and 16 GiB of DDR3 RAM at 1600MHz using an operating system based on the Linux kernel version 5.16.0-arch1-1. Used for program compilation is the GNU compiler collection (gcc) version 11.1.0 with the optimization flag O3 enabled. While on our system `std::chrono::steady_clock` reports a resolution of 1 nanosecond, the actual accuracy of the time measurements is likely to be lower which nonetheless has

proven sufficient for our purposes.
Note that the performance indicators of (2) – (5) abstract the actual computational cost that is measured by (1) and (2) in elapsed seconds and maximum required memory. We consider them because they characterize the runtime behaviour intuitively and provide insights that are otherwise hidden behind measurements that depend on the hardware and operating system.

3.4. Efficient Persistent Relative Homology

The previous section discusses algorithmic optimizations and implementation schemes for the reduction algorithm that improve its computational efficiency. In this section we approach the topic of performance from a different angle, motivated by feasibility more so than the pursuit of an optimal implementation. Considering Vietoris-Rips complexes, the total number of simplices grows exponentially in the number of data points [30, Table 1]. Even if only degree zero and one are considered, the number of required column reductions easily grows beyond the capabilities of current state-of-the-art implementations due to quadratic scaling in the number of data points. The analysis of the SARS-CoV-2 genom data presented in [6] provides an example near the limits of what is currently feasible. There, a data set of around 150,000 data points is the basis for a Vietoris-Rips filtration cut off at a relatively low distance threshold. On server-grade hardware, Ripser computed the barcode of persistent homology up to degree one in roughly 40 minutes requiring about 45 gigabytes of memory [6, Figure 4].
This unfavorable scaling is not unique to Vietoris-Rips complexes, Čech complexes being another well-known example. We refer to [30, Section 5.2] for an overview of other commonly used complexes and their properties.
If a data set is too large for the application of persistent homology within the given time and hardware constraints, a simple solution is to restrict to a smaller subset of points. This reduces the number of columns in the (co)boundary matrix substantially as Figure 3.2 illustrates in the left frame for Vietoris-Rips filtrations. It shows that the fraction of columns required to be reduced (of those required for the whole data set) decreases at least linearly in the fraction of data points cut off. To which extent the topological features identified for such a subset reflect those of the whole data set depends on the particular subset chosen. Structure exclusive to the omitted parts is certainly missed but also features that only partially lie in the excluded set of data points may vanish. A pathological example can be constructed with the data set introduced in Chapter 1.

Example 3.19. The 45 point data set visualized in Figure 1.1 consists of two parts: 30 samples of the outer circle and 15 samples of the inner circle. For each part separately, consider the persistent homology of the Vietoris-Rips filtration (with chains over $\mathbb{F}_2$). The barcode and homology representatives in degree one corresponding to these two

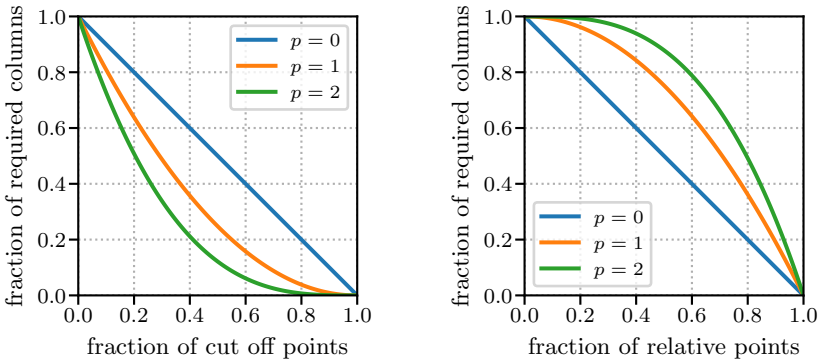


Figure 3.2.: With respect to the Vietoris-Rips filtration on a data set of 100 points, the left plot shows how the fraction of required column reductions decreases as the fraction of points cut off from the complete data set increases. The right plot shows the same relationship of fractions for persistent relative homology, taking the cut off points to induce the relative subcomplex.

filtrations are shown in Figure 3.3 in orange and red, respectively. In both instances persistent homology identifies only a single feature that describes either the outer or inner circle. In comparison with the barcode for the whole data set (also shown in Figure 1.1), the ten features of shorter persistence have vanished from either barcode.
Figure 3.3 also includes a third barcode and set of representatives in blue that are the result of computing persistent relative homology (as the barcode decomposition of $H_1(-, L) \circ K_I$) as described in Section 2.8. In this instance the filtration K_I is the Vietoris-Rips filtration on all 45 points and the relative subcomplex L is the Vietoris-Rips complex on the points comprising the outer circle. The corresponding barcode contains eleven bars, one associated with the inner circle (identical to the single bar in the barcode above it) and ten additional bars that reflect the ten features of short persistence also detected by persistent absolute homology applied to the whole data set. Compared to the barcode shown in Figure 1.1, the only missing bar corresponds to the outer circle that lies entirely within the relative subcomplex.

The above example proposes an alternative to the simple but often overly lossy method of cutting off data points: Computing persistent relative homology with a filtered absolute and a constant relative part (as opposed to persistent relative homology with a constant absolute and filtered relative part). Moving the previously excluded portion of data into a relative subcomplex, eleven out of twelve features are just as well detected by persistent relative homology as in the absolute case with closely matching persistence intervals. This correspondence of barcodes is not always as straight forward however and is discussed in more detail in Subsection 3.4.3. Beforehand, we briefly reiterate the adaptations required for the efficient computation of persistent relative homology, described our implementation and present our performance analysis.

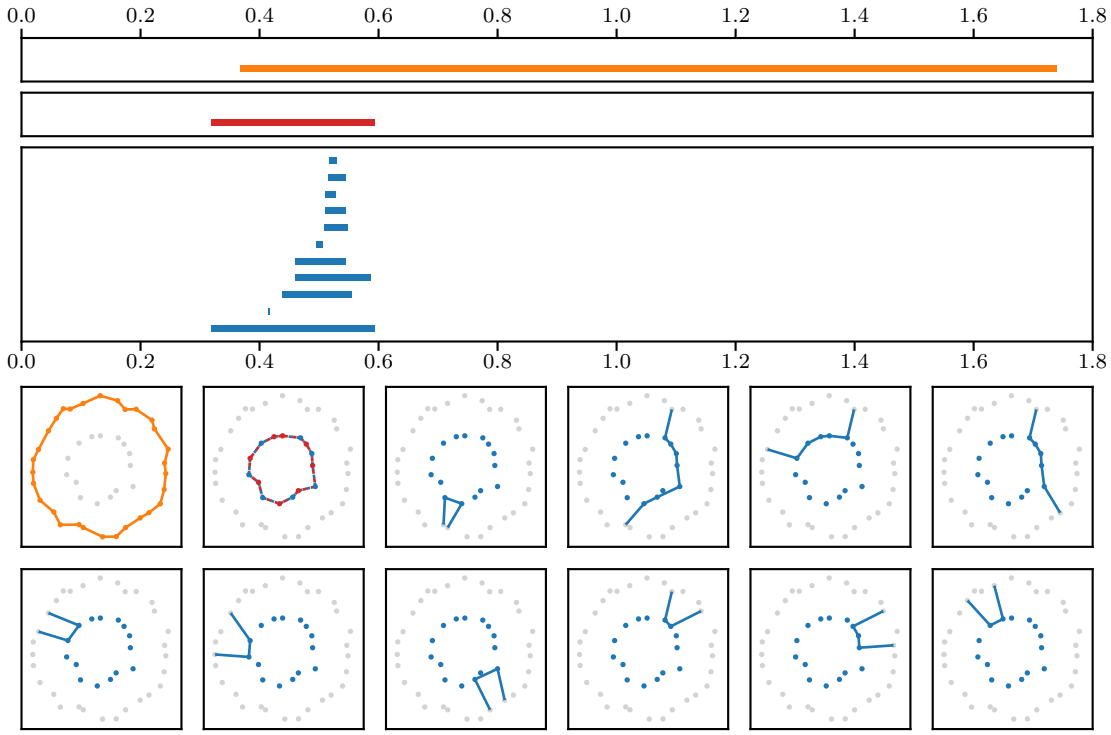


Figure 3.3.: Recall the data set visualized in Figure 1.1. Shown in the top-most three frames are three barcodes in degree one: In orange and red, the barcode of the persistent (absolute) homology of the Vietoris-Rips filtration on the outer and inner circle, respectively. In blue, the barcode of the persistent relative homology of the Vietoris-Rips filtration of the whole data set, where the complex on the outer circle is taken as the relative subcomplex. Associated to each bar is one of the representative shown in the bottom twelve frames. The dashed cycle encompassing the inner circle is identically the representative of the sole red and bottom-most blue bar.

Remark 3.20. The term "persistent relative homology" is overloaded and ambiguous. Given a filtration K_I of a simplicial complex K with subcomplex L, it may refer to either the persistence module $H_\bullet(-, L) \circ K_I$ or $H_\bullet(K, -) \circ K_I$, both being based on relative chain groups but having fundamentally different structure (see Section 2.8 for details). As emphasized above, this section is concerned with the former type that filters the absolute complex while keeping the relative subcomplex constant. This remark likewise applies to persistent relative cohomology.

3.4.1. Computing Persistent Relative Homology

The methods of Section 3.1–3.3 may be transferred to the relative setting without conceptual change following the outline given in Section 2.8. We briefly recapitulate the key concepts with regard to an efficient computation.
Let K_I denote an essential simplexwise filtration and let $L \subseteq K$ be a subcomplex. Providing the *relative boundary matrix* $\widetilde{D}$ representing the relative boundary map $\widetilde{\partial}_\bullet : C_\bullet(K,L) \to C_\bullet(K,L)$ with respect to the canonical basis $\mathcal{C}(K,L)$ in filtration order to Algorithm 3.1 solves Problem 3.12 for the instance $M = H_\bullet(-,L) \circ K_I$ without any further changes.
The same transition applies in the cohomology setting, replacing $D^\perp$ with the *relative coboundary matrix* $\widetilde{D}^\perp$ representing $\widetilde{\partial}^\bullet : C^\bullet(K,L) \to C^\bullet(K,L)$ with respect to $\mathcal{C}^*(K,L)$ in reverse filtration order. However, the instance $M = H^\bullet(-,L) \circ K_I$ of Problem 3.12 is not immediately solved by reducing $\widetilde{D}^\perp$. As required for the computation of persistent absolute cohomology, a translation of the output is necessary that performs the transition from relative to absolute cohomology (see Section 3.2 and Proposition 2.83). More precisely, the relative coboundary matrix does not encode the projections $C^\bullet(K_i, L \cap K_i) \twoheadrightarrow C^\bullet(K_j, L \cap K_j)$ of our target $M = H^\bullet(-,L) \circ K_I$ but rather the inclusions

$$C^\bullet(K,L)/C^\bullet(K_i, L \cap K_i) \hookrightarrow C^\bullet(K,L)/C^\bullet(K_j, L \cap K_j)$$

for $i \leq^{\mathrm{op}} j$. The latter may be defined by the inclusion of bases

$$\mathcal{C}^*(K,L) \setminus \mathcal{C}^*(K_i, L \cap K_i) \hookrightarrow \mathcal{C}^*(K,L) \setminus \mathcal{C}^*(K_j, L \cap K_j)$$

that simplifies to $\mathcal{C}^*(K,L) \setminus \mathcal{C}^*(K_i) \hookrightarrow \mathcal{C}^*(K,L) \setminus \mathcal{C}^*(K_j)$. In cohomology, these maps induce the structure maps of a persistence module that may justify the name *persistent relative relative cohomology*, where the first "relative" refers to taking relative cochains with respect to $C^\bullet(K_i, L)$ for $i \in I$ and the second to the general setting of relative cochains $C^\bullet(K,L)$ inherent to the topic of this section.
These algebraic details aside, the aforementioned translation of barcode and representatives proceeds in exactly the same manner of that required for persistent absolute cohomology which is described in the last bullet point towards the end of Section 3.2.

Since the transition to relative (co)homology only affects the input matrix, requiring no modifications to the reduction algorithm itself, it follows that all optimizations of Section 3.3 may be employed without any change. For the implementation of persistent relative cohomology based on Ripser this means that besides an additional program parameter specifying those data points which induce the relative subcomplex, only the enumeration of facets and cofacets needs to be adapted. In our implementation we introduce two new functions for this purpose:

- `is_relative_vertex` takes a vertex represented by its index in the combinatorial numbering system and returns `true` if it lies in the specified relative subcomplex.

Since this index is precisely the position of the corresponding data point in the data set, checking whether this data point lies in the excluded portion of points (specified for instance by a list of intervals of data set indices) suffices.

- `is_relative_simplex` takes a simplex represented by its index in the combinatorial numbering system and its dimension and returns `true` if it lies in the specified relative subcomplex. This is implemented by enumerating all of its vertices with `get_max_vertex` (a function provided by Ripser) and checking each with `is_relative_vertex`. As soon as a vertex outside the relative subcomplex is encountered, it is clear that the simplex in question does not lie in the relative subcomplex and `false` is returned. Only if all vertices pass the check, `true` is returned.

At every point of (co)facet enumeration, `is_relative_simplex` is checked for each enumerated (co)facet. If `true` is returned the (co)facet in questions is discarded and the enumeration proceeds. Such enumerations occur for instance as part of column additions, apparent pair checks and the assembly of those column indices passed to the actual reduction. In particular, this implements an implicit matrix representation of $\widetilde{D}^{\perp}$. Only during the aforementioned assembly of indices (implemented by `assemble_columns_to_reduce`) is it necessary to retain all simplices of K in the current degree, relative or not. This is the case for the list of simplices `next_simplices` that is assembled for the purpose of enumerating those for the next degree. If relative simplices are omitted, the enumeration of cofacets will certainly skip over non-relative simplices as a consequence of calling `has_next` with the optional parameter `all_cofacets` set to `false` (see [2, p. 414] for details).

3.4.2. Performance Analysis

In comparison with the computation of persistent absolute homology for either the whole data set or a smaller subset thereof, relative homology falls in between these two extremes. It promises improved performance over the computation of persistent absolute homology for the whole data set while capturing structure that partially lies in the data excluded by the persistent absolute homology applied to a subset.
In the following, we analyze the performance of computing persistent relative homology applied to Vietoris-Rips filtrations. We restrict the computation to the barcode, which allows us to use the cohomological instead of the homological setting since both give rise to the same barcode (see Corollary 2.79).
The first consideration is the number of columns required to be reduced which is equal to the number of basis elements in the canonical basis and thereby the number of p-simplices lying outside the relative subcomplex. Given a data set of n points, the number of p-simplices of the full simplex on this data is given by $\binom{n}{p+1}$. If r denotes the fraction of data points used to induce the relative subcomplex, the fraction of p-simplices lying in

this subcomplex is given by

$$\frac{rn}{n}, \qquad \frac{rn(rn-1)}{n(n-1)}, \qquad \frac{rn(rn-1)(rn-2)}{n(n-1)(n-2)}$$

for $p \in \{0,1,2\}$, respectively. For a fixed data set of size n, the number of columns required to be reduced in order to compute the persistent relative cohomology in degree p is thus governed by a polynomial in r of degree $p+1$. The benefit of persistent relative cohomology thus diminishes with increasing degree. Despite this adverse scaling in p, a substantial amount of column reductions can be avoided especially in degrees zero and one, provided that the fraction r of points in the relative part is chosen to be sufficiently large. The right frame of Figure 3.2 visualizes the above fractions for $n = 100$ as a function of $r \in [0,1]$, illustrating the extent to which column reductions are avoided by computing persistent homology in the relative instead of the absolute setting.

Remark 3.21. As a function of n, the derivatives of the above three fractions with respect to n quickly converge 0. For degree one (the second fraction) and $n = 100$ the maximum value with respect to $r \in [0,1]$ of the derivative lies well below 0.0001.

Example 3.22. Continuing from Example 3.19, we compare the number of required column reductions in order to reduce the absolute and relative boundary matrix (in degree one) in the table below. In the absolute setting, both the full data set and the 15 point subset comprising the outer circle are considered. In the relative setting we take those 15 points instead into the relative subcomplex. We distinguish between a version without optimizations, a version that implements clearing (see Subsection 3.3.2) and a fully optimized version, employing both clearing and the apparent pairs shortcut (see Subsection 3.3.3).

	abs. hom. 45 pts.	rel. hom, $r = 1/3$	abs hom. 15 pts.
no-opt	990	550	105
cl	946	540	91
cl-app	12	11	1

Without the apparent pairs shortcut, the effect of computing persistent relative cohomology is considerable. It disappears almost entirely when the apparent pairs shortcut is employed in addition. The only column reduction avoided in this case corresponds to the homology class of the outer circle which lies entirely in the relative subcomplex.

Example 3.22 foreshadows an effect that we observe in many instances whenever clearing is used in combination with the apparent pairs shortcut: The amount of columns required to be reduced does not decrease substantially until the ratio r of relative points is high. On the contrary, for small values of r the number may increase. This adds a dynamic to the performance characteristics that is difficult to interpret. For this reason, we analyze the performance of persistent relative cohomology with and without the emergent and

apparent pairs shortcut and moreover take more granular performance indicators into consideration. The experimental setup is summarized as follows.

- The computation of (the barcode of) persistent relative homology is implemented as a degreewise reduction in the cohomological setting with chains over $\mathbb{F}_2$. Our implementation is based on Ripser v.1.2.1 modified as outlined in Subsection 3.4.1 and uses implicit matrix representation (see Subsection 3.3.1) as well as clearing (see Subsection 3.3.2). The optimizations described in Subsection 3.3.4 and 3.3.5 are not implemented and all representatives are discarded.
- We distinguish between two versions, one that does not use the emergent and apparent pairs shortcut and one that does, implemented as `ripser_cohom_rel.cpp` and `ripser_cohom_rel_optimized.cpp`, respectively. Both are available in [37].
- In order to gain an overview of the effect of computing persistent relative cohomology, a range of values $r \in [0, 1]$ determining the fractional part of data points inducing the relative subcomplex is considered. Our analysis uses 20 increments of 0.05 from 0 to 1 resulting in 21 examined values for r. Motivated by time series data, the relative portion of data points is always a prefix of the whole data set, that is the first $\lfloor rn \rfloor$ data points, even if the data is ordered arbitrarily.
- The indicators of performance are those described in Subsection 3.3.7, including runtime measurements of elapsed time and maximum required (virtual) memory, the number of columns required to be reduced as well as the number of calls to various elementary functions used for the reduction.
- Instead of absolute values we present these metrics relative to the instance $r = 0$. This means that all shown values are factors of those values corresponding to persistent absolute cohomology.

In this section we show results for two data sets and degree $p = 1$. The data set *random100* is a 100 point uniform random sample of $[0, 1]^4$ taken from [33]. As a random sample this data set is not expected to contain distinctive topological features and is intended as a base line for testing. We prefer this data set to the similar *random* data set of 50 random points in $[0, 1]^{16}$ used in [2, 30] because the increased number of points minimizes the performance impact of overhead. The choice of substantially larger data sets on the other hand is inconvenient for a concise visualization of barcodes.
The second considered data set is a distance matrix of 15,763 aligned SARS-CoV-2 viurs genome sequence created as part of [6]. The raw genome data was obtained from GISAID [23]. The data preprocessing, including sequence alignment and the generation of the distance matrix is described in [6, pp. 13–14]. Due to our limited computing resources we consider a 10,000 point prefix of this data set, in the following denoted by *covid10000*, and use a diameter threshold of $t = 2$ imitating the setup of [6].
The results of our analysis are shown in Figure 3.4 and 3.5. For both data sets and both versions (with and without the emergent and apparent pairs shortcut), computing

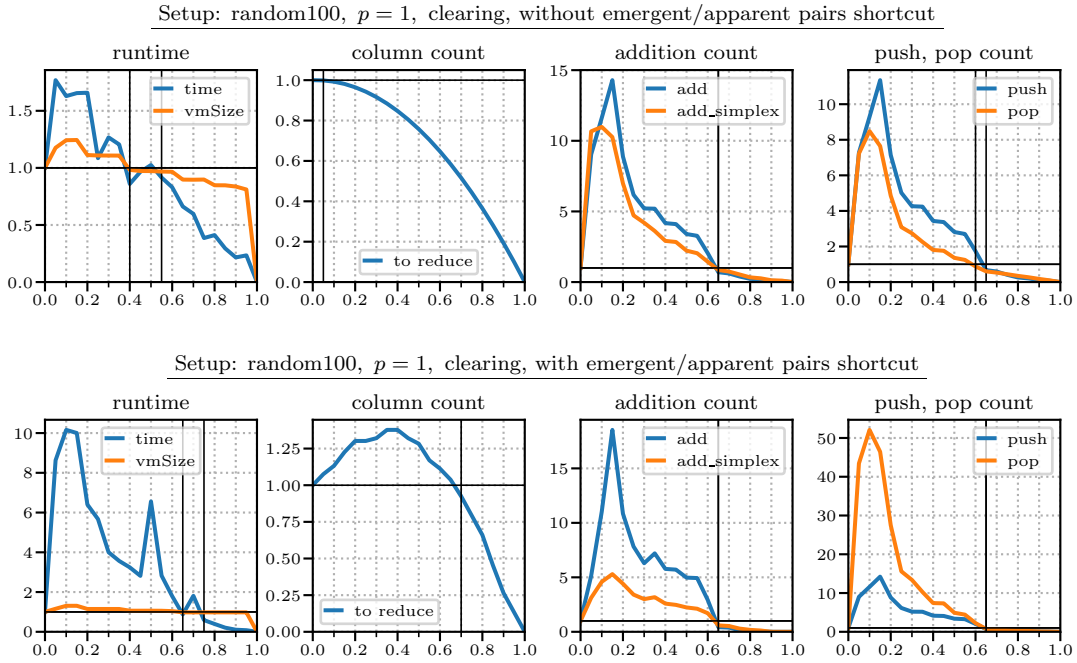


Figure 3.4.: The performance analysis of persistent relative cohomology in comparison to persistent absolute homology for the data set random100. Plotted against 21 values of $r \in [0, 1]$ specifying the ratio of data points taken into the relative subcomplex are the performance indicators introduced in Subsection 3.3.7, relative to the case $r = 0$ (corresponding to persistent absolute homology). All shown values are thus factors of the value at $r = 0$. The vertical guide lines in black indicate the smallest value of r after which the respective performance metric surpasses the reference case $r = 0$.

persistent relative cohomology improves the runtime with respect to elapsed time over persistent absolute homology if r is sufficiently large. For the covid10000 data set, the thresholds to obtain a performance improvement regarding the elapsed runtime are $r = 0.3$ and $r = 0.35$, the latter corresponding to the optimized version implementing apparent pairs. For the random100 data set these values are larger with $r = 0.55$ and $r = 0.75$. Noteworthy is the number of required column reductions that behaves as expected if the apparent pairs shortcut is not implemented (compare with Figure 3.2) but otherwise exhibits the aforementioned initial increase in comparison with the absolute case, particularly for the random100 data set. A possible explanation is the linear decrease of the number of non-essential pairs with cohomological death simplex in degree zero in contrast to the quadratic (thus slower) decrease in degree one. This leads to an increased number of non-essential pairs with death simplex in degree one and thereby a similar increase in the number of required column reductions compared to the absolute case. In other words, the effect of clearing is affected by the removal of vertices (moved to the relative subcomplex) causing a proliferating number of additional non-essential pairs as the degree

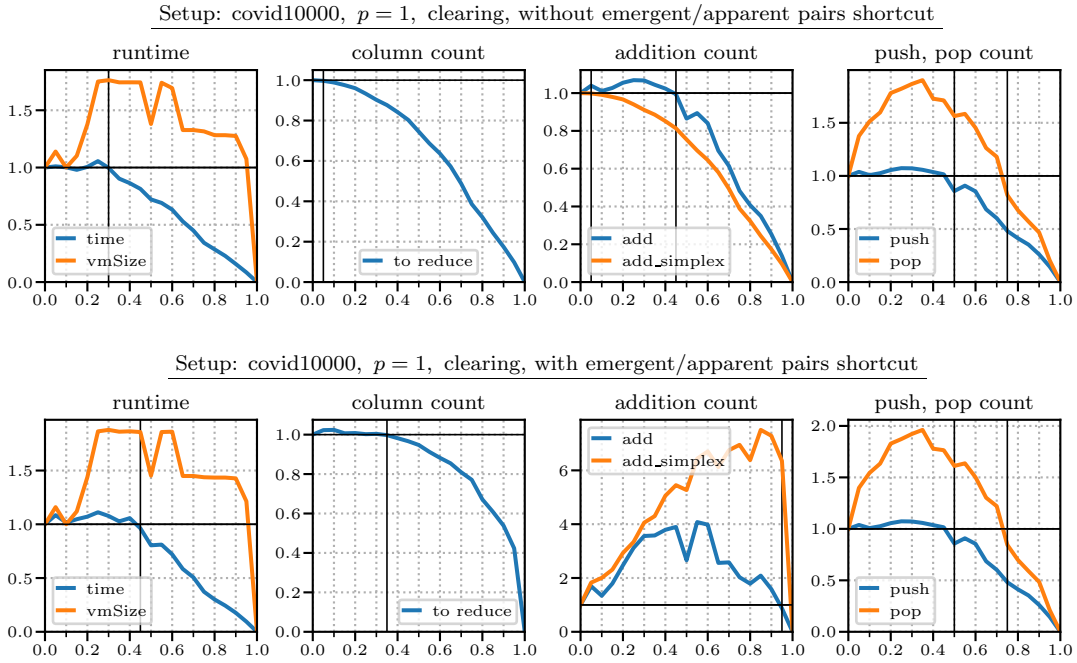


Figure 3.5.: The performance analysis of persistent relative cohomology for the covid10000 data set, presented in the same fashion as in Figure 3.4. All computations were performed for a diameter threshold $t = 2$, cutting off the Vietoris-Rips filtration after $\mathrm{Rips}_2(X)$, imitating the setup of [6].

increases. This is effect is amplified by the elimination of columns corresponding to apparent pairs.

The dynamic for additions, `push` and `pop` counts is hard to interpret and outright surprising considering the optimized version applied to the covid10000 data set. One would expect the metrics to be closely related to the number of column additions performed but this is not necessarily the case. While none of these metrics seem to be tightly coupled to the measured runtime, we expect the numbers of calls to `push` to be in close correlation with the amount of required memory. This is confirmed by the matching r values that correspond to the values of measured memory consumption and the number of calls to `push` in most instances. Particularly for the covid10000 data set this behavior may become problematic with respect to memory consumption.

In the appendix, we provide additional results of our performance analysis. There, we consider three more data sets and two additional degrees. Considering the performance analysis as a whole, the efficiency benefit of persistent relative homology seems to be highly variable and dependent on the data set at hand. In most instances, a performance improvement is observable for sufficiently large choices of r but the erratic behavior of the more abstract performance indicators asks for additional investigation.

3.4.3. Barcode Correspondence

The barcodes shown in Figure 3.3 suggest that the barcode of persistent relative homology is in some sense contained in that of persistent absolute homology, missing only those bars associated to homology classes entirely within the relative subcomplex. Generally, the correspondence of barcodes is not this simple. In fact, the barcode corresponding to persistent relative homology may contain more bars than that of persistent absolute homology as discussed in the previous section and displayed by in the plots titled "column count" in Figure 3.4 and Figure 3.5.
The theoretic framework to analyze the correspondence of barcodes is that of *matchings* that we summarize briefly, following [3, Subsection 2.2]. Generally, a matching from a set A to a set B consists of two subsets $A' \subseteq A$ and $B' \subseteq B$ together with a bijection $m : A' \to B'$. Equivalently, m may be interpreted a subset of $A \times B$, containing a tuple (a, b) if and only if $a \in A'$ and $m(a) = b$. Given a matching m from A to B and a second matching n from B to C we define their composition to be the set

$$n \circ m \;=\; \big\{(a,c) \in A' \times C' \mid \exists b \in B' \text{ such that } (a,b) \in m \text{ and } (b,c) \in n\big\} \;\subseteq\; A \times C.$$

Taking sets as objects and matchings as morphisms forms a category denoted by **Mch**. Applied to our setting, the objective is to obtain a matching of barcodes that reflects the algebraic connection between absolute and relative cohomology. For this purpose, fix a degree p and recall from Section 2.3 the long exact sequence of relative homology. Dualized, this sequence yields morphisms

$$\rho : H^p(K_i, L) \to H^p(K_i) \quad \text{and} \quad \delta_p^* : H^{p-1}(L \cap K_i) \to H^p(K_i, L)$$

for all $i \in I$, the latter being the dual of the connecting homomorphism $\delta : H_p(K_i, L) \to H_{p-1}(L)$ (abbreviating the intersection $L \cap K_i$ to L for the notation of relative cohomology). Interpreted as pointwise instances, they each specify a morphism of persistence modules, mapping

$$H^p(-, L) \circ K_I \;\to\; H^p \circ K_I \quad \text{and} \quad H^{p-1}(L \cap -) \circ K_I \;\to\; H^p(-, L) \circ K_I.$$

The next step is to translate these maps between persistence modules to matchings of the corresponding barcodes. Best would be the construction of a functor from the category of (pointwise finite-dimensional) persistence modules $\mathbf{Fun}(\mathbf{I}, \mathbf{Vect}_{\mathbb{F}})$ to **Mch**, mapping persistence modules to barcodes and morphisms thereof to matchings. Unfortunately, [3, Proposition 5.10] shows that no such functor exists. Only when restricting $\mathbf{Fun}(\mathbf{I}, \mathbf{Vect}_{\mathbb{F}})$ to the subcategory containing either only monomorphisms or epimorphisms is it possible to construct such a functor. We refer to the methods presented in [3, Section 4] describing the specifics of a canonical construction for either case. The above morphisms do not fall into either subcategory however. Nevertheless, a non-functorial matching may be constructed by factoring through the image, that is

$$H^p(-, L) \circ K_I \;\twoheadrightarrow\; \mathrm{im}(\rho) \;\hookrightarrow\; H^p \circ K_I \qquad \text{(similarly for } \delta_p^*\text{)}$$

and composing the matchings induced canonically by this epimorphism and monomorphism. For further details and results addressing the quality such matchings, we refer to [3].
In summary, the relation of persistent relative and absolute cohomology encoded in the long exact sequence of relative cohomology may be translated to two barcode correspondences by constructing induced matchings. Since the considered morphisms are neither monomorphisms nor epimorphisms, the construction is inevitably non-functorial and critically relies on the image of the persistence module morphisms in question. From a computational perspective, this requires tools to compute *image persistence*, a topic of ongoing research that is beyond the scope of this thesis.

We conclude this section with a heuristic that while primitive, provides a limited amount of insight into the similarity of the barcodes associated with persistent relative and absolute homology, assuming that the distances within the data set at hand are largely pairwise distinct: Searching for persistence intervals that identically appear in either barcode. Figure 3.6 shows the resulting "matchings" for the data set random100 in degree one, comparing the choices $r = 0.4$ and $r = 0.7$. As the fraction of data points used to induce the relative subcomplex grows, the number of matched intervals declines. Note also that persistence intervals in the barcode of $H^0(L \cap -) \circ K_I$ are never matched with those corresponding to $H^1(-, L) \circ K_I$ since the (cohomological) death index is always zero in the former case while never zero in the latter case.

3.5. Homology Representatives

Section 3.3 suggests that the most efficient way to obtain the barcode for a given data set is by computing persistent relative cohomology, that is to say reducing the coboundary matrix $D^\perp$ and using Corollary 2.79 and 2.83 to translate the barcode. For certain applications [6, 34] however, not only the barcode but also the interval module generators are of interest. The output specification of Problem 3.1 and 3.12 is formulated to include these generators in form of one representative for each (co)homology class for this reason. Note however, that an instance of either problem in the cohomological setting entails cohomology representatives and Algorithm 3.1 and 3.2 accordingly output a basis containing cocycles when applied to $D^\perp$ and $D^\uparrow$. If specifically homology generators are required, choosing the cohomological setting for efficiency reasons is at odds with the required output.
This section investigates three methods for the computation of homology representatives. In the following subsection we give the homological setting a second look and provide evidence that this simple approach to homology representatives is unlikely to be feasible for moderately large data set. We then review the current state-of-the-art method that has been used in [6] and is described in [16] using the performance analysis of the previous subsection to explain its efficiency. In the third subsection we present and evaluate an

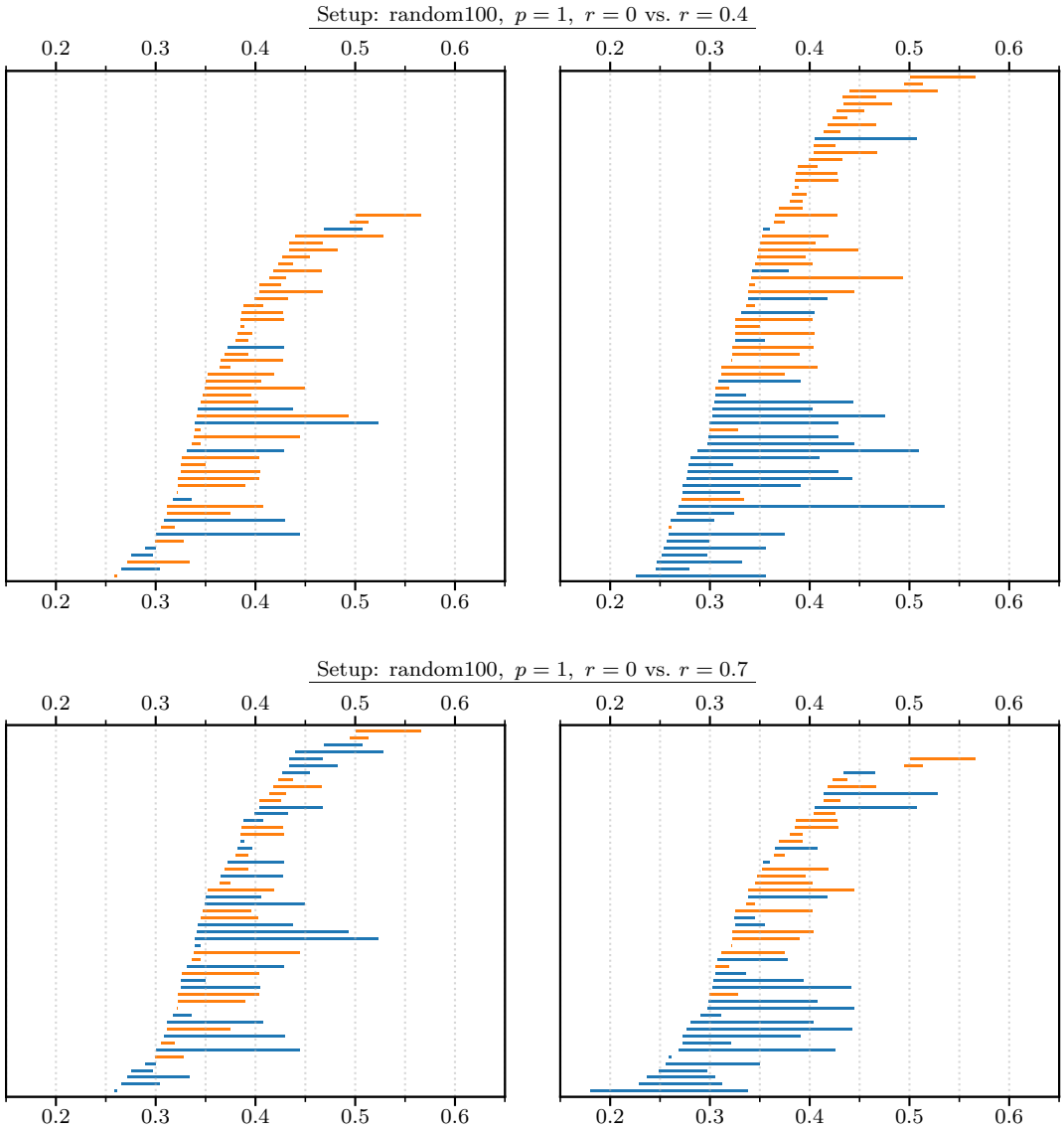


Figure 3.6.: Recall from Section 3.4.2 the experimental setup used to compute persistent (relative) cohomology. The four barcodes shown are the result of computing persistent absolute cohomology (left side) and persistent relative cohomology (right side) with respect to the indicated parameters, p denoting the degree and r specifying the fraction of points used to induce the relative subcomplex. The bars highlighted in orange occur identically in the barcode to the left and right, providing a primitive measure of similarity.

alternative method developed and implemented as part of this thesis project that is based on Proposition 2.81.

3.5.1. Homology Reduction

Since Algorithm 3.1 solves Problem 3.1 for the instance $M = H_\bullet \circ K_I$ directly, the most natural approach to computing homology generators is the application of the reduction algorithm in the homological setting. In this case, the cost directly associated with the computation of representatives is near zero since they arise as columns of the matrices R and V and merely need to be stored. The overall efficiency of this approach is thus solely determined by that of the matrix reduction itself.
The discussion in Subsection 3.3.2 and 3.3.6 suggests that the reduction algorithm performs better in the cohomological setting compared to the homological if clearing is applied. This is due to the lack thereof in the top degree in the homology setting. In the following, we examine empirically to which extent the computational efficiency differs in an attempt to answer if the homology reduction can possibly be a viable approach for the computation of homology representatives. To this end, we compare four implementations of the reduction algorithm using the following experimental setup.

- All versions implement a degreewise reduction with chains over $\mathbb{F}_2$. Our implementation is based on Ripser v.1.2.1, using clearing (in all applicable degrees) as well as the implicit representation of the (co)boundary matrix and R. The optimizations described in Subsection 3.3.4 and 3.3.5 are not implemented.
- The difference between the four versions is characterized by two binary properties: The setting (homological versus cohomological) and optimizations (no further optimizations versus emergent and apparent pairs). The corresponding source files [37] are
  `ripser_hom_clearing.cpp`,
  `ripser_hom_clearing_optimized.cpp`,
  `ripser_cohom_clearing.cpp` and
  `ripser_cohom_clearing_optimized.cpp`.
- The indicators of performance are those described in Subsection 3.3.7. As in Subsection 3.4.2, we display values relative to those of a fully optimized homology reduction. In other words, all values are multiples of those corresponding to a reference, namely the optimized homology reduction.

The results of our performance analysis based on our introductory data set of 45 points in $\mathbb{R}^2$ (visualized in Figure 1.1) are presented in Figure 3.7. The two plots on the left side show that matrix reduction in the homological setting is more efficient with respect to all considered performance indicators except for "add" and "adds" (denoting the number of calls to `add_[co]boundary` and `add_simplex_[co]boundary`, respectively). For these two instances, p is not the top degree and clearing is applicable in both the homological and cohomological setting. The striking efficiency of the matrix reduction in the homological setting crucially depends on clearing however, as illustrated by the two plots on the right side. There, the setup specifies $p = 2$ and $p = 3$ as the top degree (the upper and lower

right plot, respectively) which precludes clearing in the homological setting. The observed performance impact is substantial: In degree two, the elapsed time for the full optimized version of the cohomology reduction is now improved by a factor of approximately 0.091 compared to the fully optimized homology reduction. In degree three this factor is reduced even further to approximately 0.054. All other performance metrics behave similarly.
This not only provides an example for the effectiveness of clearing but also shows that computing persistent homology in the top degree, even if fully optimized, performs considerably worse than the computation of persistent cohomology. This is compounded by the fact that the computation of persistent homology requires an additional degree in order to match the output of persistent cohomology due to the structural difference between the two approaches. In summary, the computation of persistent homology is almost certainly not a viable approach for the computation of homology representatives if the size of the data set exceeds a few hundred points.

3.5.2. Cohomology with Secondary Homology Reduction

Although the reduction of the coboundary matrix $D^\perp$ only provides cohomology representatives, it determines birth and death indices in homology and cohomology alike. This is a consequence of the duality results in Proposition 2.80 and Corollary 2.82:
All non-essential death indices identified by computing persistent relative cohomology are identically homological birth indices. Reducing the coboundary matrix thus imitates clearing in the homology setting by determining the non-essential homological birth indices that correspond to superfluous column reductions as described in Subsection 3.3.2. In particular, the reduction of $(D^\perp)^p$ may substitute the clearing optimization for the homology reduction including that in the top degree.
Somewhat surprisingly, the cost of applying the reduction algorithm twice, first fully optimized in the cohomological setting (in order to determine the homological birth indices) and then again in the homological setting (to compute homology representatives) is substantially faster compared to a single reduction in homology without clearing in the top degree.
This is illustrated by the left plots in Figure 3.7. As pointed out in the previous section, the matrix reduction in the homological setting has substantially improved performance compared to that in cohomology if clearing is applicable. This is can be explained by comparing the number of non-zero entries contained in the columns of the boundary and coboundary matrix: If the Vietoris-Rips filtration is not cut off and thus ends in the full simplex on the set of vertices, all columns of D^p have exactly $p+1$ non-zero entries, noticeably less than the columns in $(D^\perp)^p$ containing $n-p$ such entries (where n is the number of vertices).
For a detailed description of this approach and further performance analyses we refer to [16].

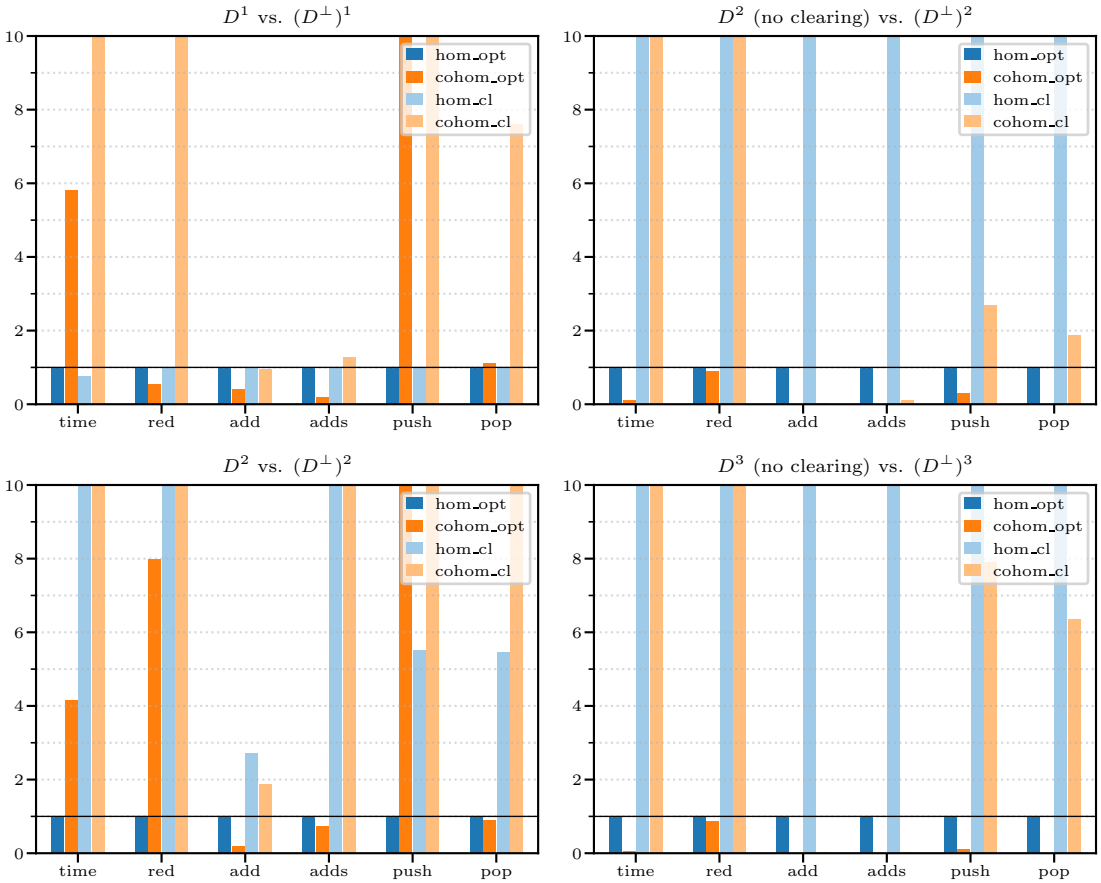


Figure 3.7.: Our performance analysis comparing the efficiency of the matrix reduction in the homological (blue) and cohomological (orange) setting. We distinguish between versions that only use clearing (light colors) and those that also use the emergent and apparent pairs shortcut (dark colors). The data set is our introductory example of 45 points in $\mathbb{R}^2$ (visualized in Figure 1.1) and the performance indicators are those presented in Subsection 3.3.7. All values are factors left-most bar. Those exceeding 10 are cut off.

3.5.3. Computation of the Dual Basis

The duality of persistent homology and cohomology established in Section 2.9 may be used to derive homology representatives from a compatible basis $\mathcal{T}$ of $C^\bullet(K)$ (see Definition 2.77) that is computed by the reduction algorithm applied to the coboundary matrix $D^\perp$ (or $D^\uparrow$). By Corollary 2.82 its dual basis $\mathcal{T}^*$ is a compatible basis of $C_\bullet(K)$ that contains a set of homology generators by Theorem 2.53 . This subsection examines how this dual basis can be computed by matrix inversion and we then present an algorithm implementing this inversion as a modification of the reduction algorithm. Moreover, we provide details regarding our implementation based on Ripser followed by a performance

analysis.
The starting point is the following result from linear algebra that formulates the transition to the dual basis in terms of a change of basis transformation.

Proposition 3.23. Let V be a n-dimensional vector space, $\mathcal{A}$ and $\mathcal{B}$ be ordered bases and denote by $P \in \mathbb{F}^{n\times n}$ the change of basis transformation $\mathcal{A} \to \mathcal{B}$. Then the change of basis transformation $\mathcal{A}^* \to \mathcal{B}^*$ of dual bases is given by $(P^T)^{-1}$.

Proof. The matrix P represents the identity $\mathrm{id} : V \to V$ with respect to the bases $\mathcal{A}$ and $\mathcal{B}$. The dual map $\mathrm{id}^* : V^* \to V^*$ is thus represented by the transpose P^T with respect to bases $\mathcal{B}^*$ and $\mathcal{A}^*$ constituting the change of basis transformation $\mathcal{B}^* \to \mathcal{A}^*$. Inverting P^T yields the change of basis transformation $\mathcal{A}^* \to \mathcal{B}^*$ as desired. □

The columns of P are precisely the coordinate vectors of the basis elements in $\mathcal{A}$ relative to $\mathcal{B}$. Dually, the columns of $(P^T)^{-1}$ are the coordinate vectors of the basis elements in $\mathcal{A}^*$ with respect to $\mathcal{B}^*$. We apply this relation for our purposes as follows.
Recall from Section 2.9 that Proposition 2.67 and 2.68 are applicable in the setting of persistent relative cohomology by reducing the coboundary matrix $D^\perp$ representing the coboundary map with respect to $\mathcal{C}^*(K)$ in reverse filtration order. The resulting decomposition $R = D^\perp V$ yields a compatible basis $\mathcal{T}$ of $C^\bullet(K)$ by selecting certain columns of R and V. In particular, the exchange of all columns V_j satisfying $j \in \mathrm{pivots}(R) \setminus \{0\}$ with corresponding columns R_ℓ satisfying $j = \mathrm{pivot}(R_\ell)$, yields a change of basis transformation from $\mathcal{T}$ to $\mathcal{C}^*(K)$, in the following denoted by P. Applying Proposition 3.23, we obtain a change of basis transformation from $\mathcal{T}^*$ to $\mathcal{C}^{**}(K)$ that is canonically identified with $\mathcal{C}(K)$ (see Proposition 2.22). Note that the latter basis retains the order from $\mathcal{C}^*(K)$, namely the reverse filtration order. This is corrected for by composing with a second change of basis transformation from $\mathcal{C}(K)$ to itself, reversing its order and thereby yielding the dual basis of $\mathcal{T}$ as coordinate vectors with respect to $\mathcal{C}(K)$ in filtration order as desired. We summarize this in the following proposition.

Proposition 3.24. Let P be the change of basis matrix as described above. Then the columns of

$$I^\perp \left(P^T\right)^{-1} \quad \text{where} \quad I^\perp = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix}$$

are a compatible basis of $C_\bullet(K)$. □

As a compatible basis of $C_\bullet(K)$, the columns of $I^\perp(P^T)^{-1}$ contain unbounded (essential) cycles, bounded (non-essential) cycles and the chains bounding them. Proposition 2.80 instructs on how these roles are assigned by taking the dual basis: Basis elements corresponding to unbounded cocycles in $\mathcal{T}$ are again unbounded cycles in $\mathcal{T}^*$. For non-essential pairs the relation is switched however. Dual basis elements of bounding cochains are now bounded cycles whereas dual basis elements of bounded cocycles are

bounding chains in $\mathcal{T}^*$. In other words, the relevant column indices of $I^\perp(P^T)^{-1}$ are those that correspond to both essential and non-essential birth indices in persistent relative cohomology (that is computed by reducing the coboundary matrix). From a practical perspective, this approach may be realized as follows.

1. Apply the reduction algorithm in the cohomological setting, reducing $D^\perp$, in order to compute a compatible basis of $C^\bullet(K)$.
2. Assemble those columns required for the change of basis transformation P.
3. Invert P^T and reverse the row order to account for the inverted basis order.
4. Select those columns that correspond to essential and non-essential cohomological birth indices (identified by the reduction of $D^\perp$ in (1)).

The first step is efficiently solved by the reduction algorithm as discussed in Section 3.3. Regarding the inversion of P^T, note that i-th column of the inverse $Q = (P^T)^{-1}$ is the solution of

$$P^T Q_i = e_i, \qquad i \in \{1, \dots, |K|\}$$

where e_i denotes the i-th unit vector and $i \in \{1, \dots, |K|\}$. Since only a subset of columns of Q contains homology representatives, solving this system separately for all relevant indices $i \in I$ improves the efficiency over the inversion of P^T as a whole. Using the lower-triangular structure of P^T, forward substitution (see for instance [29, p. 30]) can be applied yielding the recursion formula

$$Q_{ji} \;=\; \begin{cases} 0 & \text{if } i > j, \\ (P^T)_{ii}^{-1} & \text{if } i = j, \\ -(P^T)_{jj}^{-1} \sum_{k=i}^{j-1} (P^T)_{jk} Q_{ki} & \text{if } i < j \leq |K|. \end{cases}$$

In words, the first non-zero entry of Q_i is the inverse of the diagonal element $(P^T)_{ii}$. To compute any subsequent entry at index j, we first take the inner product of the j-th row of P^T up to the diagonal entry and the first $j-1$ elements of Q_i (computed earlier). The sum implementing this inner product may start at $k = i$ instead of $k = 1$ since the entries of Q_{ki} are zero up to $k = i$. Then, the negated inverse of the diagonal element $(P^T)_{jj}$ is multiplied as a factor in order to obtain the j-th entry Q_{ji} of the column Q_i.

Remark 3.25. For coefficients in the field $\mathbb{F}_2$ the above formula simplifies to

$$Q_{ii} \;=\; 1 \quad \text{and} \quad Q_{ji} \;=\; \sum_{k=i}^{j-1} (P^T)_{jk} Q_{ki} \quad \text{for } i < j \leq |K|.$$

To compute a column Q_i we need access to the columns P_j for $j \in \{|K| - i, \dots, |K|\}$. Since P contains all non-zero columns of R it should not be assembled and stored in memory. Instead, the strategy of Subsection 3.3.1 can be applied, assembling the required

columns of R (as part of P) by recomputing them as the matrix vector of product $D^{\perp}V$. This comes at a computational cost however, which we aim to minimize by structuring our algorithm in a columnwise fashion with respect to P. This means that the columns of Q are assembled in parallel, one row index at a time as we progress through the columns of P one by one. Moreover, this allows us to interleave the matrix reduction of $D^{\perp}$ and the update of the Q by means of the above recursion formula. Algorithm 3.3 realizes this approach using implicit matrix reduction. The column P_j is assigned in Line 21 or recomputed in Line 24. Lines 25–27 then implement the update of the inverse Q as described above.

Algorithm 3.3: The reduction algorithm implementing implicit matrix representation and clearing, modified to compute $(P^T)^{-1}$.

```
1  J := ∅;  V := id;  Q := 0
2  pivot-index(−) := −1
3  for j := 1 to |K| do
4      if pivot-index(j) = 0 then
5          R_j := D_j^↑
6          ℓ = pivot(R_j)
7          while ℓ ≠ 0 do
8              if i := pivot-index(ℓ) ≠ −1 then
9                  R_i = D^↑ · V_i
10                 λ := R_ℓj / R_ℓi
11                 R_j := R_j − λR_i
12                 V_j := V_j − λV_i
13                 ℓ := pivot(R_j)
14             else
15                 pivot-index(ℓ) = j
16                 break
17         if ℓ ≠ 0 then
18             J := J ∪ [ℓ, j)
19         else
20             J := J ∪ [j, ∞)
21         P_j := V_j
22     else
23         ℓ := pivot-index(j)
24         P_j := D^↑ · V_ℓ
25     Q_jj := P_jj^{−1}
26     for i := 1 to j − 1 do
27         Q_ji := −P_jj^{−1} Σ_{k=1}^{j−1} P_kj Q_ki
```

This approach was implemented as part of this thesis project based on Ripser v1.2.1 in `ripser_cohom_repinverse.cpp` [37]. We conclude this section with an outline of the notable aspects to this implementation and an evaluation of our approach by a performance analysis.

- Due to the degreewise implementation of the cohomology reduction, the (columnwise) assembly of P in degree p requires V^{p-1} (if $p > 0$) in addition to V^p, in order to recompute the columns of R^{p-1} that replace columns in V^p to form the compatible basis that comprises the columns of P.
- Similarly, it is required to keep track of column indices that correspond to non-essential birth indices and are thus subject to clearing. This index set is required to determine which columns of V^p have to be replaced by columns of R^{p-1}. This information is not available as part of the reduction in degree p but implicitly given by the matrix V^{p-1} if it were fully assembled. It is not and we therefore use a `std::vector` instead.
- If the apparent pairs shortcut is to be used for the cohomology reduction, column indices corresponding to such pairs have to be treated separately since Ripser omits such indices from the reduction loop entirely.
- The recompuation of P_j as the product $D^{\uparrow}V_\ell$ in line of Algorithm 3.3 is implemented in fashion of the recomputation of R_i in Line 9. Note, that this requires a number of calls to `push`.
- Computing the inner product of Line 27 is implemented by iterating through the `std::priority_queue` that encodes the column P_j with decreasing row index. For every retrieved element P_{kj} we check if the partially assembled column Q_i contains that same element. If this is the case, the coefficients are multiplied. Adding all such products and multiplying with $-P_{jj}^{-1}$ yields the entry of Q_i. Associated with each retrieval of P_{kj} is a call to `pop`.

We evaluate our approach with a performance analysis similar to that of Subsection 3.5.1, comparing its computational efficiency with that of the state-of-the-art method described in the previous subsection and implemented as part of this thesis project (based on Ripser v.1.2.1) in `ripser_cohom_rephom.cpp` [37]. The setup for this analysis is described as follows.

- Only the computational cost directly associated with the computation of homology representatives is compared. For our approach this means the cost of Lines 21, 24 and 25–27 of Algorithm 3.3, while for the state-of-the-art approach this comprises the cost of the secondary homology reduction.
- In addition to the runtime performance we consider the number of calls to the function `add_simplex_coboundary` (in Figure 3.8 denoted by "adds") as well as the number of calls to `push` and `pop`.

- Similar to the evaluation format of Subsection 3.4.2 and 3.5.1, the shown values are relative to a reference, here the state-of-the-art approach. As such, all values a factors, cut off above 25.

The results of our analysis for the data sets random100 and covid10000 (introduced in Subsection 3.4.2) are shown in Figure 3.8, imitating the presentation of Figure 3.7. In all instances, our approach performs substantially worse than the state-of-the-art approach. It is clearly visible that the matrix inversion requires considerably more elementary operations. One possible explanation for the poor performance lies in the columns of R that are part of P. These columns often contain orders of magnitude more elements than the columns originating in V and necessitate a much larger number of elementary operations in order to be assembled and processed during the computation of the inner product.
Without considerable algorithmic improvements the computation of homology representatives by a dual basis calculation is likely to fall short of the current state-of-the-art method for most data sets. Beside the conceptional appeal, the calculation of the dual basis may find applications in future work however, as it provides a different set of homology representatives compared to those produced by a homology reduction. Figure 3.9 visualizes a set of representatives obtained by dual basis computation in degree one for our introductory example, highlighting this difference.

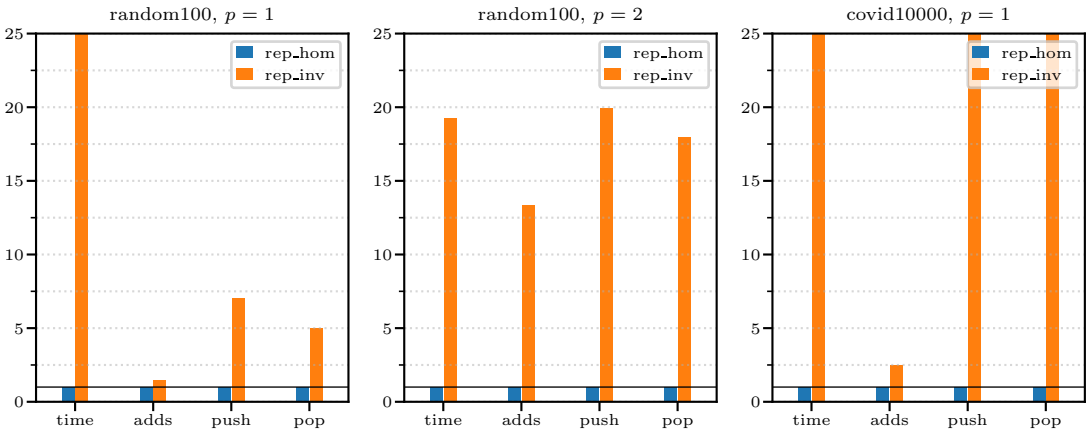


Figure 3.8.: Our performance analysis comparing the efficiency of two approaches for the computation of homology representatives based on the performance indicators described in Subsection 3.3.7 and the random100 data set. In blue, the values obtained for the state-of-the-art approach based on a secondary homology reduction, serving as a reference. In orange, our approach based on a dual basis computation via matrix inversion. All values are relative to the former version and are cut off if a factor of 25 is exceeded.

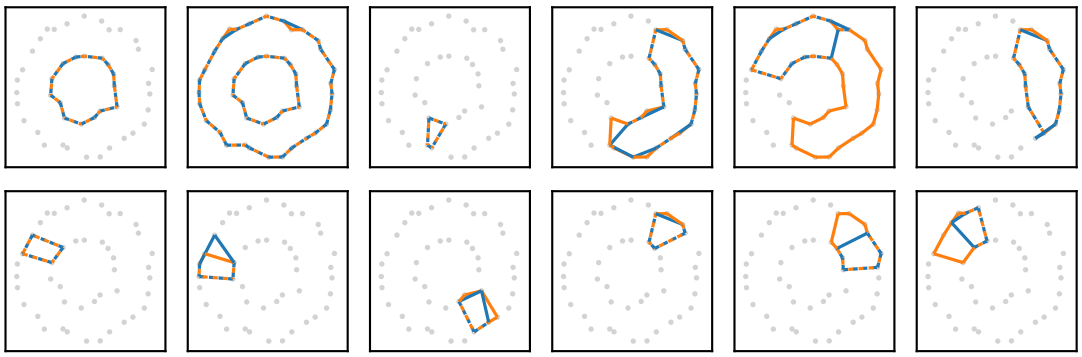


Figure 3.9.: Two sets of homology representatives in degree one, computed for the data set shown in Figure 1.1. In blue, the set of representatives computed by reducing the boundary matrix. In orange, a second, distinct set of homology representatives, obtained by our approach based on the computation of a dual basis.

4. Conclusion and Outlook

Persistent homology is a relatively new subject of research at the intersection of algebraic topology and topological data analysis with first publications appearing in the early 1990s [11, p. 2]. Computational aspects followed in the 2000s with influential contributions such as [20] and [39], while persistent homology gained traction as a method for data analysis. Since then, a large body of swiftly advancing research has formed around the general topic of persistence and more specifically the computation of persistent homology.

In the first part of this thesis, we provide a streamlined and relatively self-contained exposition of the theoretical framework underlying persistent homology as a data analysis method in an effort to consolidate notation and results. After a review of the key definitions and results concerning simplicial homology, filtrations and persistence modules in Section 2.1–2.6, we examine persistent homology in Section 2.7 using the notion of a compatible basis as the central construct. Based on a similar concept in [17] and [2], we describe how a compatible basis brings about the barcode decomposition and thereafter show how such a basis arises from the reduction of the boundary matrix. Following [17], we then review variations of persistent homology in Section 2.8 and 2.9, reiterating the duality results that play an important role for the efficient implementation of persistent homology in the software Ripser [2] and giotto-ph [31].

The computational aspects of persistent homology are the topic of the second part of this thesis. We begin by formulating persistent homology as a computational problem, describe the required preliminary setup for the computation following [2] and examine how the reduction algorithm solves the computational problem for all problem instances. In Section 3.3, we begin with the topic of computational efficiency and review the optimizations responsible for the fast implementations currently available using the software Ripser as a reference. We also introduce a number of performance metrics specific to Ripser that in addition to the commonly used runtime measurements are part of the performance analysis in the following sections.

Motivated by the analysis of the large SARS-CoV-2 virus genome data set presented in [6], Section 3.4 investigates persistent relative homology as an alternative to persistent absolute homology promising improved performance by reducing the size of the (co)boundary matrix. After a short theoretic justification, we summarize the modifications of Ripser required for our implementation of persistent relative cohomology. We then present our performance analysis that compares implementations of persistent absolute and relative cohomology with respect to the aforementioned performance indicators showing the results for two data set: random100, containing 100 points sampled uniformly from $[0, 1]^4$ and covid10000, a data set similar to the one analyzed in [6]. Our analysis confirms the expectation that the computation of persistent relative cohomology with sufficiently large relative subcomplex improves the runtime compared to the state-of-the-art implementation of

persistent (absolute) cohomology. Evaluating and interpreting the remaining performance indicators is less clear-cut. We observe for instance a substantial increase in the number of column additions, contrary to the decrease of required column reductions and the improved runtime. With respect to the memory consumption, this may cause issues for truly large data sets. Considering the results shown in the Appendix, the computation of persistent relative homology seems to be highly dependent on the data set at hand and the relative subcomplex chosen. Subsection 3.4.3 concludes the topic of persistent relative homology with a short introduction to the topic of induced matchings following [3] outlining how the barcode of persistent relative and absolute cohomology can be related. However, details and computation are beyond the scope of this thesis and we instead present a simple matching heuristic in order to gain a sense of barcode similarity.
The final section in this second part compares three approaches to the efficient computation of homology representatives. First, the simple yet inherently inefficient approach of a (optimized) homology reduction is considered and evaluated. This performance analysis also provides an explanation for the striking efficiency of the second approach examined, the current state-of-the-art method based on a secondary homology reduction after performing a reduction of the coboundary matrix. The third approach, developed as part of this thesis, is based on the duality of homology and cohomology. Homology representatives are obtained from a compatible basis in the cohomological setting by computing its dual basis via matrix inversion. Although we provide an efficient algorithm and implementation thereof, obstructions seemingly inherent to the matrix inversion have a considerable negative impact on the computational efficiency preventing our approach from surpassing the established method mentioned above.

As part of this thesis project, we developed the tools that made the performance analysis of Sections 3.4 and 3.5 possible. This includes the specification of program parameters through configuration files, utility to gather performance data at a granular level and output generation, all implemented as additional features for Ripser. All source code files, also including evaluation and visualization scripts, configuration files and many data sets are available as a fork of Ripser at https://github.com/skreisel-tum/ripser/tree/thesis-project.

We conclude with a list of open questions and directions for future work.

- Lemma 2.51 lies at the core of obtaining the barcode decomposition from a compatible basis $\mathcal{S}$, describing how such a basis develops throughout an essential simplexwise filtration. Its proof does not need the condition $\partial_\bullet(s_i) = s_\ell$ for non-essential pairs (s_ℓ, s_i) however. Instead it only requires that the restriction $\mathcal{S}(K_i)$ contains a linear combination (containing s_i as a summand) bounding s_ℓ. It follows that this weaker condition also suffices for Theorem 2.53. We wonder if this slight generalization of [17] can be carried over to the related theory or is rather an obstacle for results such as the correspondence of relative and absolute (co)homology.

- In Lemma 2.51 we call the index ℓ of a non-essential pair (s_ℓ, s_i) the pivot of index of s_i, borrowing terminology usually used in the context of matrix reduction. In Subsection 2.7.2, we then identify it with the pivot index of a matrix column and in this setting prove its uniqueness in Proposition 2.62. This uniqueness proof should be transferable to the more abstraction setting of compatible bases.
- As mentioned above, our performance analysis of persistent relative cohomology displays unintuitive characteristics, in particular with respect to the number of required column additions and the number of calls to `push` and `pop`. Analyzing the performance metrics for separate column reductions could provide better insights into this matter. Our tools allow the collection of such granular data but a proper evaluation thereof was beyond the scope of this thesis.
- Repeating the performance analysis of our implementation of persistent relative cohomology for larger data sets on server-grade hardware should allow for a more definite verdict on the viability of this method as a substitute for persistent absolute homology.
- Relating the barcodes obtained by computing persistent relative and absolute homology is only briefly touched upon in Subsection 3.4.3. There are ongoing efforts to implement image persistence (see for instance [5]) for morphisms of persistence modules that are induced by an inclusion at the chain complex level. This approach may be generalized or adapted to fit the setting of persistent relative homology.
- Our approach to homology representatives, that is the computation of a dual basis, turned out to be less efficient than a secondary homology reduction. We wonder what underlying cause of this disparity is and whether the matrix inversion has additional structure that could be used to alleviate it.

A. Appendix

A.1. Algorithms

The two algorithms below are variations of Algorithm 3.1 implementing the optimizations of Subsection 3.3.1 and 3.3.2 separately. Algorithm 3.2 presented in Subsection 3.3.2 combines both optimizations.

Algorithm A.1: The reduction algorithm modified to use implicit matrix reduction

```
1  J := ∅;  E := ∅;  B := ∅
2  V := id
3  pivot-column-index(−) := −1
4  for j := 1 to |K| do
5      R_j := D_j^↓
6      ℓ = pivot(R_j)
7      while ℓ ≠ 0 do
8          if i := pivot-column-index(ℓ) ≠ −1 then
9              R_i = D^↓ · V_i
10             λ := R_ℓj / R_ℓi
11             R_j := R_j − λR_i
12             V_j := V_j − λV_i
13             ℓ = pivot(R_j)
14         else
15             pivot-column-index(ℓ) := j
16             break
17     if ℓ = 0 then
18         J := J ∪ [j, ∞)
19         E := E ∪ V_j
20     else
21         J := J ∪ [ℓ, j) \ [ℓ, ∞)
22         B := B ∪ R_j
23         E := E \ V_ℓ
```

Algorithm A.2: The reduction algorithm modified to use clearing

```
1  J := ∅;  E := ∅;  B := ∅
2  R := D;  V := id
3  pivot-column-index(−) := −1
4  for j := 1 to |K| do
5      if pivot-column-index(j) = −1 then
6          R_j := D_j^↓
7          while there exists k < j such that ℓ := pivot(R_k) = pivot(R_j) do
8              λ := R_{ℓj}/R_{ℓk}
9              R_j := R_j − λR_k
10             V_j := V_j − λV_k
11         ℓ := pivot(R_j)
12         if ℓ ≠ 0 then
13             J := J ∪ [ℓ, j)
14             B := B ∪ R_j
15             pivot-column-index(ℓ) := j
16         else
17             J := J ∪ [j, ∞)
18             E := E ∪ V_j
```

A.2. Performance Analysis: Further Results

In this section we present additional results that were obtain as part of our performance analysis of the computation of persistent relative homology. The computational setup is that of Subsection 3.4.2 and all figures shown below have the same format as Figure 3.4 and 3.5. Before presenting the results we briefly motivate their inclusion.

- We consider a 2500 and 5000 point prefix of the covid10000 data set in order to investigate if this substantially changes the performance characteristics.
- For the data sets random100 we add the results for $p = 0$ and $p = 2$ to provide an example of these two degrees omitted in most other instances.
- We apply our methods to the data sets *sphere192* and *dragon1000*. The former is a random sample of 192 points on the unit sphere in $\mathbb{R}^3$ used in [2]. The latter is a random sample of 1000 points on the well-known Stanford Dragon used in [30] and obtained from [33].
- To include a second biological data set we provide the performance results for *hiv1088*, a distance matrix of 1088 distinct HIV virus genome sequence. This data set is analyzed in [9] and also appears in [30]. We obtained it from [33].

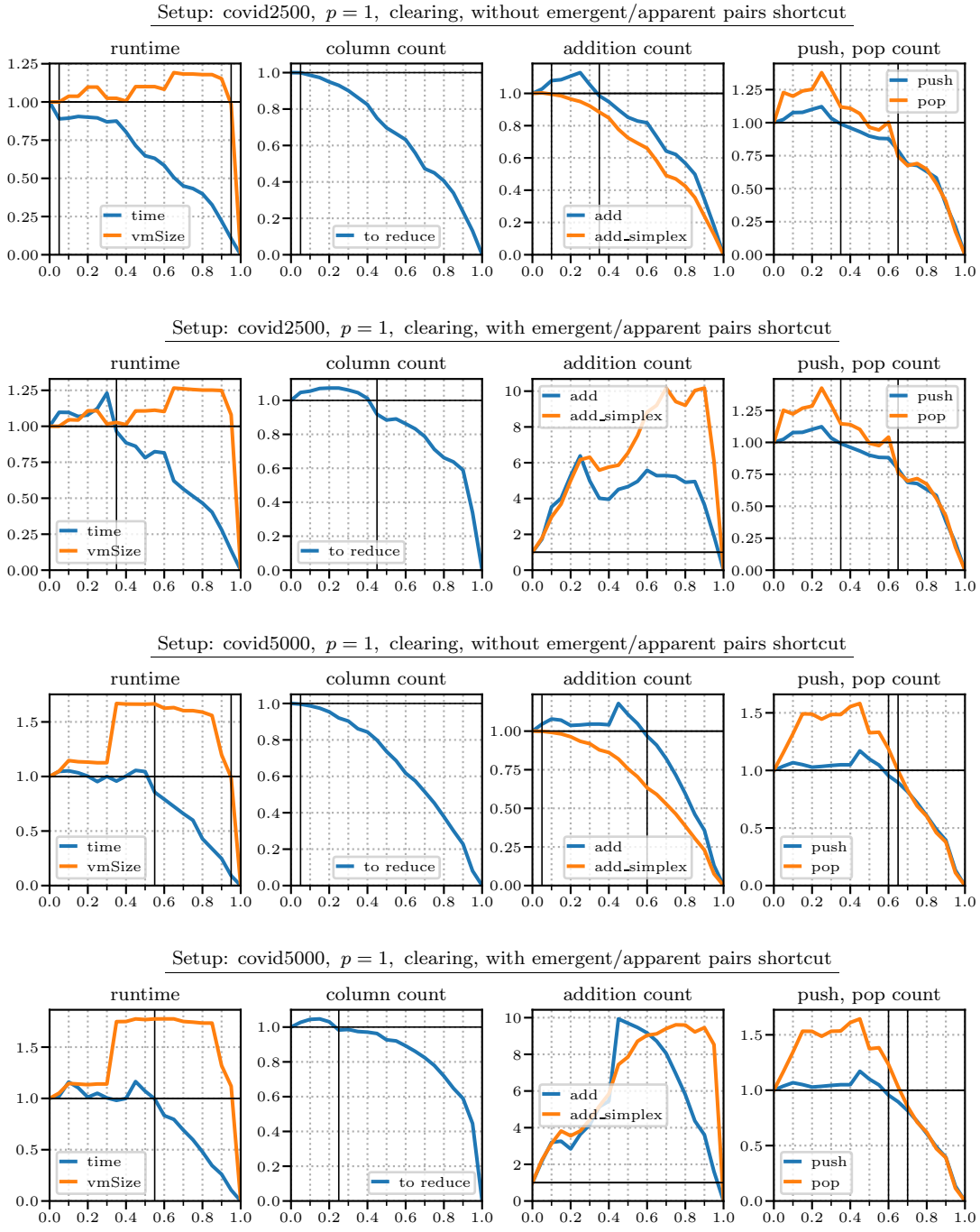


Figure A.1.: The performance analysis for a 2500 and 5000 point prefix for the covid10000 data set in degree one using the diameter threshold $t = 2$.

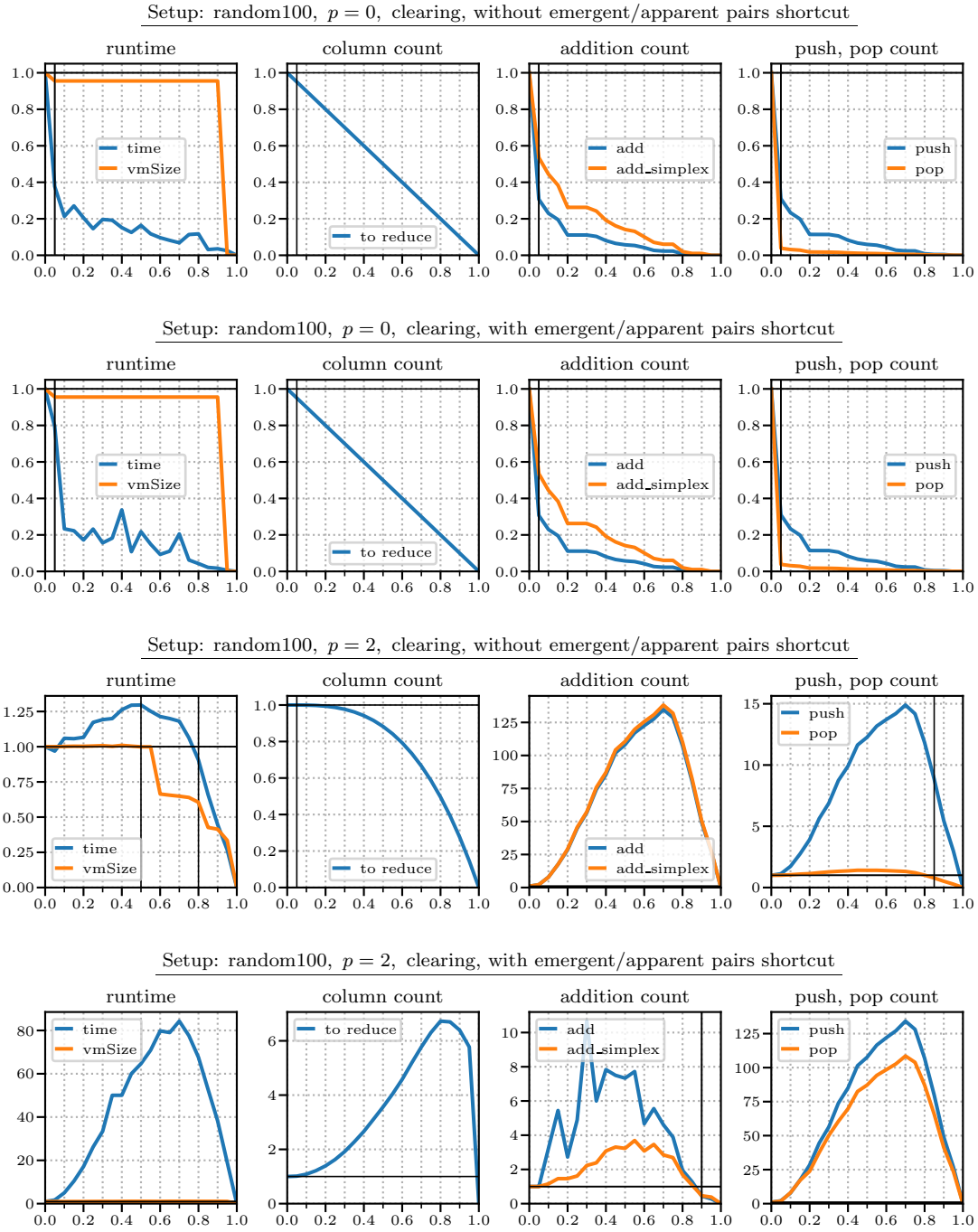


Figure A.2.: The performance analysis for the random100 data set in degrees zero and two. The results for degree one are shown in Figure 3.4.

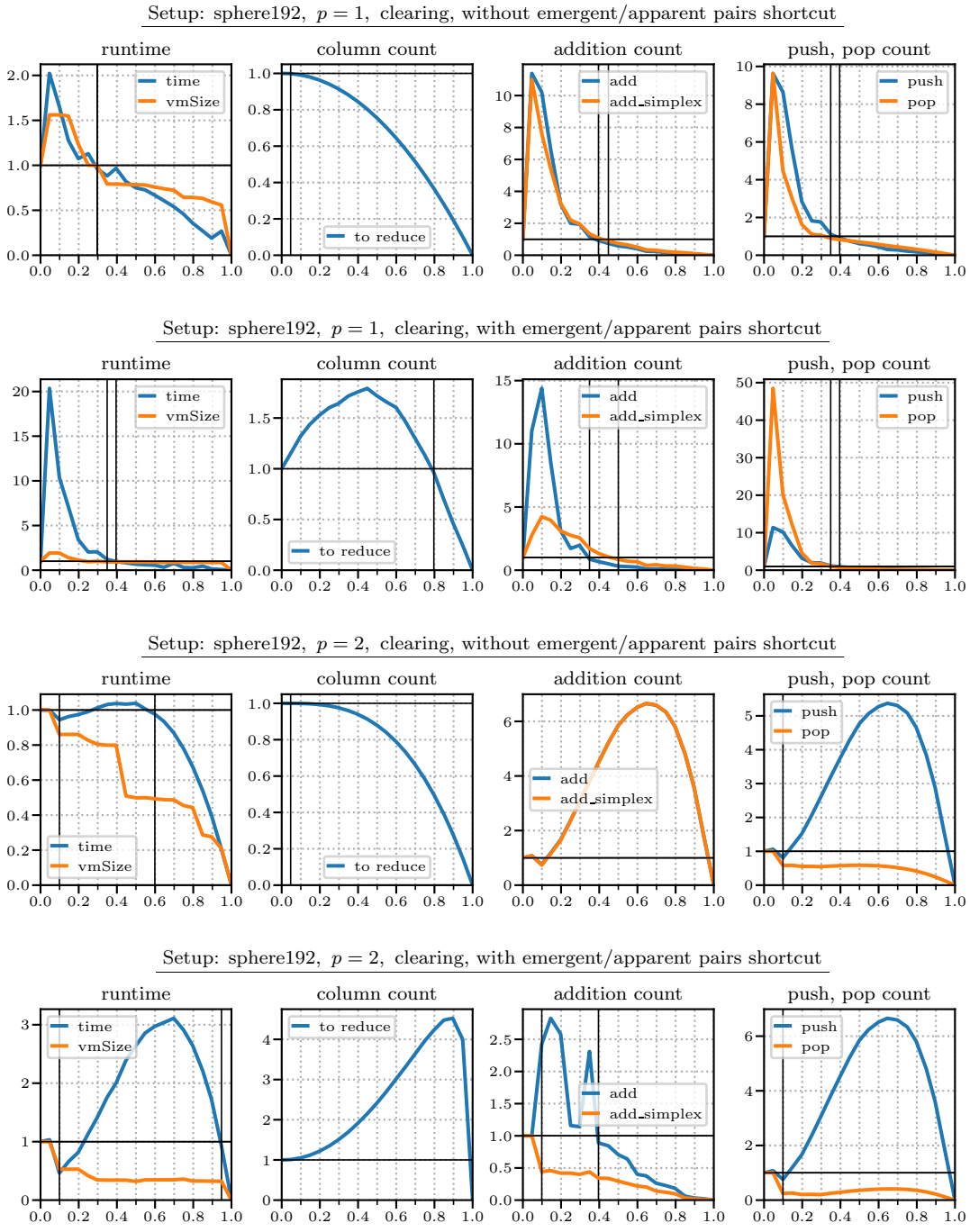


Figure A.3.: The performance analysis for the sphere192 data set in degrees one and two.

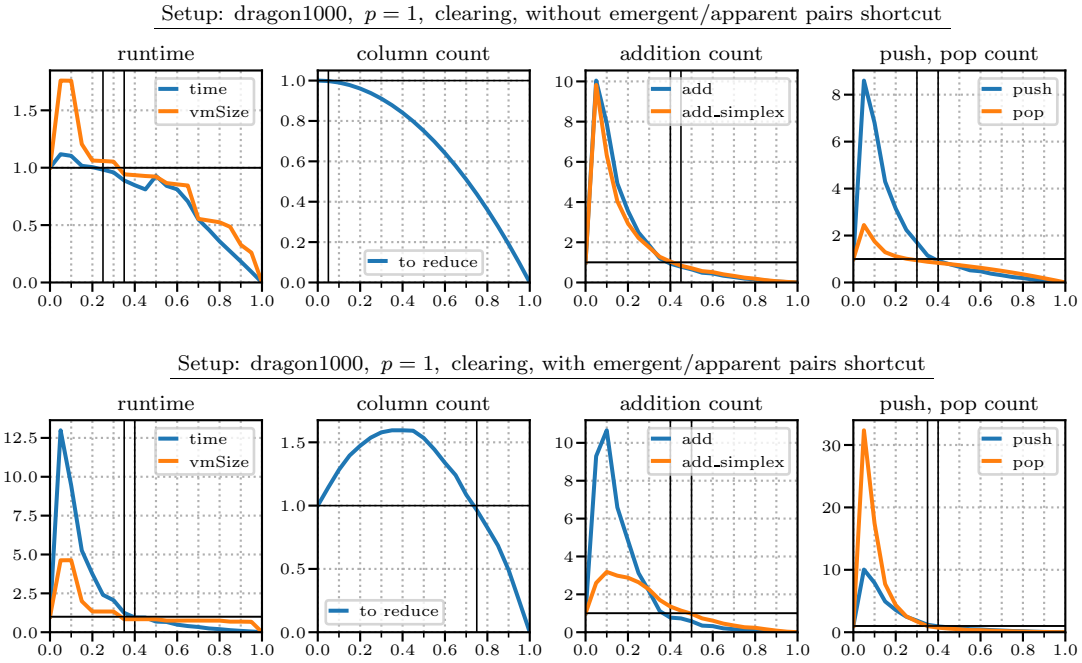


Figure A.4.: The performance analysis for the dragon1000 data set in degree one.

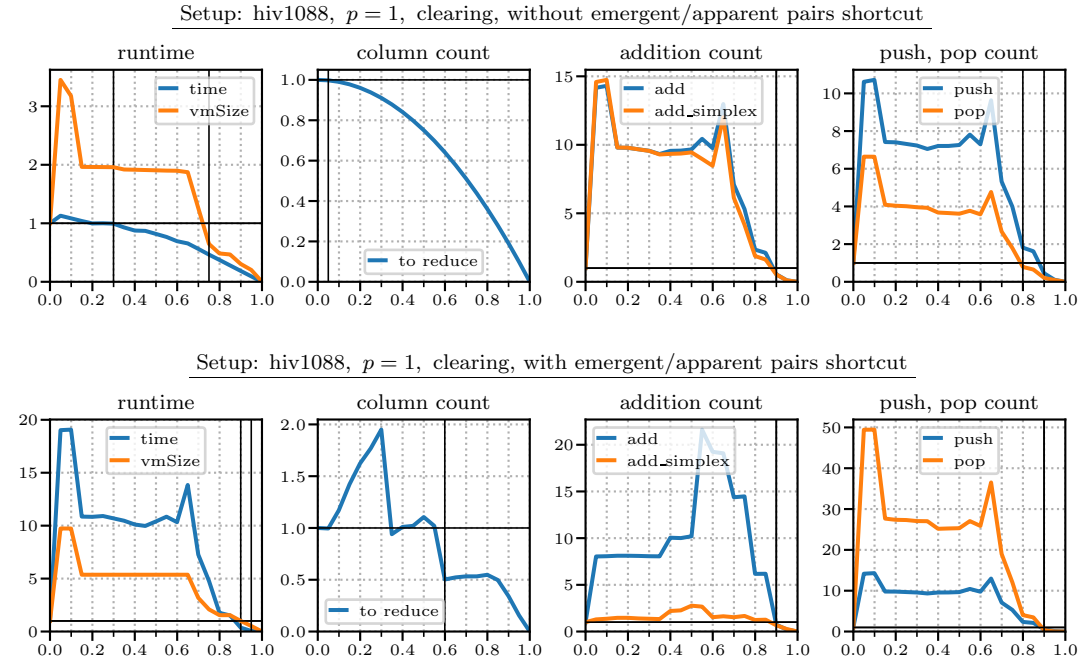


Figure A.5.: The performance analysis for the hiv1088 data set in degree one.

Acknowledgments

I would like to thank Prof. Dr. Ulrich Bauer for giving me the opportunity to work on a topic that combines my interests in algebra and computational methods in mathematics. The helpful advice and guidence I received were essential for this thesis project.

The analysis of the SARS-CoV-2 virus genome data would not have been possible without the work of Michael Bleher, Lukas Hahn and Andreas Ott, providing me with the distance matrix of the aligned sequences.

I am very grateful to Paul, Vinzenz, Valentin and my mum for their time spent proofreading and to Jenny and my dad for their support. Thank you!

Bibliography

[1] G. Azumaya. "Corrections and Supplementaries to My Paper concerning Krull-Remak-Schmidt's Theorem." In: *Nagoya Mathematical Journal* 1 1950, pp. 117–124.

[2] U. Bauer. "Ripser: efficient computation of Vietoris–Rips persistence barcodes." In: *Journal of Applied and Computational Topology* 5 2021, pp. 391–423.

[3] U. Bauer and M. Lesnick. "Induced Matchings of Barcodes and the Algebraic Stability of Persistence." In: *Proceedings of the 30th Annual Symposium on Computational Geometry*. 2014, pp. 355–364.

[4] U. Bauer and F. Roll. "Gromov hyperbolicity, geodesic defect, and apparent pairs in Vietoris-Rips filtrations." 2021. arXiv: 2112.06781 [math.AT].

[5] U. Bauer and M. Schmahl. "Efficient Computation of Image Persistence." 2022. arXiv: 2201.04170 [math.AT].

[6] M. Bleher, L. Hahn, J. A. Patino-Galindo, M. Carriere, U. Bauer, R. Rabadan, and A. Ott. "Topological data analysis identifies emerging adaptive mutations in SARS-CoV-2." 2022. arXiv: 2106.07292 [q-bio.PE].

[7] P. Bubenik and J. A. Scott. "Categorification of persistent homology." In: *Discrete & Computational Geometry* 51.3 2014, pp. 600–627.

[8] Z. Cang, L. Mu, K. Wu, K. Opron, K. Xia, and G.-W. Wei. "A topological approach for protein classification." In: *Computational and Mathematical Biophysics* 3.1 2015.

[9] J. M. Chan, G. Carlsson, and R. Rabadan. "Topology of viral evolution." In: *Proceedings of the National Academy of Sciences* 110.46 2013, pp. 18566–18571.

[10] F. Chazal, W. Crawley-Boevey, and V. De Silva. "The observable structure of persistence modules." In: *Homology, Homotopy and Applications* 18.2 2016, pp. 247–265.

[11] F. Chazal, V. De Silva, M. Glisse, and S. Oudot. "The Structure and Stability of Persistence Modules." Springer, 2016.

[12] C. Chen and M. Kerber. "Persistent homology computation with a twist." In: *Proceedings 27th European Workshop on Computational Geometry*. 2011, pp. 197–200.

[13] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. "Stability of persistence diagrams." In: *Discrete & computational geometry* 37.1 2007, pp. 103–120.

[14] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. "Vines and Vineyards by Updating Persistence in Linear Time." In: *Proceedings of the 22nd Annual Symposium on Computational Geometry*. 2006, pp. 119–126.

[15] W. Crawley-Boevey. "Decomposition of pointwise finite-dimensional persistence modules." In: *Journal of Algebra and its Applications* 14.05 2015.

[16] M. Čufar and Ž. Virk. "Fast computation of persistent homology representatives with involuted persistent homology." 2021. arXiv: 2105.03629 [math.AT].

[17] V. De Silva, D. Morozov, and M. Vejdemo-Johansson. "Dualities in persistent (co) homology." In: *Inverse Problems* 27.12 2011, p. 124003.

[18] V. De Silva, D. Morozov, and M. Vejdemo-Johansson. "Persistent cohomology and circular coordinates." In: *Discrete & Computational Geometry* 45.4 2011, pp. 737–759.

[19] H. Edelsbrunner and J. Harer. "Computational topology: an introduction." American Mathematical Society, 2010.

[20] H. Edelsbrunner, D. Letscher, and A. Zomorodian. "Topological persistence and simplification." In: *Discrete & Computational Geometry* 28.4 2002, pp. 511–533.

[21] R. J. Gardner, E. Hermansen, M. Pachitariu, Y. Burak, N. A. Baas, B. A. Dunn, M.-B. Moser, and E. I. Moser. "Toroidal topology of population activity in grid cells." In: *Nature* 2022, pp. 1–6.

[22] A. Garin and G. Tauzin. "A Topological "Reading" Lesson: Classification of MNIST using TDA." In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE. 2019, pp. 1551–1556.

[23] GISAID's public webpage. URL: https://www.gisaid.org/ (visited on 03/01/2022).

[24] M. Kashiwara and P. Schapira. "Categories and sheaves." Grundlehren der mathematischen Wissenschaften. Springer, 2006.

[25] S. Lang. "Linear Algebra." Undergraduate Texts in Mathematics. Springer, 1987.

[26] T. Leinster. "Basic category theory." Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2014.

[27] D. Morozov and A. Nigmetov. "Towards lockfree persistent homology." In: *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. 2020, pp. 555–557.

[28] J. R. Munkres. "Elements of Algebraic Topology." CRC Press, 1984.

[29] P. J. Olver, C. Shakiban, and C. Shakiban. "Applied linear algebra." Springer, 2006.

[30] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. "A roadmap for the computation of persistent homology." In: *EPJ Data Science* 6 2017.

[31] J. B. Pérez, S. Hauke, U. Lupo, M. Caorsi, and A. Dassatti. "giotto-ph: A Python Library for High-Performance Computation of Persistent Homology of Vietoris-Rips Filtrations." 2021. arXiv: 2107.05412 [cs.CG].

[32] R. Pérez-Moraga, J. Forés-Martos, B. Suay-García, J.-L. Duval, A. Falcó, and J. Climent. "A COVID-19 Drug Repurposing Strategy through Quantitative Homological Similarities Using a Topological Data Analysis-Based Framework." In: *Pharmaceutics* 13.4 2021.

[33] PH-roadmap on github: Data sets and scripts. URL: https://github.com/n-otter/PH-roadmap (visited on 03/01/2022).

[34] F. T. Pokorny, K. Goldberg, and D. Kragic. "Topological trajectory clustering with relative persistent homology." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 16–23.

[35] R. Rabadán and A. J. Blumberg. "Topological data analysis for genomics and evolution: topology in biology." Cambridge University Press, 2019.

[36] Y. Shu and J. McCauley. "GISAID: Global initiative on sharing all influenza data–from vision to reality." In: *Eurosurveillance* 22.13 2017.

[37] This thesis project on github: Source code, configuration files, scripts and datasets. URL: https://github.com/skreisel-tum/ripser/tree/thesis-project (visited on 03/01/2022).

[38] C. A. Weibel. "An introduction to homological algebra." Cambridge University press, 1995.

[39] A. Zomorodian and G. Carlsson. "Computing persistent homology." In: *Discrete & Computational Geometry* 33.2 2005, pp. 249–274.