Computational methods for multi-parameter persistence

Fabian Lenzen

Technische Universität München
TUM School of Computation, Information and Technology

Computational methods for multi-parameter persistence

Fabian Lenzen

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Gregor Kemper

Prüfer*innen der Dissertation:

1. Prof. Dr. Ulrich Bauer
2. Prof. Dr. Michael Kerber (TU Graz)
3. Prof. Dr. Magnus Botnan (VU Amsterdam)

Die Dissertation wurde am 28.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 23.11.2023 angenommen.

Computational methods for multi-parameter persistence

PhD thesis

Fabian Lenzen

June 26, 2023

TU München

Supervisor: Prof. Dr. Ulrich Bauer

Contents

List of Algorithms

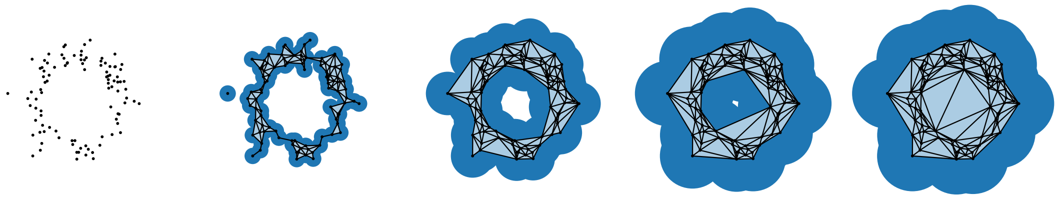
Chapter 1

Introduction

Topological data analysis [31, 66] is a mathematical field that seeks to apply methods from topology to analyze shape and topological features of data sets. *Persistent homology* [58, 59, 61, 64, 107, 108], a central and arguably one of its most popular tools, is a multi-scale approach that captures the changes of the homology of a filtered topological space along the filtration. This can be interpreted as how clusters, holes and higher dimensional voids appear and vanish during the filtration. The theory of persistent homology touches upon algebraic and computational topology, discrete Morse theory [6, 69], representation theory [108] and others.

A common pipeline for the application of persistent homology is the following [76]:

1. Starting from a point cloud $S \subseteq Y$ lying in a topological space Y, build a topological subspace of Y filtered by a scale parameter. For example, if (Y, d) is a metric space, the spaces $S^{(r)} := \bigcup_{s\in S} B_r(s)$ for $r \in \mathbf{R}$ (where $B_r(s) = \{y \in Y \mid d(s,y) \leq r\}$) satisfy $S^{(r)} \subseteq S^{(r')}$ if $r \leq r'$, and thus assemble to an $\mathbf{R}$-indexed filtration, called the *offset filtration* of S.
2. This filtered space is replaced by a finite filtered simplicial complex K_*, usually with the same (or similar) homology as the filtered space. For example, let Y be a Euclidean space, and define the *Voronoi* domain of $s \in S$ as
$$\mathrm{Vor}(s) = \{y \in Y \mid d(y,s) \leq d(y,t)\ \forall t \in S\}.$$
Then $\{B_r(s) \cap \mathrm{Vor}(s) \,|\, s \in S\}$ is a cover of $S^{(r)}$, and its nerve is a simplicial complex filtered by r, called the *Delaunay* or *alpha filtration* [60]; see Figure 1.1. By the (functorial) nerve theorem [13], the Delaunay filtration and the offset filtration are (functorially) homotopy equivalent. Other common choices are the *Čech-filtration* and the the *Vietoris–Rips filtration* with vertex set S; see Examples 2.1.7 and 2.1.8.
3. Computing the simplicial homology $H_\bullet(K_r)$ (with coefficients in a field k) for every r yields a collection $H_\bullet(K_*)$ of finite dimensional vector spaces $H_\bullet(K_r)$ for every r, connected by the morphisms $H_\bullet(K_r) \to H_\bullet(K_{r'})$ induced by $K_r \hookrightarrow K_{r'}$ for all $r \leq r'$. This collection is called the *persistent homology* of K_*.



(a) $r = 0.0$ **(b)** $r = 0.3$ **(c)** $r = 0.8$ **(d)** $r = 1.2$ **(e)** $r = 1.3$

Figure 1.1: *Offset filtration* (blue) $\bigcup_{s\in S} B_r(s)$, of a point cloud S (black) at different values of r, superimposed with the *Delaunay filtration* (black, light blue).

Figure 1.2: Persistence diagram of the Delaunay filtration K_* from Figure 1.1. Each point (x, y) in the diagram stands for a homology class such that for each r, the classes with $x \leq r < y$ form a basis of $H_\bullet(K_r)$. There is a single point for $H_1(K_*)$ that is far from the diagonal. This corresponds to the homology of the annulus the points are sampled from.

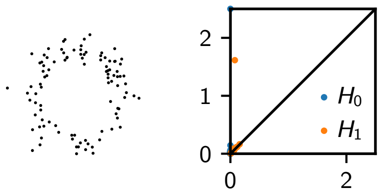
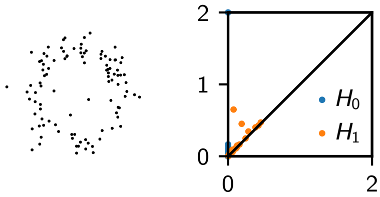


Figure 1.3: Point cloud obtained from adding three outliers to Figure 1.1 (left) and persistence diagram of the resulting Delaunay filtration (right). The actual homology of the sampled annulus cannot be seen from in the diagram anymore.



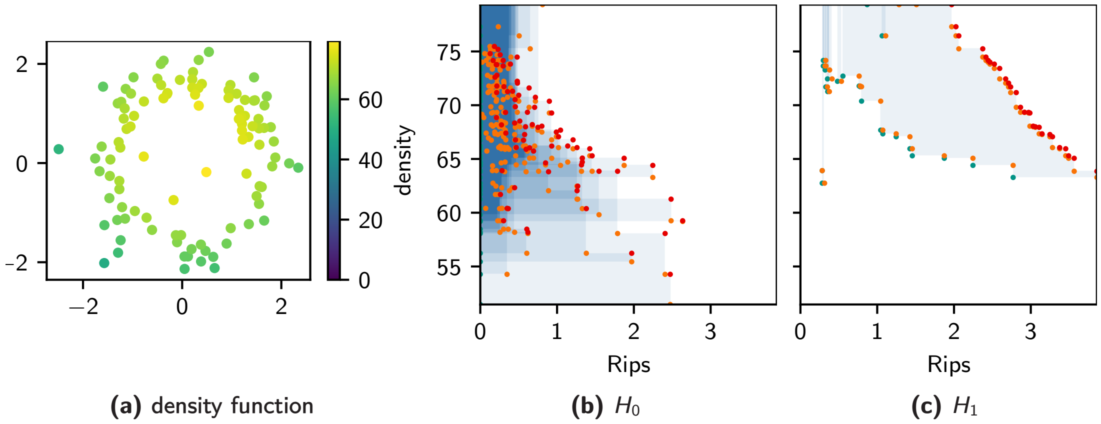
4. The persistent homology $H_\bullet(K_*)$ can be compactly described by its *barcode* [76, 128], or by its *persistence diagram*, see Figure 1.2. Each point (x, y) of the persistence diagram represents a cycle in K_z for all $z \geq x$. It is homologous to older cycles if $z \geq y$. Thus, for each z, the points (x, y) of the diagram with $x \leq z < y$ represent a basis of $H_z(K_*)$. The barcode can be easily computed by a Gaussian column reduction scheme [128], and efficient implementations are widely available [1, 8, 81, 111, 122].
   If S is sampled from a subspace $X \subset Y$, then under certain conditions on S, one can infer the homology of X from the persistent homology of S [43, 48, 106]. In our example, we have $Y = \mathbf{R}^2$, $X \supset S$ is an annulus, and the long lived homology classes in Figure 1.2 correspond to the homology of X.

Particular strengths of this approach lie in the fact that it captures topological features of a space at different scales at the same time. If X is a submanifold of $\mathbf{R}^n$, then representatives of the homology classes can be used to reconstruct a triangulation of X [18, 49, 57]. A vectorized version of persistence diagrams can be used as input for other (e.g., machine learning) data analysis methods [92, 114, 115]. Furthermore, persistent homology is stable [16, 28, 40, 48], in the sense that small perturbations to the input lead to only small perturbations of the output, in a way that can be made precise.

Despite these stability results, however, traditional one-parameter persistent homology is susceptible to outliers. To make this precise, assume that $S \subset Y$ is a sample of a subspace $X \subset Y$ that also contains (few) outliers not in X. Already for relatively small values of $|S \setminus X|/|S|$, the homology $H_\bullet(X)$ may become unintelligible from $H_\bullet(S^{(r)})$; see Figure 1.3. Additionally, correctly capturing the topology of X from S is difficult if X has topological features of different scales.

Multi-parameter persistent homology [34, 91] is seen as a possible remedy for the above shortcomings. It extends (one-parameter) persistent homology by introducing additional parameters that control, for instance, the density of the sampled points. A common problem is to compute a *minimal free resolution* of $H_\bullet(K_*)$, which generalizes the construction of the barcode to more than one parameter.

Although, contrary to the situation in one-parameter persistence, the indecomposable multiparameter persistence modules cannot be classified as easily as in the one-parameter case [34, 74] and give no stable invariant [20], many other invariants, such as graded Betti numbers [90, 99], the (generalized) rank invariant [56, 89], signed [27] and fibered barcodes [100], shift dimension [37] and others, can be computed from a minimal free resolution, which motivates the interest in computing these. An example for such a minimal free resolution is shown in Figure 1.4. The



(a) density function **(b)** H_0 **(c)** H_1

Figure 1.4: The point cloud from Figure 1.3, equipped with a density function. The right two plots show the Hilbert function (blue) and the graded Betti numbers (teal: β_0, red: β_1, orange: β_2) of the associated function-Rips homology. See Figure 2.6a for details. A horizontal slice at density value ρ through the Betti diagram corresponds to the one-parameter persistent homology of the subsample having only points with density value at most ρ. For example, a horizontal slice through the diagram for H_1 at $\rho \approx 70$ would produce one long bar in the barcode, corresponding to the region in the parameter space for which H_1 is one-dimensional. This is precisely the long bar in the barcode of H_1 one would see if one removes the points with density greater than 70 from the pointcloud (which comprises also the outliers inside the circle).

picture shows the Hilbert function and graded Betti numbers of the function-Rips persistent homology of the same point cloud as above.

For one- and two-parameter persistence, the computational worst-case complexity for computing the persistent homology of a (one-critical) filtered simplicial complex is cubic in the number of simplices. In practice, however, computing the minimal free resolution of two-parameter persistent homology is much more difficult than computing the barcode of a one-parameter filtered complex [33, 73, 85, 86, 98]. For more than two parameters, no method with cubic complexity is known.

Since the first algorithm for one-parameter persistent homology [128] has been described, substantial performance improvements have been achieved, some of which rely on computing (relative) persistent cohomology instead of (absolute) persistent homology. A duality principle allows to relate barcodes of absolute and relative homology and cohomology [19, 55]. It has been unclear so far, however, how this approach can be generalized beyond one-parameter persistent homology.

Contribution

Summary In this thesis, we explore different duality principles extending the known dualities in one-parameter persistence [55] to two- and multi-parameter persistent (co)homology. We present different ways to compute a minimal free resolutions of the persistent cohomology of a finite, one-critically two-parameter-filtered simplicial complex, and explain how this can be related to minimal free resolutions of its persistent homology. We devise an algorithm that allows for an optimization scheme similar to *clearing* in one-parameter persistence. A C++-implementation of our algorithm is publicly available [94]. Experiments demonstrate the practicability of our approach.

Cohomology computation Let K_* be a finite n-parameter filtered simplicial complex that is *one-critical*; that is, every simplex enters the filtration at a unique minimal value. Let $K = \operatorname{colim}_{z\in\mathbf{Z}^n} K_z$. In the following, when talking about (co)homology, we always mean reduced (co)homology with coefficients in a fixed field k.

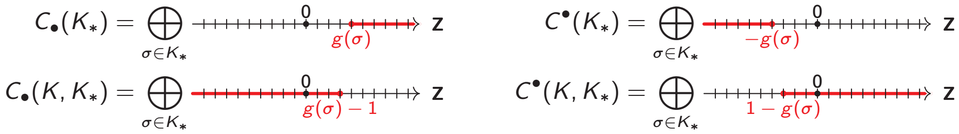

$C_\bullet(K_*) = \bigoplus_{\sigma\in K_*}$
0
$g(\sigma)$
Z
$C^\bullet(K_*) = \bigoplus_{\sigma\in K_*}$
0
$-g(\sigma)$
Z
$C_\bullet(K, K_*) = \bigoplus_{\sigma\in K_*}$
0
$g(\sigma) - 1$
Z
$C^\bullet(K, K_*) = \bigoplus_{\sigma\in K_*}$
0
$1 - g(\sigma)$
Z

Figure 1.5: Absolute and relative simplicial (co)chains of a $\mathbf{Z}$-filtered simplicial complex K_*. A persistence module is free if it is a direct sum of interval modules supported on intervals of the form $[b, \infty)$ for b in $\mathbf{Z}$. In particular, $C_\bullet(K_*)$ and $C^\bullet(K, K_*)$ are (co)chain complexes of free modules, where the red line indicate the support of a simplex $\sigma \in K_*$ in the respective (co)chain complexes.

The main challenge in computing the absolute and relative cohomology of K_* lies in the fact that in multi-parameter persistence, $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$ are no cochain complexes of free modules. This is different from one-parameter persistence, where $C^\bullet(K, K_*)$ is a cochain complex of free modules; see Figures 1.5 and 1.6 for an illustration. Working with (co)chain complexes of free modules is necessary to ensure applicability of the common column reduction schemes to compute persistent (co)homology. In this thesis, we propose two different ways to remedy this, which we summarize in the following.

First approach As said above, in one-parameter persistence, $C^\bullet(K, K_*)$ is a cochain complex of free modules. The approach presented in Chapter 3 generalizes this to the multi-parameter case in the following way. We define a certain cochain complex $N^\bullet(K_*) := (\nu C_\bullet(K_*))^*$, where $(-)^*$ denotes the pointwise dual persistence module and ν the *Nakayama functor*; see Definitions 2.1.10, 3.1.2 and 3.2.2. In the one-parameter case, we have $C^\bullet(K, K_*) = (\nu C_\bullet(K_*))^*$. In persistence over more than one parameter, $C^\bullet(K, K_*)$ is not a cochain complex of free modules; however, $N^\bullet(K_*)$ is. In Chapter 3, we work with the latter complex and show:

Theorem A (page 42). *Let K_* be a one-critically $\mathbf{Z}^n$-filtered simplicial complex, such that $H_\bullet(K_*)$ is finitely supported. Then there is a natural isomorphism*

$$H^d(K_*) \cong H^{d+n}(N^\bullet(K_*))$$

for all d.

Here, we say that a persistence module M is *finitely supported* if its components M_z are zero for all but finitely many $z \in \mathbf{Z}^n$. This theorem is a generalization of the fact that if K is acyclic, then $H_d(K_*) \cong H^{d+1}(K, K_*)^*$, which can be seen from the long exact sequence of the pair (K, K_*). If $H_*(K_*)$ is not finitely supported, one can replace $C_\bullet(K_*)$ by a complex $\hat{C}_*$ finitely supported homology, which can be used to compute $H^d(K_*)$; see Section 3.7. Furthermore, for any number of parameters, the cohomology of $N^\bullet(K_*)$ can be related to $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ (see Theorem A and Corollary 3.2.13).

Second approach The second approach, presented in Chapter 4, is to replace $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$ by quasi-isomorphic cochain complexes of free modules. We do this by choosing free resolutions of $C^d(K_*)$ (resp. $C^d(K, K_*)$) for each d and deriving an explicit formula for a minimal free resolution of $H^d(K_*)$ (resp. $H^d(K, K_*)$) using these resolutions.

Necessity of the finite-support condition In one-parameter persistence, it is known that $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ uniquely determine each other up to isomorphism even if $H^\bullet(K) \neq 0$ (see Corollary 2.2.11). In Sections 3.6 and 4.6, we show by counterexamples that corresponding statements do not hold in two-parameter persistence, namely:

Theorem B (pages 55 and 98). *Let K_* be a one-critically 2-parameter filtered simplicial complex.*

(a) *Unless $H_\bullet(K_*)$ is finitely supported, $H^\bullet(K_*)$ and $H^\bullet(N^\bullet(K_*))$ need not determine each other up to isomorphism.*

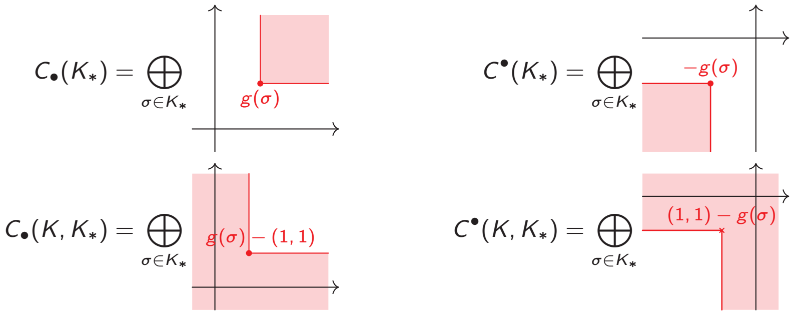

$C_\bullet(K_*) = \bigoplus_{\sigma \in K_*}$
$g(\sigma)$
$C^\bullet(K_*) = \bigoplus_{\sigma \in K_*}$
$-g(\sigma)$
$C_\bullet(K, K_*) = \bigoplus_{\sigma \in K_*}$
$g(\sigma) - (1,1)$
$C^\bullet(K, K_*) = \bigoplus_{\sigma \in K_*}$
$(1,1) - g(\sigma)$

Figure 1.6: Absolute and relative simplicial (co)chains of a $\mathbf{Z}^2$-filtered simplicial complex K_*. A module is free if it is a direct sum of modules $F(z)$, each of which has components $F(z)_w = \begin{cases} k & \text{if } w \geq z, \\ 0 & \text{otherwise;} \end{cases}$ i.e., the module of the upper right quadrant with minimal element z. In particular, if K_* is one-critically $\mathbf{Z}^2$-filtered, then $C_\bullet(K_*)$ is a chain complex of free modules. None of $C_\bullet(K, K_*)$, $C^\bullet(K_*)$, $C^\bullet(K, K_*)$ is a (co)chain complex of free modules. This is different from 1-parameter persistence; see Figure 1.5.

(b) *Unless $H_\bullet(K) = 0$, $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ need not determine each other up to isomorphism.*

Higher coboundary morphisms In one-parameter persistent homology, one can show that if $H^\bullet(K) = 0$, then $H^{d+1}(K, K_*)$ can be computed from the (matrix representing the) coboundary morphism

$$\delta^{d+1} \colon C^d(K, K_*) \to C^{d+1}(K, K_*)$$

alone. That is, it is not necessary to consider the coboundary morphism δ^{d+2}, because the assumption ensures that all relevant information is already contained in δ^{d+1}; see Section 2.2.1. This is of practical importance, because for Vietoris–Rips complexes, a matrix representing δ^{d+2} would be prohibitively large. By providing explicit algorithms, we show the following analogue statements for two-parameter persistence:

Theorem C (page 54)**.** *If K_* is a finite two-parameter filtered complex such that $H_\bullet(K_*)$ is finitely supported, then a minimal free resolution of $H^{d+2}(N^\bullet(K_*))$ can be computed from the coboundary morphism $N^d(K_*) \to N^{d+1}(K_*)$ alone.*

Theorem D (page 90)**.** *If K_* is a finite two-parameter filtered complex such that $H_\bullet(K) = 0$, where $K = \operatorname{colim}_z K_z$, then a minimal free resolution $H^{d+1}(K, K_*)_\bullet$ of $H^{d+1}(K, K_*)$ can be computed from the coboundary morphism $C^d(K, K_*) \to C^{d+1}(K, K_*)$ alone.*

Duality and free resolutions So far, we have concentrated on computing a minimal free resolution of $H^d(K_*)$, either directly, or by computing a minimal free resolution of $H^{d+1}(K_*)$ or $H^{d+n}(N^\bullet(K_*))$ and showing that these cohomology modules are isomorphic $H^d(K_*)$ under certain conditions. In applications, however, we might be interested in a free resolution of persistent *homology* $H_d(K_*)$. We show that if $H^d(K_*)$ has finite total dimension, then there is a simple correspondence between minimal free resolutions of $H^d(K_*)$ and $H_d(K_*)$. More generally, we show:

Theorem E (page 48)**.** *Let M be a finitely generated n-parameter persistence module with bounded support. For graded matrices $U_1, \dots, U_n$, the following are equivalent:*

(*i*) *$U_1, \dots, U_n$ represent a free resolution of module M,*

(*ii*) *$U_1, \dots, U_n$ represent an injective resolution of the shifted module $M\langle -\epsilon \rangle$,*

(*iii*) *the graded transposes $U_1^\top, \dots, U_n^\top$ represent a free resolution of the dual module $M\langle -\epsilon \rangle^*$.*

Computation Combining Theorems A and E allows us to obtain a minimal free resolution of $H_d(K_*)$ for a two-parameter filtered complex K_* in the following steps:

1. If using b) or c) in the following, ensure before that K_* has finitely supported homology, e.g. by replacing K_* by a suitable complex with finitely supported homology as described in Section 3.7.
2. Compute a free resolution of either
   a) $H^d(K_*)$ (see Section 4.2.1);
   b) $H^{d+1}(K, K_*)$ (see Section 4.2.2) and obtain a free resolution of $H^d(K_*)$ by the long exact sequence of cohomology; or
   c) $H^{d+2}(N^\bullet(K_*))$ (see Chapter 3) and obtain a free resolution of $H^d(K_*)$ by Theorem A.
3. From that, obtain a free resolution of $H_d(K_*)$, using Theorem E.

Only the second step is computationally involved. Without loss of generality, assume that K_* has finitely supported homology. We propose strategies algorithms to compute minimal free resolutions of $H^{d+2}(N^\bullet(K_*))$, $H^d(K_*)$ and $H^{d+1}(K, K_*)$ in the two-parameter case, see Sections 3.5, 4.2 and 4.4.

Pulling back from the colimit For $H^{d+2}(N^\bullet(K_*))$, the core of the proposed algorithm (see Algorithm 11) is the following insight, which also underlies the proofs of Theorems C and D. For a persistence module M, we define the vector space $\operatorname{colim} M := \operatorname{colim}_{z \in \mathbf{Z}^n} M_z$.

Theorem F (page 50)**.** *Let M, N be free persistence modules and $f\colon M \to N$ be a morphism, and let U be the unique maximal submodule of M with* $\operatorname{colim} U = \operatorname{colim} \ker f$*. Then $U = \ker f$.*

For a general submodule $V \subseteq M$, it is not true that for the unique maximal submodule $U \subseteq M$ with $\operatorname{colim} U = \operatorname{colim} V$, the inclusion $U \subseteq V$ is an equality, even if U and/or M are free. The lemma shows that this is the case for kernels indeed.

If M and N are free two-parameter modules, then also $\ker f$ is free, which can be derived from Hilbert's syzygy theorem; see Corollary 2.3.20. We propose an algorithm (see Theorem 3.4.7) that computes a basis of $\ker f$, given basis of $\operatorname{colim} \ker f$. Using this, we show how to compute $H^{d+1}(K, K_*)$ and $H^{d+2}(N^\bullet(K_*))$; see Algorithms 11 and 13. Both algorithms allow for a optimization that is analogous to the *clearing* scheme in one-parameter persistence [11, 46]; see Remark 3.5.3.

Implementation and experiments Earlier experiments that we will not report here have shown that that the approach using the complex $N^\bullet(K_*)$ from Chapter 3 is much more efficient than working with free resolutions of $C^\bullet(K_*)$ or $C^\bullet(K, K_*)$ as described in Chapter 4; see also Section 4.5. For this reason, for the implementation and runtime experiments that we describe in Chapter 5, we concentrate on the former approach. The implementation is open source and publicly available [94]. This implementation was also used to produce the diagram in Figure 1.4.

We use our implementation to compute minimal free resolutions of $H_\bullet(K_*)$ for density-Rips complexes and other two-parameter filtered complexes of different sizes; see Section 5.2. We compare our approach to the state-of-the-art algorithm from [73], both with respect to run time and memory footprint. We analyze the effect of the well-known chunk preprocessing [72] and implementation details (such as the sparse matrix representation employed) on the performance.

The experiments show that our approach is able to outperform the method from [73], both with respect to run time and memory. This is true both for the heap and the vector format [12] for sparse matrices. When computing $H_d(K_*)$ for $d \geq 2$, our algorithm was faster on almost all instances, by a factor of often up to 15. There were instances (for instances the cyclooctane data set or points clouds samples orthogonal groups, see Section 5.2) that were tractable only with the cohomology algorithm, while the homology algorithm did not terminate within acceptable time. Our algorithm can be seen as an alternative to chunk preprocessing, since both serve to efficiently remove trivial summands from the input complex. Consequently, while the algorithm from [73] is known to strongly benefit from chunk preprocessing, our algorithm does not require chunk preprocessing to be efficient. Independently of our cohomology algorithm, we propose a cohomological version of chunk preprocessing for the homology algorithm [73] that we call *chunk* preprocessing.*

Acknowledgements

I want to express my gratitude for my supervisor Ulrich Bauer for the opportunity to connect my interest for representation theory and algebra to algebraic topology, computational geometry, computer science, software development and data analysis. Working in computational topology introduced me to many facets of mathematics that were concealed to me before. I also want to thank him for the support and encouragement atmosphere I enjoyed in his group.

I want to thank Michael Lesnick, who offered me a big portion of his time to discuss our research. I also want to thank him for the opportunity for a research visit to Albany, which, apart from being a unique experience on its own, enabled me to substantially improve and extend the contents of this thesis.

I want to thank my colleagues for the uncountable inspiring discussions we had, let apart the stimulating atmosphere I enjoyed during the recent years. In particular, I want to thank Fabian Roll for his thinking about my research problems and providing ideas and feedback for many drafts of my writing. I want to thank him and my other proofreaders David Hien, Lukas Kühne and Nico Stucki for their help with bringing this thesis into shape.

Chapter 2

Background

In this section, we give an overview of the relevant preliminaries of single and multi-parameter persistent homology. We introduce persistence modules, both as functors over a poset and as graded modules over an algebra, and explain how they arise from data. We explain the algorithmic details of the computation of one-parameter persistent homology and cohomology, and explain how minimal free resolutions of two-parameter persistent homology are computed.

2.1 Persistence modules

Let k be a field, and let Vec (respectively, vec) denote the category of k-vector spaces (resp., finite dimensional k-vector spaces). Let P be a poset, viewed as a category with one object p for every $p \in P$ and precisely one morphism from p to q for each $p \leq q$ in P.

Definition 2.1.1. A *P-persistence module* is a functor $M\colon P \to \mathrm{Vec}$ that assigns to every $p \in P$ a k-vector space M_p, and to every pair $p \leq q$ of comparable elements in P a morphism $M_{qp}\colon M_p \to M_q$. The vector spaces M_p are called the *components* of M, and he maps M_{qp} are called the *structure maps* of M. For $m \in M_p$, we denote by $g(m) := p$ the *grade* of m. The module M is called *pointwise finite dimensional* if M_p is finite dimensional for all $p \in P$. The *support* of M is $\operatorname{supp} M := \{p \in P \mid M_p \neq 0\}$. A morphism of persistence modules is a natural transformation of functors. We write Vec^P (resp., vec^P) for the category of P-persistence modules (resp., pointwise finite dimensional modules).

If M and N are persistence modules, then $M \oplus N$ is the persistence module with $(M \oplus N)_p = M_p \oplus N_p$. If $f\colon M \to N$ is a morphism of persistence modules, then $\ker f$ and $\operatorname{coker} f$ are the persistence modules with $(\ker f)_p = \ker f_p$ and $(\operatorname{coker} f)_p = \operatorname{coker} f_p$. These render Vec^P and vec^P an abelian category. For example,

$$\begin{tikzcd}
 & 0 \arrow[r] & 0 \arrow[r] & k \arrow[rrrr, bend left, "\left(\begin{smallmatrix}1\\-1\\1\end{smallmatrix}\right)"] & & k \arrow[r, "\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)"] & k^2 \arrow[r, "\left(\begin{smallmatrix}1&0\\0&1\\0&0\end{smallmatrix}\right)"'] & k^3 \arrow[rrrr, bend left, "\left(\begin{smallmatrix}1&1&0\\0&1&1\end{smallmatrix}\right)"] & & k \arrow[r, "\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)"] & k^2 \arrow[r, equal] & k^2 & \\
0 \to & 0 \arrow[r] \arrow[u] & 0 \arrow[r] \arrow[u] & 0 \arrow[u] & \to & 0 \arrow[r] \arrow[u] & k \arrow[r, "\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)"'] \arrow[u, "\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)"] & k^2 \arrow[u, "\left(\begin{smallmatrix}0&0\\1&0\\0&1\end{smallmatrix}\right)"'] & \to & 0 \arrow[r] \arrow[u] & k \arrow[r, "\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)"'] \arrow[u, "\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)"] & k^2 \arrow[u, equal] & \to 0 \\
 & 0 \arrow[r] \arrow[u] & 0 \arrow[r] \arrow[u] & 0 \arrow[u] & & 0 \arrow[r] \arrow[u] & 0 \arrow[r] \arrow[u] & k \arrow[u, "\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)"'] & & 0 \arrow[r] \arrow[u] & 0 \arrow[r] \arrow[u] & k \arrow[u, "\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)"'] &
\end{tikzcd}$$

is a short exact sequence of modules over the poset $\{0, 1, 2\}^2$.

2.1.1 Filtered spaces and complexes

Typically, persistence modules arise as the homology of filtered topological spaces or filtered simplicial complexes. We write Top for the category of topological spaces.

Definition 2.1.2. A *P-filtered space* is a functor $X_*\colon P \to \mathrm{Top}, p \mapsto X_p$, such that $X_{qp} : X_p \to X_q$ is an inclusion for all $p \leq q$ in P. We write $\mathrm{Top}^{\subseteq P}$ for the category of P-filtered topological spaces.

Example 2.1.3 (Sublevel sets). Let X be a topological space, equipped with a function $f\colon X \to P$. The *sublevel set filtration* $X_{f\leq *}$ is the P-filtered topological subspace of X with $X_{f\leq p} = f^{-1}(\{q \in P \mid q \leq p\})$.

Example 2.1.4 (Offset filtration [79]). The *offset filtration* $S^{(*)}$ of a subset S of a metric space X is the $\mathbf{R}$-filtered subspace of X with $S^{(r)} := \bigcup_{s\in S} B_r(x)$. Here $B_r(s)$ denotes the open ball of radius r centered at s. If $M \subseteq \mathbf{R}^n$ is a submanifold and $S \subset M$ a finite sample, then under certain conditions, the homology of M can be obtained from $M^{(r)}$ and $S^{(r)}$ [61, p. 12]. See also [48].

For computational aspects, it is more convenient to work with a discrete model of topological spaces, namely, filtered simplicial complexes.

Definition 2.1.5. An *(abstract) simplicial complex* K is a set of nonempty finite sets such that if $\sigma \in K$ and $\rho \subseteq \sigma$ is non-empty, then $\rho \in K$. For any $d \geq 0$, the elements of $K^d = \{\sigma \in K \mid |\sigma| = d+1\}$ are called the *d-simplices* of K. The 0-simplices are called the *vertices* of K. A morphism $f\colon K \to L$ of simplicial complexes is a morphism $K \to L$ of sets such that $f(K^0) \subseteq L^0$, and $f(\sigma) = \bigcup_{v\in\sigma} f(\{v\})$. We denote by Simp the category of simplicial complexes. A P-filtered simplicial complex is a functor $K_*\colon P \to \mathrm{Simp}, p \mapsto K_p$ such that $K_{qp}\colon K_p \to K_q$ is injective for all $p \leq q$. If K_* is a filtered simplicial complex, we write $K := \operatorname{colim}_{p\in P} K_p$. We call K_* finite if K is finite. We write $\mathrm{Simp}^{\subseteq P}$ for the category of P-filtered simplicial complexes.

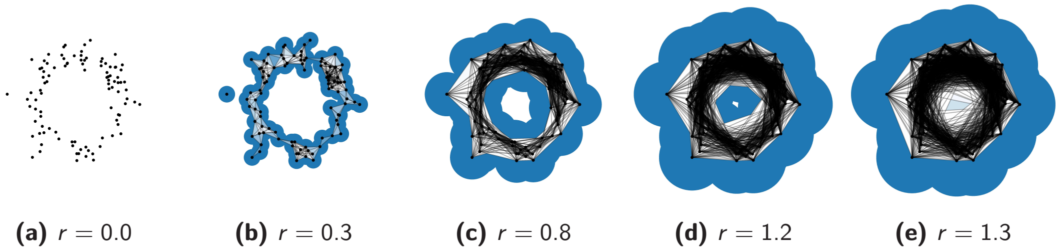
Example 2.1.6 (Delaunay complex [60]). Let S be a finite subset of a metric space X. For $s \in S$, let $\mathrm{Vor}(s) = \{x \in X \mid \forall t \in S\colon d(x,t) \geq d(x,s)\}$ be the *Voronoi domain* of s with respect to S, and let $\mathrm{Vor}_r(s) = \mathrm{Vor}(s) \cap B_r(s)$. Then the *Delaunay filtration* or *α-filtration* $\mathrm{Del}_*(S)$ is the $\mathbf{R}$-filtered simplicial complex with $\mathrm{Del}_r(S) = \{\sigma \subseteq S \mid 0 < |\sigma| < \infty \text{ and } \bigcap_{s\in\sigma} \mathrm{Vor}_r(s) \neq \emptyset\}$; see Figure 1.1.

Example 2.1.7 (Čech complex). Let S be a subset of a metric space X. The *Čech complex* $\check{C}_*(S)$ is the $\mathbf{R}$-filtered abstract simplicial complex given by $\check{C}_r(S) = \{\sigma \subseteq S \mid 0 < |\sigma| < \infty \text{ and } \bigcap_{s\in\sigma} B_r(s) \neq \emptyset\}$.

See Figures 1.1 and 2.1 for examples. The Delaunay complex $\mathrm{Del}_r(S)$ is the nerve of the closed cover $\{\mathrm{Vor}_r(s) \mid s \in S\}$ of the offset space $S^{(r)}$. The Čech complex $\check{C}_r(S)$ is the nerve of the closed cover $\{B_r(s) \mid s \in S\}$ of $S^{(r)}$. Under suitable conditions (for example, if X is Euclidean), the functorial nerve theorem [13] guarantees that $S^{(r)}$ and the geometric realizations $|\mathrm{Del}_r(S)|$ and $|\check{C}_r(S)|$ are homotopy equivalent to $S^{(r)}$ in an $\mathbf{R}$-filtered way; see also [10]. Therefore, both can be seen as a combinatorial model of the filtered space $r \mapsto S^{(r)}$. However, both complex are computationally involved in high dimensions or in non-Eucidean spaces. A common alternative is the following:

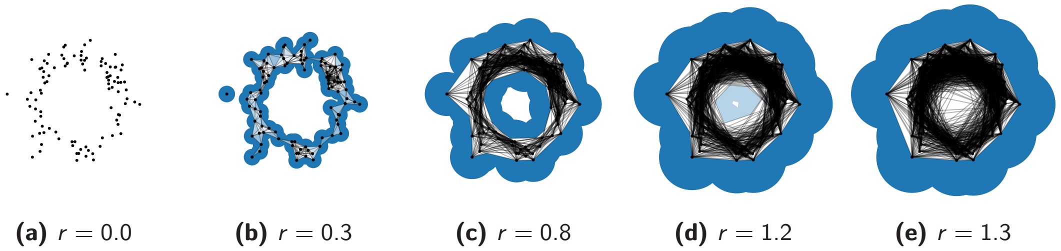
Example 2.1.8 (Vietoris–Rips complex [125]). Let S be a metric space. For $\sigma \subseteq S$, let $\operatorname{diam}\sigma = \max_{s,t\in\sigma} d(s,t)$. The *Vietoris–Rips complex* $VR_*(S)$ associated to S is the $\mathbf{R}$-filtered abstract simplicial complex given by $VR_r(S) = \{\sigma \subseteq S \mid 0 < |\sigma| < \infty \text{ and } \operatorname{diam}\sigma \leq r\}$.

The Vietoris–Rips complex is a *clique complex*; meaning that for $d > 2$, a d-simplex $\sigma \in VR_r(S)$ if $\rho \in VR_r(S)$ for all its 2-faces $\rho \in \binom{\sigma}{2}$. See Figure 2.2 for an example. For $S \subseteq \mathbf{R}^n$, the Vietoris–Rips complex relates to the Čech complex via mutual inclusions $VR_r(S) \subseteq \check{C}_{r'/2}(S) \subseteq VR_{r'}(S)$ whenever $\frac{r'}{r} \geq \sqrt{2n/(n+1)}$ [54, Theorem 2.5, 4, 41]. If n is large, it is considerably easier to decide if a set $\sigma \subset S$ lies in $VR_r(S)$ than to decide of σ lies in $\mathrm{Del}_r(S)$ or $\check{C}_r(S)$, which motivates the interest in $VR(S)$.



(a) $r = 0.0$ **(b)** $r = 0.3$ **(c)** $r = 0.8$ **(d)** $r = 1.2$ **(e)** $r = 1.3$

Figure 2.1: *Offset filtration* (blue) $\bigcup_{s\in S} B_r(s)$, of the point cloud from Figure 1.1a, superimposed with a projection of the 2-skeleton of the *Čech filtration* (black, light blue).



(a) $r = 0.0$ **(b)** $r = 0.3$ **(c)** $r = 0.8$ **(d)** $r = 1.2$ **(e)** $r = 1.3$

Figure 2.2: Like Figure 2.1, but with the *Vietoris–Rips filtration* VR_{2r}.

If S is finite, let $r_1 < r_2 < \cdots < r_n$ be the distinct values in $\{\operatorname{diam}\sigma \mid \sigma \subseteq S, \sigma \neq \emptyset\}$. By setting

$$K_i(S) = \begin{cases} VR(S)_{r_1} & \text{if } i \leq 1, \\ VR(S)_{r_i} & \text{if } 1 \leq i \leq n, \\ VR(S)_{r_n} & \text{if } n \leq i, \end{cases} \tag{2.1}$$

we obtain a $\mathbf{Z}$-filtered simplicial complex K_* that we also call the Vietoris–Rips complex of S and also denote by $VR_*(S)$.

2.1.2 Persistent homology

Since Vec^P is an abelian category, it makes sense to consider chain complexes of persistence modules. A *chain complex* $C_\bullet$ is a sequence

$$\cdots \to C_{d+1} \xrightarrow{\partial_{d+1}} C_d \xrightarrow{\partial_d} C_{d-1} \to \cdots$$

of modules and morphisms, such that $\partial_d\partial_{d+1} = 0$ for all d. Its *d-cycles* $Z_d(C_\bullet) := \ker \partial_d$, *d-boundaries* $B_d(C_\bullet) := \operatorname{im}\partial_{d+1}$ and *d-homology* $H_d(C_\bullet) := Z_d(C_\bullet)/B_d(C_\bullet)$ are persistence modules. Dually, a *cochain complex* $C^\bullet$ is a sequence

$$\cdots \to C^{d-1} \xrightarrow{\delta^d} C^d \xrightarrow{\delta^{d+1}} C^{d+1} \to \cdots$$

of modules and morphisms, such that $\delta^{d+1}\delta^d = 0$ for all d. Its *d-cocycles* $Z^d(C^\bullet) := \ker \delta^{d+1}$, *d-boundaries* $B^d(C^\bullet) := \operatorname{im}\delta^d$ and *d-cohomology* $H^d(C^\bullet) := Z^d(C^\bullet)/B^d(C^\bullet)$ are persistence modules. We say that a (co)chain complex $C_\bullet$ (resp., $C^\bullet$) has finite (total) dimension if $\bigoplus_d C_d$ (resp. $\bigoplus_d C^d$) is finite dimensional. A (co)chain complex is *acyclic* if its (co)homology is zero in all dimensions. (Co)chain complexes of persistence modules form an abelian category.

Remark. We silently require that all (co)chain complexes we work with are bounded below.

We assume that P has an involution $P \to P, p \mapsto -p$ that identifies P with its opposite poset P^{op}; that is, $p \leq q$ if and only if $-q \leq -p$. We also assume that P is a lattice; that is, each $p, p' \in P$ have a unique *least upper bound* or *join*, denoted by $p \vee p'$, and a unique *greatest lower bound* or *meet*, denoted by $p \wedge p'$.

Example 2.1.9. The sets $\mathbf{Z}^n$ or $\mathbf{R}^n$ are posets with the ordering $(z_1, \dots, z_n) \leq (z'_1, \dots, z'_n)$ if $z_i \leq z'_i$ for all i. Negation identifies both with their opposite poset. Both are lattices, where

$$(z_1, \dots, z_n) \vee (z'_1, \dots, z'_n) = (\max\{z_1, z'_1\}, \dots, \max\{z_n, z'_n\}),$$
$$(z_1, \dots, z_n) \wedge (z'_1, \dots, z'_n) = (\min\{z_1, z'_1\}, \dots, \min\{z_n, z'_n\}).$$

Let the k-dual vector space of a k-vector space V be denoted by V^*.

Definition 2.1.10. The *dual module* M^* of a persistence module $M \in \mathrm{Vec}^P$ is the P-persistence module with $(M^*)_p = (M_{-p})^*$ and $(M^*)_{qp} = (M_{-p,-q})^*$.

Let $C_\bullet$ be a chain complex, then its dual complex $C^\bullet \coloneqq (C_\bullet)^*$ is the cochain complex with $C^d = (C_d)^*$ and $\delta^d = (\partial_d)^*$. The functor $(-)^* \colon \mathrm{Vec}^P \to \mathrm{Vec}^P$ is exact contravariant. Therefore, there is a natural isomorphism

$$H^\bullet((C_\bullet)^*) \cong H_\bullet(C_\bullet)^*.$$

Definition 2.1.11. Let $K_* \in \mathrm{Simp}^{\subseteq P}$. The *(absolute) persistent chain complex* of K_* is the chain complex $C_\bullet(K_*)$ of P-persistence modules, with components

$$C_d(K_*) \colon P \to \mathrm{Vec},\ p \mapsto C_d(K_p),$$

where $C_\bullet(K_p)$ denotes the simplicial chain complex of the simplicial complex K_p. We denote its d-cycles, d-boundaries and dth homology by $Z_d(K_*)$, $B_d(K_*)$ and $H_d(K_*)$ The module $H_d(K_*)$ is called the *dth (absolute) persistent homology* of K_*.

We can see a persistence module $M \in \mathrm{Vec}^P$ as a P-indexed diagram of vector spaces. In that sense, we define the functors

$$\begin{aligned} \mathrm{colim} \colon \mathrm{Vec}^P &\to \mathrm{Vec}, & \Delta \colon \mathrm{Vec} &\to \mathrm{Vec}^P, & \lim \colon \mathrm{Vec}^P &\to \mathrm{Vec}, \\ M &\mapsto \mathrm{colim}_P\, M, & V &\mapsto \left\{ \begin{smallmatrix} p & \mapsto V, \\ (p \leq q) & \mapsto \mathrm{id}_V \end{smallmatrix} \right., & M &\mapsto \lim_P M. \end{aligned}$$

The functor Δ, called the *diagonal*, is right adjoint to colim and left adjoint to lim. For a vector space V, ΔV is the module that is constantly V.

Definition 2.1.12. Let $K_* \in \mathrm{Simp}^{\subseteq P}$ and $K \coloneqq \bigcup K_*$. The *relative persistent chain complex* of K_* is the chain complex

$$C_\bullet(K, K_*) \coloneqq \frac{\Delta C_\bullet(K)}{C_\bullet(K_*)}$$

of persistence modules. We denote the relative d-cycles, d-boundaries and dth homology by $Z_d(K, K_*)$, $B_d(K, K_*)$ and $H_d(K, K_*)$, respectively.

Remark. For every $p \in P$, the absolute chain complex $C_\bullet(K_p)$ is spanned by the simplices $\sigma \in K_p$, while the relative chain complex $C_\bullet(K, K_p)$ is spanned by the simplices $\sigma \in K \setminus K_p$.

Definition 2.1.13. For $K_* \in \mathrm{Simp}^{\subseteq P}$ and $K \coloneqq \bigcup K_*$, we define the *absolute persistent cochain complex* $C^\bullet(K_*) \coloneqq C_\bullet(K_*)^*$ and the *relative persistent cochain complex* $C^\bullet(K, K_*) \coloneqq C_\bullet(K, K_*)^*$ of K_*. We denote the respective cocycles, coboundaries and cohomology by $Z(-)$, $B^\bullet(-)$ and $H^\bullet(-)$. The modules $H^d(K_*)$ and $H^d(K, K_*)$ are called the *dth absolute* (resp., *relative*) *persistent cohomology* of K_*.

The inclusion $i \colon K_* \hookrightarrow K$ induces an inclusion $i_\bullet \colon C_\bullet(K_*) \to C_\bullet(K)$ whose cokernel is $C_\bullet(K, K_*)$, and a restriction morphism $i^\bullet \colon C^\bullet(K) \to C^\bullet(K_*)$, $\gamma \mapsto \gamma|_{K_*}$ whose kernel equals $C^\bullet(K, K_*)$. There is a short exact sequence

$$0 \to C_\bullet(K_*) \xrightarrow{i_\bullet} \Delta C_\bullet(K) \xrightarrow{p_\bullet} C_\bullet(K, K_*) \to 0$$

that gives rise to a long exact sequence of P-persistence modules

$$\cdots \to H_{d+1}(K, K_*) \xrightarrow{\delta} H_d(K_*) \xrightarrow{i_d} \Delta H_d(K) \xrightarrow{p_d} H_d(K, K_*) \xrightarrow{\delta_d} H_{d-1}(K_*) \to \cdots . \tag{2.2}$$

Dually, the short exact sequence

$$0 \to C^\bullet(K, K_*) \xrightarrow{p^\bullet} \Delta C^\bullet(K) \xrightarrow{i^\bullet} C^\bullet(K_*) \to 0$$

gives rise to a long exact sequence

$$\cdots \to H^{d-1}(K_*) \xrightarrow{\delta^d} H^d(K, K_*) \xrightarrow{p^d} \Delta H^d(K) \xrightarrow{i^d} H^d(K_*) \xrightarrow{\delta^{d+1}} H^{d+1}(K, K_*) \to \cdots . \tag{2.3}$$

Remark 2.1.14. As mentioned above, we work with *reduced* (co)homology where not stated otherwise. In particular, if $H_d(K) = 0$ for all d, then the above exact sequences show that for all d, we have

$$H_d(K_*) \cong H_{d+1}(K, K_*), \qquad H^d(K_*) \cong H^{d+1}(K, K_*).$$

2.1.3 Free modules

Let M be a P-persistence module.

Definition 2.1.15. A *(homogeneous) generating set of* M is a collection $\{m_i \in M; i \in I'\}$ of elements of M, indexed by some indexing set I, such that for every $m \in M$, there exist coefficients $\lambda_i \in k$ for $i \in I$, such that only finitely many are non-zero and

$$m = \sum_{i \in I} \lambda_i M_{g(m), g(m_i)}(m_i).$$

The generating system is *minimal* if no proper subset of it is a generating system. It is called a *basis* of M if for all $p \in P$,

$$\sum_{i \in I} \lambda_i M_{p, g(m_i)}(m_i) = 0$$

implies that $\lambda_i = 0$ for all $i \in I$. A module is *finitely generated* if it has a finite generating system, and *free* if it has a basis.

A finitely generated module has a minimal generating system. If M is finitely generated (resp., free), then the multiset $\{g(m_i) | i \in I\}$ does not depend on the chosen minimal generating system (resp., basis) $\{m_i \mid i \in I\}$ of M. If M is free, we call the multiset $\{g(m_i) \mid i \in I\}$ the *graded rank* of M, denoted by $\operatorname{rk} M$.

For every $p \in P$, the persistence module $F(p)$ with

$$F(p)_q = \begin{cases} k \text{ if } p \le q, \\ 0 \text{ otherwise,} \end{cases} \qquad\qquad F(p)_{rq} = \begin{cases} \text{id if } p \le q \le r, \\ 0 \text{ otherwise.} \end{cases}$$

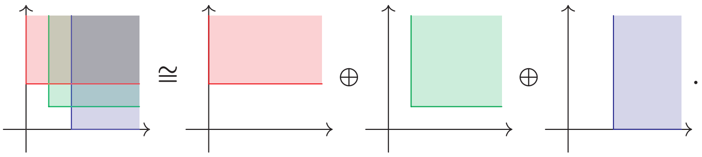
is free of rank $\operatorname{rk} F(p) = \{p\}$. If $p_i \in P$ for all $i \in I$ for some indexing set i, then $F := \bigoplus_{i \in I} F(p_i)$ is free of rank $\{p_i \mid i \in I\}$. The *standard basis* of F is the basis $\{e_i \mid i \in I\}$, where the i*th standard basis vector* e_i is the element $e_i = 1 \in F(p_i)_{p_i}$ of the ith summand. There is a one-to-one correspondence between generating systems (resp., bases) $\{m_i \mid i \in I\}$ of a module M and surjections (resp., isomorphisms)

$$\bigoplus_{i \in I} F(g(m_i)) \to M, e_i \mapsto m_i.$$

Example 2.1.16. The $\mathbf{Z}^2$-persistence module on the left of

$$\left[\begin{tikzcd}
\vdots & \vdots & \vdots & \\
k \arrow[u] \arrow[r, "\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)"] & k^2 \arrow[u] \arrow[r, "\left(\begin{smallmatrix}1&0\\0&1\\0&0\end{smallmatrix}\right)"] & k^3 \arrow[u] \arrow[r] & \cdots \\
0 \arrow[u] \arrow[r] & k \arrow[u, "\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)"] \arrow[r, "\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)"'] & k^2 \arrow[u, "\left(\begin{smallmatrix}0&0\\1&0\\0&1\end{smallmatrix}\right)"'] \arrow[r] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & k \arrow[u, "\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)"'] \arrow[r] & \cdots
\end{tikzcd}\right]
\cong
\left[\begin{tikzcd}
\vdots & \vdots & \vdots & \\
k \arrow[u, equal] \arrow[r] & k \arrow[u, equal] \arrow[r, equal] & k \arrow[u, equal] \arrow[r, equal] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & \cdots
\end{tikzcd}\right]
\oplus
\left[\begin{tikzcd}
\vdots & \vdots & \vdots & \\
0 \arrow[u] \arrow[r, equal] & k \arrow[u, equal] \arrow[r, equal] & k \arrow[u, equal] \arrow[r, equal] & \cdots \\
0 \arrow[u] \arrow[r] & k \arrow[u, equal] \arrow[r, equal] & k \arrow[u, equal] \arrow[r, equal] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & \cdots
\end{tikzcd}\right]
\oplus
\left[\begin{tikzcd}
\vdots & \vdots & \vdots & \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & k \arrow[u, equal] \arrow[r, equal] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & k \arrow[u, equal] \arrow[r, equal] & \cdots \\
0 \arrow[u] \arrow[r] & 0 \arrow[u] \arrow[r] & k \arrow[u, equal] \arrow[r, equal] & \cdots
\end{tikzcd}\right]$$

is a direct sum of the three free modules on the right. Because this is tedious to write and read, we usually draw two-parameter modules with diagrams such as



Thus, a module is free if it is a direct sum of modules whose support is an upper right quadrant.

Definition 2.1.17 (One-critical filtered complexes)**.** We call a P-filtered complex K_* *one-critical* if $\{p \in P \mid \sigma \in K_p\}$ has a single minimal element for every $\sigma \in K$, denoted by $g(\sigma)$ and called the *grade* of σ.

If P is totally ordered, then all every P-filtered simplicial complexes K_* with $\bigcap K_* = \emptyset$ is one-critical. If K_* is one-critical, then $C_d(K_*)$ is free. The *standard basis* of $C_d(K_*)$ is the basis $\{\sigma \mid \sigma \in K_*^d\}$.

Definition 2.1.18 (P-graded matrix)**.** Let P be a poset. A *P-graded $m \times n$-matrix* consists of an ordinary $m \times n$-matrix $\mathsf{u}(M)$ with entries in k, called its *underlying matrix*, and two tuples $\mathrm{rg}^M \in P^m$ and $\mathrm{cg}^M \in P^n$, called the *row* and *column grades* of M. In this case, we also say that M is a graded $\mathrm{rg}^M \times \mathrm{cg}^M$-matrix. It is called *valid* if $M_{ij} \neq 0$ only if $\mathrm{rg}_i^M \leq \mathrm{cg}_j^M$. The *sum* $M + M'$ of two graded matrices is defined if $\mathrm{rg}^M = \mathrm{rg}^{M'}$ and $\mathrm{cg}^M = \mathrm{cg}^{M'}$. In this case, it has $\mathrm{rg}^{M+M'} = \mathrm{rg}^M = \mathrm{rg}^{M'}$ and $\mathrm{cg}^{M+M'} = \mathrm{cg}^M = \mathrm{cg}^{M'}$. The *product* MM' of two graded matrices is defined if $\mathrm{cg}^M = \mathrm{rg}^{M'}$. In this case, $\mathrm{rg}^{MM'} = \mathrm{rg}^M$ and $\mathrm{cg}^{MM'} = \mathrm{cg}^{M'}$. The *graded transpose* of M is the graded $n \times m$-matrix $M^\top$ with row grades $\mathrm{rg}_i^{M^\top} = -\mathrm{cg}_i^M$, column grades $\mathrm{cg}_j^{M^\top} = -\mathrm{rg}_j^M$ and entries $[M^\top]_{ij} = [M]_{ji}$.

Remark 2.1.19. A graded matrix is valid if and only if its transpose is valid.

Remark 2.1.20. For algorithmic purposes, e.g., when maintaining a certain order (such as lexicographic or colexicographic order) on the row and column grades, it may be convenient to use the graded *anti-transpose* $M^\top$ with row grades $\mathrm{rg}_i^{M^\perp} = -\mathrm{cg}_{n+1-i}^M$, column grades $\mathrm{cg}_j^{M^\perp} = -\mathrm{rg}_{m+1-j}^M$ and entries $[M^\perp]_{ij} = [M]_{m+1-j,n+1-i}$ instead of the graded transpose.

The following follows from the fact that $\mathrm{Hom}(F(q), F(p)) \cong \left\{\begin{smallmatrix} k & \text{if } p \leq q, \\ 0 & \text{otherwise} \end{smallmatrix}\right.$:

Lemma 2.1.21. *Let F and F' be free persistence modules with bases $(b_j)_{j=1}^n$ and $(b_i')_{i=1}^m$, respectively. Then there is a one-to-one correspondence between morphisms $f\colon F \to F'$ and valid graded $(g(b_i'))_i \times (g(b_j))_j$-matrix. The morphism f corresponds to the matrix M with entries*

$$f(b_j) = \sum_{g(b_i') \leq g(b_j)} M_{ij} F'_{g(b_j),g(b_i')}(b_i')$$

for all $j \leq n$.

In this case, we identify valid graded $(g(b'_i))_i \times (g(b_j))_j$-matrices and morphisms from F to F' without making the distinction explicit.

Definition 2.1.22. A module $M \in \mathrm{Vec}^P$ is *projective* (resp., *injective*) if the contravariant functor $\mathrm{Hom}_{\mathrm{Vec}^P}(-, M)\colon \mathrm{Vec}^P \to \mathrm{Vec}$ (resp., the covariant functor $\mathrm{Hom}_{\mathrm{Vec}^P}(M, -)\colon \mathrm{Vec}^P \to \mathrm{Vec}$) is exact.

Equivalently, a module M is projective (resp., injective) if and only if every short exact sequence of modules of the form $0 \to A \to B \to M \to 0$ (resp., $0 \to M \to A \to B \to 0$) splits.

Proposition 2.1.23 ([82, Proposition 5])**.** *Every projective module $M \in \mathrm{Vec}^P$ is free.*

This is a generalization of Kaplansky's theorem [7, 83] and the Quillen–Suslin-theorem [113, 120].

2.2 One-parameter persistence

Let $P = \mathbf{Z}$ or $P = \mathbf{R}$. For $-\infty \le b < d \le \infty$, we let $I(b, d) \in \mathrm{Vec}^P$ be the *interval module* with

$$I(b,d)_p = \begin{cases} k & \text{if } b \le p < d, \\ 0 & \text{otherwise,} \end{cases} \qquad I(b,d)_{qp} = \begin{cases} \mathrm{id} & \text{if } b \le p \le q < d, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, $I(b, \infty) = F(b)$ for $b > -\infty$.

Theorem 2.2.1 (Structure Theorem [26, 52, 126], see also [70])**.** *If $M \in \mathrm{vec}^P$ for $P = \mathbf{Z}$ or $P = \mathbf{R}$, then*

$$M \cong \bigoplus_{i \in I} I(b_i, d_i)$$

for a uniquely determined finite indexing set I and $-\infty \le b_i < d_i \le \infty$ for all $i \in I$.

The multiset $\operatorname{barc} M = \{(b_i, d_i) \mid i \in I\}$ is called the *barcode* of M. A barcode is commonly depicted by a *persistence diagram* plotting the pairs (b_i, d_i) as points; see Figure 2.3b. Pairs $(b_i, d_i) \in \operatorname{barc} M$ are called *essential* if $d_i = \infty$ and *finite* or *non-essential* otherwise. Thus, the essential pairs of $\operatorname{barc} M$ correspond to a basis of the vector space $\operatorname{colim} M$.

From now on, let $P = \mathbf{Z}$. Persistence modules over $\mathbf{Z}$ are equivalent to graded modules over the principal ideal domain (PID) $k[x]$ (see Proposition 2.3.10). Therefore, if M is finitely generated, then Theorem 2.2.1 is a graded version of the invariant factor decomposition of finitely generated modules over PIDs [128, Theorem 2.1], which is essentially a manifestation of Gabriel's theorem [74].

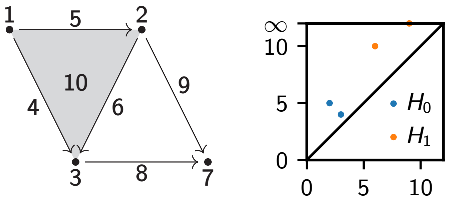
Let $C_\bullet$ be a chain complex of finite rank free persistence modules. Then $H_d(C_\bullet) \in \mathrm{vec}^{\mathbf{Z}}$ for each d, and we consider $\operatorname{barc} H_d(C_\bullet)$. Choose bases of C_d for each d, and let D_d be the graded matrix representing the boundary morphism ∂_d of $C_\bullet$ with respect to the standard basis. Since $k[x]$ is a PID, every $\mathbf{Z}$-graded matrix has a Smith normal form, and $\operatorname{barc} H_d(C_\bullet)$ can be determined from the graded Smith normal forms of D_d and D_{d-1} [128]. In practical computations, it is more convenient to work with the *Standard Algorithm* instead, which we recall now.

Definition 2.2.2. The *pivot* of a column M_j of a matrix M is $\operatorname{piv} M_j \coloneqq \max\{i \mid M_{ij} \ne 0\}$. A matrix M is *reduced* if all its non-zero columns have distinct pivots.

Let $C_\bullet = \bigoplus_{i=1}^N F(z_i)$ be a chain complex of free $\mathbf{Z}$-persistence modules of finite rank, with $z_i \le z_j$ if $i \le j$, and D_d be the matrix representing ∂_d with respect to the standard basis.

Proposition 2.2.3 (Persistence Pairing [64, 128])**.** *If V_d be an invertible valid graded matrix such that $R_d \coloneqq D_d V_d$ is reduced for all d, then*

$$\begin{aligned} \operatorname{barc} H_d(C_\bullet) = \{(z_i, z_j) \mid [R_d]_i = 0 \textit{ and } \quad & i = \operatorname{piv}[R_{d+1}]_j\} \\ \cup \{(z_i, \infty) \mid [R_d]_i = 0 \textit{ and } \nexists j\colon & i = \operatorname{piv}[R_{d+1}]_j\}. \end{aligned} \tag{2.4}$$



(a) K_* **(b)** barc $H_\bullet(K_*)$

Figure 2.3: Left: simplex-wise $\mathbf{Z}$-filtration of a simplicial complex K_* The numbers indicate the value in $\mathbf{Z}$ at which a simplex enters the filtration. Right: Persistence diagram showing the barcode of the reduced persistent homology of K_*. Each point (x, y) in the diagram denotes a pair $(x, y) \in \operatorname{barc} H_\bullet(K_*)$.

The matrices $R_\bullet$ and $V_\bullet$ can be computed from $D_\bullet$ using the Standard Algorithm; see Algorithm 1. If $[R_d]_i = 0$, then i is called a *birth index* of R_d; otherwise, i is called a *death index*. Every column index i of R_d is either birth or death. To see this, note that every submodule of a free one-parameter persistence module of finite rank is free. In particular, $0 \to Z_\bullet(C_\bullet) \to C_\bullet \to B_\bullet(C_\bullet) \to 0$ is an exact sequence of free modules and thus splits. Assuming that all columns of R_d have distinct grade, this splitting is uniquely determined and fixes a partition of the basis of $C_\bullet$ into birth and death indices.

Lemma 2.2.4. *If R_d, R_{d+1} are as in Proposition 2.2.3, then $[R_d]_i = 0$ whenever $i = \operatorname{piv}[R_{d+1}]_j$. In particular, we can simplify* (2.4) *to*

$$\begin{aligned}\operatorname{barc} H_d(C_\bullet) = \{&(z_i, z_j) \mid i = \operatorname{piv}[R_{d+1}]_j\} \\ \cup\, \{&(z_i, \infty) \mid [R_d]_i = 0 \textit{ and } \nexists j\colon i = \operatorname{piv}[R_{d+1}]_j\}.\end{aligned}$$

Example 2.2.5. Consider the filtered complex K_* in Figure 2.3. We compute a barcode for its reduced homology. Its augmented chain complex has the graded boundary matrices

$$D_0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 7 \\ \hline 1 & 1 & 1 & 1 & 1 \end{array}, \qquad D_1 = \begin{array}{c|ccccc} & \bar{4} & \bar{5} & \bar{6} & \bar{8} & \bar{9} \\ \hline 1 & \bar{1} & \bar{1} & 0 & 0 & 0 \\ 2 & 0 & 1 & \bar{1} & 0 & \bar{1} \\ 3 & 1 & 0 & 1 & \bar{1} & 0 \\ 7 & 0 & 0 & 0 & 1 & 1 \end{array}, \qquad D_2 = \begin{array}{c|c} & 10 \\ \hline 4 & \bar{1} \\ 5 & 1 \\ 6 & 1 \\ 8 & 0 \\ 9 & 0 \end{array},$$

where we write $\bar{1}$ for -1. The Standard Algorithm computes the matrices

$$V_0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 7 \\ \hline 1 & 1 & \bar{1} & 0 & 0 \\ 2 & & 1 & \bar{1} & 0 \\ 3 & & & 1 & \bar{1} \\ 7 & & & & 1 \end{array}, \qquad V_1 = \begin{array}{c|ccccc} & 4 & 5 & 6 & 8 & 9 \\ \hline 4 & 1 & 0 & \bar{1} & 0 & \bar{1} \\ 5 & & 1 & 1 & 0 & 1 \\ 6 & & & 1 & 0 & 0 \\ 8 & & & & 1 & \bar{1} \\ 9 & & & & & 1 \end{array}, \qquad V_2 = \begin{array}{c|c} & 10 \\ \hline 10 & 1 \end{array},$$

$$R_0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 7 \\ \hline 1 & 1 & 0 & 0 & 0 \end{array}, \qquad R_1 = \begin{array}{c|ccccc} & 4 & 5 & 6 & 8 & 9 \\ \hline 1 & \bar{1} & \bar{1} & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & \bar{1} & 0 \\ 7 & 0 & 0 & 0 & 1 & 0 \end{array}, \qquad R_2 = \begin{array}{c|c} & 10 \\ \hline 4 & \bar{1} \\ 5 & 1 \\ 6 & 1 \\ 8 & 0 \\ 9 & 0 \end{array}.$$

According to (2.4), we obtain

$$\operatorname{barc} H_0(K_*) = \{(2,5), (3,4), (7,8)\}, \qquad \operatorname{barc} H_1(K_*) = \{(6,10), (9,\infty)\},$$

and no non-zero higher dimensional persistent homology modules; see Figure 2.3.

Definition 2.2.6 (Persistence basis)**.** Let $C_\bullet$ be a chain complex of finite rank free modules, and assume for a d that $\operatorname{barc} H_d(C_\bullet) = \{(b_i, d_i) \mid i \in I\}$ for some indexing set I. We let $I_{\mathrm{f}} = \{i \in I \mid d_i < \infty\}$ and $I_{\mathrm{e}} = I \setminus I_{\mathrm{f}}$. A *persistence basis* of $H_d(C_\bullet)$ is a system

$$\{(z_i, c_i) \mid i \in I_{\mathrm{f}}\} \cup \{z_i \mid i \in I_{\mathrm{e}}\}, \tag{2.5}$$

where $z_i \in Z_d(C_\bullet)$ and $c_i \in C_{d+1}$, such that for $i \in I_{\mathrm{f}}$, we have $I(b_i, d_i)_{d_i, b_i}(z_i) = \partial c_i$, and such that the assignment

$$\begin{aligned}\bigoplus_{i \in I} I(b_i, d_i) &\longrightarrow H_d(C_\bullet), \\ e_i &\longmapsto [z_i]\end{aligned}$$

is an isomorphism, where e_i denotes the generator of the interval module indexed by i. The pairs in the left set in (2.5) are called *persistence pairs*, the cycles in the second pairs are representatives of the *essential homology classes*.

Example 2.2.7. If $R_\bullet$ and $V_\bullet$ are as in Proposition 2.2.3, then

$$\{([R_{d+1}]_j, [V_{d+1}]_j) \mid [R_{d+1}]_j \neq 0\} \cup \{[V_d]_i \mid [R_d]_i = 0 \text{ and } \nexists j : i = \operatorname{piv}[R_{d+1}]_j\}$$

represent a persistence basis of $H_d(C_\bullet)$.

Algorithm 1 has complexity $\mathcal{O}(N^3)$ [128], which can be attained even if $C_\bullet = C_\bullet(K_*)$ for a filtered simplicial complex K_* [105], although worst-case complexity is rarely observed in practice; see also [78]. If $K = VR(S)$ is the Vietoris–Rips complex of a finite metric space S, then $|K^d| = \binom{|S|}{d+1}$. Therefore, computing the barcode of $H_d(\mathsf{VR}_*(S))$ becomes impractical already for relatively small values of $|S|$. The following standard approach remedies this shortcoming.

Algorithm 1: Standard algorithm for one-parameter persistent homology.

```
Input:   A graded $m \times n$-matrix $D$, rows and columns sorted by grade.
Output:  A $\mathbf{Z}$-graded invertible $n \times n$-matrix $V$ such that $DV$ is reduced.
function Reduce($D$):
    $p \leftarrow (0, \dots, 0) \in \mathbf{N}^n$
    $V \leftarrow 1 \in k^{n \times n}$ with $\mathrm{rg}^V = \mathrm{cg}^V = \mathrm{cg}^D$        ▷ graded unit matrix
    foreach $j = 1, \dots, n$ do
        while $\mathrm{piv}[D]_j \neq 0$ do
            $i \leftarrow \mathrm{piv}[D]_j$
            $k \leftarrow p_i$
            if $k \neq 0$ then
                $[D]_j \leftarrow [D]_j - [D]_{ij}/[D]_{ik}[D]_k$
                $[V]_j \leftarrow [V]_j - [D]_{ij}/[D]_{ik}[V]_k$
            else $p_i \leftarrow j$; break

    return $V$
```

2.2.1 Relative cohomology and clearing

As before, let K_* be a finite one-critical $\mathbf{Z}$-filtered simplicial complex. Consider the relative cochain complex $C^\bullet(K, K_*)$. Because $C^\bullet(K, K_*)$ is the kernel of the restriction $\Delta C^\bullet(K) \to C^\bullet(K_*)$, each vector space $C^\bullet(K, K_z)$ is spanned by the dual simplices σ^* with $\sigma \notin K_z$. Therefore,

$$C^\bullet(K, K_*) \cong \bigoplus_{\sigma \in K} F(-g(\sigma) + 1)$$

is a cochain complex of free $\mathbf{Z}$-persistence modules; see Figure 1.5. With respect to the standard basis of $C_\bullet(K, K_*)$, the coboundary operator $\delta^d \colon C^{d-1}(K, K_*) \to C^d(K, K_*)$ is represented by the graded matrix $D^d := D_d^\top$.

Remark. We assumed that the rows and columns of D_d are ordered such that their grades are monotonically increasing. The reversal of the row and column order in Definition 2.1.18 ensures that D^d has the same property.

One can compute the reduced form $R^d = D^d V^d$ from D^d using Algorithm 1, and, one obtains the barcode of $H^{d+1}(K, K_*)$ from R^d and R^{d+1} analogously to Proposition 2.2.3. The following is dual to Lemma 2.2.4

Algorithm 2: A variant of Algorithm 1 that incorporates clearing.

```
Input:  A valid graded $l \times m$-matrix $D$ and a reduced valid graded $m \times l$-matrix $R$ such that $DR = 0$;
        both with rows and columns sorted by grade.
Output: A $\mathbf{Z}$-graded invertible $n \times n$-matrix $V$ such that $DV$ is reduced.
function ReduceC($D$, $R'$):
    $p \leftarrow (0, \dots, 0) \in \mathbf{N}^m$
    $V \leftarrow 1 \in k^{n \times n}$ with $\mathrm{rg}^V = \mathrm{cg}^V = \mathrm{cg}^D$
    for $j = 1, \dots, n$ do                                          ▷ clearing
        if $[R'_d] \neq 0$ then
            $[D]_{\mathrm{piv}[R_d]} \leftarrow 0$
            $[V]_{\mathrm{piv}[R_d]} \leftarrow [R_d]$
    for $j = 1, \dots, m$ do
        while $\mathrm{piv}[D]_j \neq 0$ do
            $i \leftarrow \mathrm{piv}[D]_j$
            $k \leftarrow p_i$
            if $k \neq 0$ then
                $[D]_j \leftarrow [D]_j - [D]_{ij}/[D]_{ik}[D]_k$
                $[V]_j \leftarrow [V]_j - [D]_{ij}/[D]_{ik}[V]_k$
            else $p_i \leftarrow j$; break
    return $V$
```

Lemma 2.2.8. *Let $R^d = D^d V^d$ and $R^{d-1} = D^{d-1}V^{d-1}$ be reduced, where V^d and V^{d-1} are invertible upper triangular matrices. Then $[R^d]_i = 0$ whenever $i = \mathrm{piv}[R^{d-1}]_j$. In particular,*

$$\begin{aligned}\operatorname{barc} H^{d+1}(K, K_*) = \{(z_i, z_j) \mid i = \mathrm{piv}[R^d]_j\} \\ \cup \{(z_i, \infty) \mid [R^{d+1}]_i = 0 \text{ and } \nexists j \colon i = \mathrm{piv}[R^d]_j\}.\end{aligned}$$

If Algorithm 1 is applied to R_d or R^d, it will spend most of its run time on those columns of R_d (resp., R^d) that are reduced to zero. Experience shows that if a column of R_d (resp., R^d) is not reduced to zero, it will typically reach its final state already after relatively few additions. If computing the (co)homology of Vietoris–Rips complexes, this is true in particular for cohomology, because in this case, R^d has few columns and many rows, so chances are low that two columns have the same pivot.

Looking again at Lemmas 2.2.4 and 2.2.8, we see that it is not necessary in all cases to reduce all columns of R_d (resp., R^d). Namely, if $i = \mathrm{piv}[R_{d+1}]_j$ (resp. $i = \mathrm{piv}[R^{d-1}]_j$) for some j, then we know that Algorithm 1 will result in $[R_d]_i = 0$ (resp. $[R^d]_i = 0$). We can therefore skip reducing these columns and set $[R_d]_i \coloneqq 0$, $[V_d]_i \coloneqq [R_{d+1}]_j$ (resp. $[R^d]_i \coloneqq 0$, $[V^d]_i \coloneqq [R^{d-1}]_j$) immediately. This optimization scheme is called *clearing* or the *twist algorithm* [11, 46]; see Algorithm 2. It has a great share in the efficiency of implementations such as [1, 9, 111].

We note that for Vietoris–Rips complexes, clearing is effective only when computing the barcode of $C^\bullet(K, K_*)$, rather than the barcode of $C_\bullet(K_*)$ [12, 9, p. 403]. Namely, when reducing $D_1, \dots, D_{d_{\max}}$, we have to compute $R_{d_{\max}}$ using Algorithm 1, before we can compute $R_{d_{\max}-1}, \dots, R_1$ using Algorithm 2. For Vietoris–Rips complexes, applying Algorithm 1 to $D_{d_{\max}}$ will dominate the run time. For cohomology, however, we may compute R^1 using Algorithm 1, and then use Algorithm 2 to compute $R^2, \dots, R^{d_{\max}}$. As D^1 is very small compared to the higher dimensional coboundary matrices, this is much more efficient. Furthermore, there are specialized algorithms to compute the barcode of $H^0(K, K_*)$ more efficiently [9].

It remains to relate the barcodes of $H^\bullet(K, K_*)$ and $H_\bullet(K_*)$. The first step follows from $H^\bullet((C_\bullet)^*) \cong H_\bullet(C_\bullet)^*$:

Lemma 2.2.9. *If $M \in \mathrm{vec}^{\mathbf{Z}}$, then* $\operatorname{barc} M^* = \{(-d, -b) \mid (b, d) \in \operatorname{barc} M\}$.

Next, we relate relative and absolute homology. Recall the long exact homology sequence

$$\cdots \to H_{d+1}(K, K_*) \xrightarrow{\partial_{d+1}} H_d(K_*) \xrightarrow{i_d} \Delta H_d(K) \xrightarrow{p_d} H_d(K, K_*) \xrightarrow{\partial_d} H_{d-1}(K_*) \to \cdots .$$

Proposition 2.2.10 ([19])**.** *For every d, the induced short exact sequences*

$$0 \to \operatorname{coker} p_{d+1} \to H_d(K_*) \to \ker p_d \to 0 \tag{2.6}$$
$$0 \to \operatorname{coker} i_{d+1} \to H_{d+1}(K, K_*) \to \ker i_d \to 0 \tag{2.7}$$

split.

Proof (sketch). For any module M, let

$$\operatorname{barc}_{-\infty} M := \{(b, d) \in \operatorname{barc} M \mid b = -\infty\}, \qquad \operatorname{barc}^{\infty} M := \{(b, d) \in \operatorname{barc} M \mid d = \infty\}.$$

The barcode $\operatorname{barc} \Delta H_d(K)$ consists exclusively of bars $(-\infty, \infty)$. Therefore, the barcode of any submodule of $\Delta H_d(K)$ contains only bars of the form (b, ∞) for some b. This implies that

$$\operatorname{barc} \ker p_d \subseteq \operatorname{barc}^{\infty} H_d(K_*), \tag{2.8}$$

so $\ker p_d$ is free; hence (2.6) splits. Likewise, the barcode of any quotient of $\Delta H_{d+1}(K)$ contains only bars of the form $(-\infty, d)$ for some d. This implies that

$$\operatorname{barc} \operatorname{coker} i_{d+1} \subseteq \operatorname{barc}_{-\infty} H_{d+1}(K, K_*), \tag{2.9}$$

so $\operatorname{coker} i_{d+1}$ is injective; hence (2.7) splits. See [19] for details. □

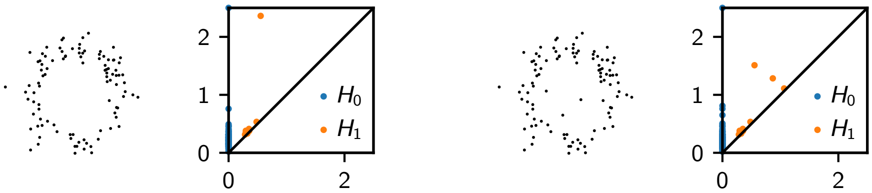
One may also check that $\operatorname{barc} H_{d+1}(K, K_*)$ and thus also $\operatorname{barc} \operatorname{coker} p_{n+1}$ contains no bars of the form (b, ∞) for any b. Similarly, $\operatorname{barc} H_d(K_*)$ and thus also $\operatorname{barc} \ker i_d$ contains no bars of the form $(-\infty, d)$ for any d. Exactness of (2.6) and (2.7) implies that (2.8) and (2.9) are equalities.

Corollary 2.2.11 ([55, §2.4])**.** *Let K_* be a finite one-critical $\mathbf{Z}$-filtered simplicial complex. For all d,*

$$\begin{aligned} \operatorname{barc} H_d(K_*) = {} & \{(b, d) \in \operatorname{barc} H_{d+1}(K, K_*) \mid -\infty < b\} \\ & \cup \{(d + 1, \infty) \mid (-\infty, d) \in H_d(K, K_*)\}, \end{aligned} \tag{2.10}$$
$$\begin{aligned} \operatorname{barc} H_{d+1}(K, K_*) = {} & \{(b, d) \in \operatorname{barc} H_d(K_*) \mid d < \infty\} \\ & \cup \{(-\infty, b - 1) \mid (b, d) \in H_{d+1}(K_*)\}. \end{aligned} \tag{2.11}$$

Therefore, the $\mathbf{Z}$-persistence modules $\bigoplus_d H_d(K_*)$ and $\bigoplus_d H_d(K, K_*)$ determine each other uniquely even if K is not acyclic. We can convert a barcode of $H^\bullet(K, K_*)$ (which is efficiently computable using Algorithm 2) to a barcode of $H_\bullet(K_*)$ using (2.10). This is also the approach followed in [9].

Remark. When looking at (2.10), it might seem surprising that $H_\bullet(K_*)$ can be determined more efficiently by computing $H^\bullet(K, K_*)$, given that one needs a barcode of $H^{d+1}(K, K_*)$ to determine the barcode of $H_d(K_*)$. It seems that one needs to compute the reduced matrix R^{d+1} to obtain $\operatorname{barc} H^{d+1}(K, K_*)$, which would be unfeasible for Vietoris–Rips complexes. A closer look at (2.10) reveals that only the *finite* bars of $\operatorname{barc} H^{d+1}(K, K_*)$ are needed to determine $\operatorname{barc} H_d(K_*)$. According to Lemma 2.2.8, these can be determined from R^d.



(a) without outliers

(b) with outliers

Figure 2.4: Persistence diagram of the Vietoris–Rips complex of a point cloud sampled from an annulus **(a)** without and **(b)** with three outliers outliers. The two point clouds are the same as in Figures 1.2 and 1.3.

2.3 Multi-parameter persistence

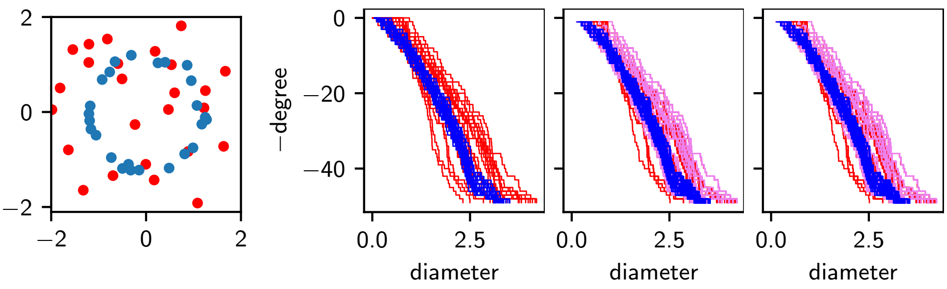
Multi-parameter persistence is the theory of $\mathbf{Z}^n$- or $\mathbf{R}^n$-persistence modules for $n > 1$. To motivate multi-parameter persistence, recall the stability results of one-parameter persistence. These assert that: a) for modules, the assignment $M \mapsto \operatorname{barc} M$ is an isometry [15, 28, 40, 96]; b) for a triangulable space X, the assignment $\{\mathbf{R}\text{-valued continuous functions on } X\} \to \mathrm{Vec}^{\mathbf{R}}$, $f \mapsto H_d(X_{f\leq *})$ is Lipschitz with respect to the supremum norm and the interleaving distance on persistence modules [48]; and c) for metric spaces X, the assignments $X \mapsto H_d(\mathit{VR}_*(X))$ and $X \mapsto H_d(\check{C}_*(X))$ are Lipschitz with respect to the Gromov–Hausdorff distance and the interleaving distance [38, 41, §5.2].

Despite these stability results, however, one-parameter persistent homology is known to be susceptible to outliers [23]. For example, Figure 2.4 shows that already adding very few outliers inside the annulus drastically changes the persistence diagram. In particular, while the the persistence diagram in **(b)** exhibits the actual homology of the annulus, this is not the case for the persistence diagram in **(a)**, Figure 1.2 shows the persistence diagrams of the Delaunay-filtration for the same point cloud.

According to [24, §1.7], common approaches to address this in the framework of one-parameter persistence are removing points that have density below a fixed threshold [32], considering the filtration by density for a fixed scale parameter [44, 45], or the development of new filtrations robust to outliers [25, 29, 39, 42, 112], most of which rely on fixing a density, scale or bandwidth parameter, which arguably contradicts the explorative approach of persistent homology. A natural remedy for this is to introduce one or more additional parameters to control for these properties, which leads to the notion of multi-parameter persistent homology. A common choice is to extend constructions such as the α-complex, the Čech complex and the Vietoris–Rips complex by a second parameter that controls for the local density of the point cloud.

Example 2.3.1 (Function-Rips bifiltration). Let X be a metric space and $f\colon X \to \mathbf{R}$ be a continuous function. The *function-Rips bifiltration* $\mathit{VR}_*(X, f)$ is the $\mathbf{R}^2$-filtered simplicial complex with $\mathit{VR}_{r,s}(X, f) \coloneqq \mathit{VR}_r(X_{f\leq s})$. It is a one-critical bifiltration. If $f(v)$ measures the density of S around v, we call $\mathit{VR}_*(X, f)$ also the *density-Rips bifiltration*.

Example 2.3.2 (Degree-bifiltrations [24, §2.3]). The *degree* of a vertex v in a simplicial complex K is the number of edges incident to v. For a P-filtered simplicial complex K_*, let $D_*(K_*)$ be the $\mathbf{Z} \times P$-filtered simplicial complex such that $D_k(K_p)$ is the maximal subcomplex of K_p on the vertices of degree at least $-k-1$. In particular, we define the *degree–Rips bifiltration* $D_*(\mathit{VR}_*(S))$ and *degree-Čech bifiltration* $D_*(\check{C}_*(S))$, which are bifiltrations over $\mathbf{R} \times \mathbf{Z}$. These need not be one-critical in general. For example, Figure 2.5a contains $N = 51$ points sampled from a unit circle, with a single outlier in the origin. Every simplex has a curve describing the pairs (–degree, diameter) at which the simplex enters the filtration; see Figure 2.5. For the vertices, each of these curves has N steps; one for each edge connecting the vertex to one other simplex.



(a) Point cloud **(b)** 0-simplices **(c)** 1-simplices **(d)** 2-simplices

Figure 2.5: Generator curves of a degree-Rips bifiltration. **(a)** The underlying point cloud consists of 50 points sampled from a perturbed unit circle (blue), plus 25 additional points sampled uniformly from the square (red). **(b)–(d)** Generator curves for the 0-, 1- and 2-simplices of the associated degree-Rips complex. The generator curve for simplices spanned by "red" vertices are red, curves for simplices involving "blue" and "red" samples are drawn magenta.

Another common bifiltrations of spaces is the *multi-cover bifiltration* [118]. Other common bifiltered simplicial complexes are the *subdivision–Rips* and *subdivision-Čech bifiltration* and the the *rhomboid tilings* [51, 62, 63]. Multi-parameter persistence modules can be equipped with an interleaving metric [96], and under suitable conditions, assigning to a finite metric space its multi-cover bifiltration, subdivision- or degree-Rips or -Čech bifiltration is Lipschitz with respect to the (Gromov–)Prokhorov and the (generalized, homotopy-)interleaving distance [24, theorem 1.6, 1.7, 116].

Besides remedying the impact of outliers, multi-parameter persistent homology has been used in image classification [35], analysis of time-dependent spaces [88], time series [87] and clustering [116].

2.3.1 Invariants of multi-parameter persistence modules

In general, multi-parameter persistence modules do not admit a classification as simple as the barcode in one-parameter persistence. To explain this, we note that by virtue of Theorem 2.2.1, the barcode of a one-parameter persistence module is a way to describe its decomposition into indecomposable direct summands. For any P, a module in $M \in \mathrm{vec}^P$ is *indecomposable* if $M = M' \oplus M''$ implies that one of M', M'' is zero. Every module $M \in \mathrm{vec}^P$ is isomorphic to a direct sum of indecomposable modules that have local endomorphism rings [26]. The Krull–Remak–Schmidt–Azumaya Theorem [5, thm. 1.(ii), 68, thm. 2.12] implies that the decomposition of M is essentially unique. For $P = \mathbf{Z}$, the indecomposable modules in $\mathrm{vec}^{\mathbf{Z}}$ are precisely the interval modules $I(b, d)$ for $-\infty \leq b < d \leq \infty$. For $n > 1$, the indecomposable $\mathbf{Z}^n$-persistence modules cannot be classified by such a simple structure. In particular, an indecomposable $\mathbf{Z}^n$-persistence modules need not be determined by its support.

Example 2.3.3. The $\mathbf{Z}^2$-persistence module

$$\begin{tikzcd}
\vdots \arrow[d, equal] & \vdots \arrow[d, equal] & \vdots \arrow[d, equal] & \\
k \arrow[r, "\binom{1}{0}"] & k^2 \arrow[r, equal] & k^2 \arrow[r, equal] & \cdots \\
0 \arrow[r] \arrow[u] & k \arrow[r, "\binom{a}{b}"] \arrow[u, "\binom{\alpha}{\beta}"] & k^2 \arrow[r, equal] \arrow[u, equal] & \cdots \\
0 \arrow[r] \arrow[u] & 0 \arrow[r] \arrow[u] & k \arrow[r, equal] \arrow[u, "\binom{0}{1}"] & \cdots
\end{tikzcd}$$

is indecomposable for every $a, b \neq 0$, because it contains the indecomposable representation

$$M = \left(\begin{tikzcd} k \arrow[rr, "\binom{1}{0}"] & & k^2 \\ & k \arrow[ur, "\binom{a}{b}"'] & \\ & & k \arrow[u, "\binom{0}{1}"'] \end{tikzcd} \right) \text{ of the quiver } Q = \left(\begin{tikzcd} u \arrow[rr] & & x \\ & v \arrow[ur] & \\ & & w \arrow[u] \end{tikzcd} \right)$$

One way to see that M is indecomposable is by listing all isomorphism classes of submodules and quotients of M and observing that no non-trivial submodule of M is isomorphic to a quotient of M. Another way to see this is by considering the endomorphism ring of M. Explicitly, $\operatorname{End} M$ can be seen as the set of those matrices

$$\begin{array}{c|ccccc} & M_u & M_v & M_w & M_x & \\ \hline M_u & * & & & & \\ M_v & & * & & & \\ M_w & & & * & & \\ M_x & & & & * & * \\ & & & & * & * \end{array} \in k^{5\times 5}$$

that commute with the matrices

$$\begin{pmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ 1 & & & 0 & 0 \\ 0 & & & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & a & & 0 & 0 \\ & b & & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & & & & \\ & 0 & & & \\ & & 0 & & \\ & & 0 & 0 & 0 \\ & & 1 & 0 & 0 \end{pmatrix}$$

corresponding to the structure maps of M. This defines a linear system. Solving this shows that $\operatorname{End} M \cong k$ unless $a = 0$ or $b = 0$. In particular, $\operatorname{End} M$ is a local ring unless $a = 0$ or $b = 0$, hence M is indecomposable by the Azumaya–Krull–Remak–Schmidt theorem [5, 26, §2, 68]. In particular, this gives a family of indecomposable $\mathbf{Z}^2$-persistence modules parametrized by $k\mathbf{P}^1 \setminus \{[0:1], [1:0]\}$. This implies that there is no discrete invariant (in the sense of [34]) classifying all indecomposable persistence modules.

Gabriel's theorem [74] implies that there exist infinitely many indecomposable $\{1, \ldots, k\}^n$-persistence modules for any $k \geq 1$ and $n > 1$. In fact, for any $n > 1$, there are indecomposable $\mathbf{Z}^n$-persistence modules of arbitrarily large dimension [30]. For multi-parameter persistence modules, instead of a barcode, one seeks to define and compute other meaningful invariants from a module $M \in \operatorname{vec}^{\mathbf{R}^n}$ or $\operatorname{vec}^{\mathbf{Z}^n}$, such as the graded Betti diagrams $\beta_i(M)$ (see Definition 2.3.6), the *Hilbert function* $\mathbf{Z}^n \to \mathbf{N}, p \mapsto \dim M_p$, Hilbert series, associated primes and local cohomology [80], the *rank invariant* $\{(p, q) \in (\mathbf{Z}^n)^2 \mid p \leq q\} \to \mathbf{N}, (p, q) \mapsto \operatorname{rk} M_{qp}$ for $p \leq q$ [34], the generalized rank invariant [56, 88, 90, 109], the signed barcode [27], the fibered barcode [22, 100, 123], the shift dimension [37, 75], or various others.

Of particular interest is computing a minimal free resolution or a minimal free presentation of a given finitely generated persistence module $M \in \operatorname{vec}^{\mathbf{Z}^n}$, because finitely generated modules are in bijection with isomorphism classes of minimal free resolution. Additionally, several of the above invariants can computed from a minimal free resolution of M. Therefore, it is a natural question how to compute a minimal free resolution of $H_\bullet(K_*)$ for a finite $\mathbf{Z}^n$-filtered simplicial complex K_*.

Definition 2.3.4. A *free presentation* of a persistence module M is a morphism $\partial\colon F_1 \to F_0$ of free modules such that $M \cong \operatorname{coker} \partial$. It is called *finite* if F_1, F_0 have finite rank. A *free resolution* of a persistence module M is a chain complex $F_\bullet$ of free modules concentrated in non-negative degrees such that there is an exact sequence

$$\cdots \to F_1 \to F_0 \xrightarrow{f_0} M \to 0 \tag{2.12}$$

It is called *finite* if all F_i have finite rank, and only finitely many F_i are non-zero. The *length* of a finite free resolution $F_\bullet$ is the smallest integer ℓ such that $F_i = 0$ for all $i > \ell$. A *morphism of free resolutions* is a morphism of chain complexes. This renders $\operatorname{Ch}(\operatorname{Vec}^P)$ an abelian category. The morphism f_0 is called the *augmentation map* of the resolution, and the sequence in (2.12) is sometimes called the *augmented resolution*; usually, however, by a slight abuse of language that should not confuse the reader, the sequence (2.12) is also called a free resolution of M.

Example 2.3.5. If $M \in \mathrm{vec}^{\mathbf{Z}}$ is finitely generated and $\operatorname{barc} M = \{(b_i, d_i); i \in I\}$ for some index set I, then $0 \to \bigoplus_{i\in I, d_i \neq \infty} F(d_i) \to \bigoplus_{i \in I} F(b_i)$ is a free resolution of M.

Every persistence module has a (not necessarily finite) free resolution. Finite free resolutions (resp., free presentations) are a convenient way to describe a module. Namely, after having chosen bases for the modules in the resolution (presentation), the morphisms in the resolution (presentation) can be represented by graded matrices, which can be stored and processed by a computer program.

Remark (Lifts of morphisms). If $F_\bullet \to M$ and $G_\bullet \to N$ are free resolutions of two modules M and N, and $\phi\colon M \to N$ is a morphism, then there exists a (not necessarily unique) morphism $\phi_\bullet\colon F_\bullet \to G_\bullet$ such that the diagram

$$\begin{tikzcd} F_\bullet \arrow[r] \arrow[d, "\phi_\bullet"'] & M \arrow[d, "\phi"] \\ G_\bullet \arrow[r] & N \end{tikzcd}$$

commutes. This follows from exactness of the rows, and from the universal property of the free (or, more generally, projective) modules involved. The morphism $f_\bullet$ is called a *lift* of f.

Finite resolutions need not be unique. However, any two free resolutions of the same module are chain homotopy equivalent. Finitely generated $\mathbf{Z}^n$-persistence modules have an essentially unique smallest free resolution, called their *minimal free resolution*; see Theorem 2.3.18. The minimal free resolution of a module M is minimal in the sense that it is a direct summand of any other free resolution of M. We explain this in detail in Section 2.3.3.

Definition 2.3.6. The *ith graded Betti number* $\beta_i(M)$ of a finitely generated module $M \in \mathrm{vec}^{\mathbf{Z}^n}$ is the graded rank $\beta_i(M) = \operatorname{rk} F_i$ for any minimal free resolution $F_\bullet$ of M.

Theorem 2.3.18 below shows that a finitely generated module has an essentially unique free resolution, which implies that its graded Betti numbers are independent of the chosen minimal free resolution and hence well-defined.

Example 2.3.7. The barcode of a one-parameter persistence module $M \in \mathrm{vec}^{Z}$ represents a minimal free resolution. If $M \in \mathrm{vec}^{\mathbf{Z}}$ is finitely generated and $\operatorname{barc} M = \{(b_i, d_i) \mid i \in I\}$ for some finite indexing set I, then $\beta_0 = \{b_i \mid i \in I\}$ and $\beta_1(M) = \{d_i \mid i \in I, d_i < \infty\}$.

2.3.2 Examples

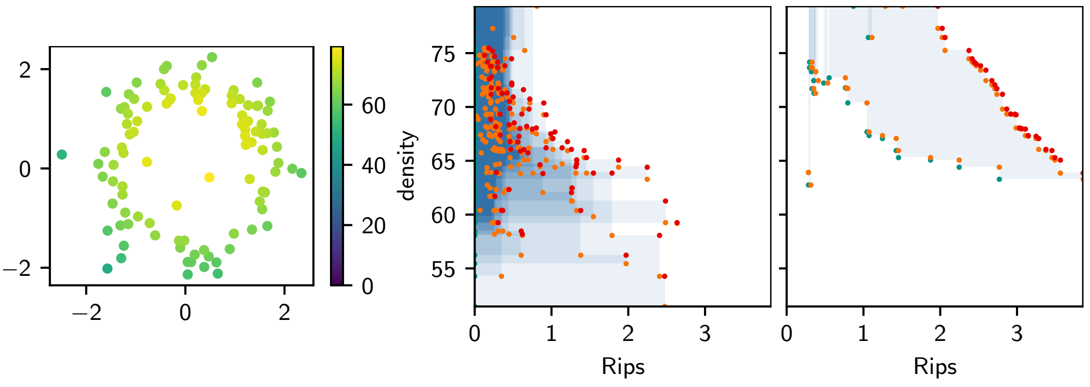
Density-Rips filtrations We equip the point cloud S from Figure 2.4b with the density function

$$\rho\colon S \to \mathbf{R}_{\geq 0}, \quad p \mapsto \sum_{p \neq q \in S} \exp\left(-\tfrac{\|p-q\|^2}{2\sigma^2}\right) \tag{2.13}$$

for a manually chosen bandwidth parameter σ, see Figure 2.6a for an example. We have computed a minimal free resolution of $H_\bullet(VR_*(S,\rho))$, using our implementation [94]. The corresponding Betti diagrams and Hilbert functions are shown in Figure 2.6. Note that the diagrams show reduced homology.

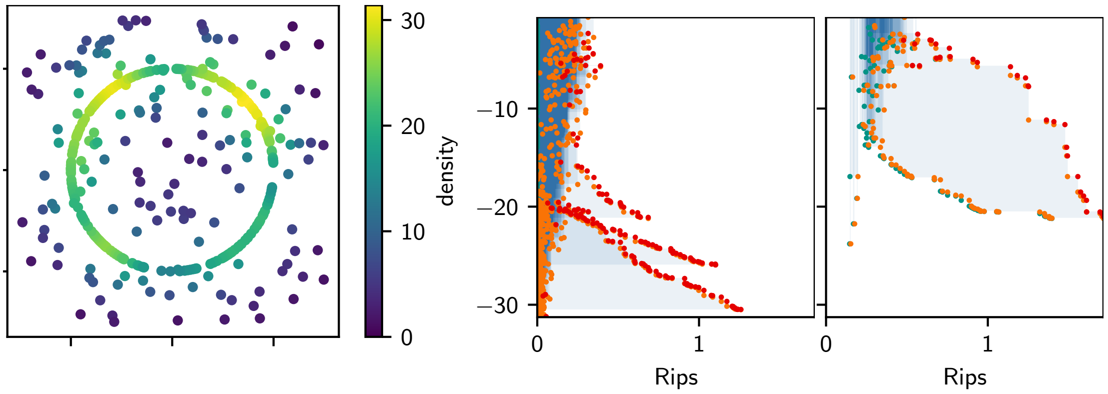
For a second example, let S be the point cloud shown Figure 2.7a. Despite the fact that the circle is clearly visible in the picture, the homology type of S^1 would be invisible in the corresponding Vietoris–Rips persistence diagram. We equip the point cloud with a density function ρ as in (2.13). Since ρ takes lower values on the supposed outliers, we filter the point cloud by $-\rho$. The graded Betti diagrams and Hilbert function of $H_\bullet(VR_*(S, -\rho))$ are shown in Figure 2.7.

From this diagram, one can obtain an estimate for a value of ρ that discerns the outliers. Each horizontal slice through the resolution at a value $-\rho_0$ corresponds to a resolution of the one-parameter persistent homology of $\{p \in S \mid \rho(p) \geq \rho_0\}$; that is, a barcode. One sees that for $\rho_0 \approx 14$, the corresponding horizontal slice contains a large region where $H_1(-)$ is one-dimensional. The corresponding barcode is shown in Figure 2.8.



(a) filtered point cloud **(b)** H_0 **(c)** H_1

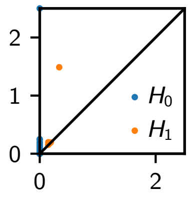
Figure 2.6: **(a)** The point cloud S from Figure 2.4b, equipped with the Gaussian density function $\rho\colon p \mapsto \sum_{p\neq q\in S}\exp(-\frac{\|p-q\|^2}{2\sigma^2})$ with $\sigma = 2.5$. The function ρ that takes larger values on the outliers, and lower values on the desired points. **(b)**, **(c)** Graded Betti numbers (teal: β_0, red: β_1, orange: β_2) and Hilbert function (shades of blue increasing from dimension $= 0$ to dimension ≥ 10) of the reduced homology $H_\bullet(VR_*(X,\rho))$ of the density-Rips complex of the point cloud from **(a)**.

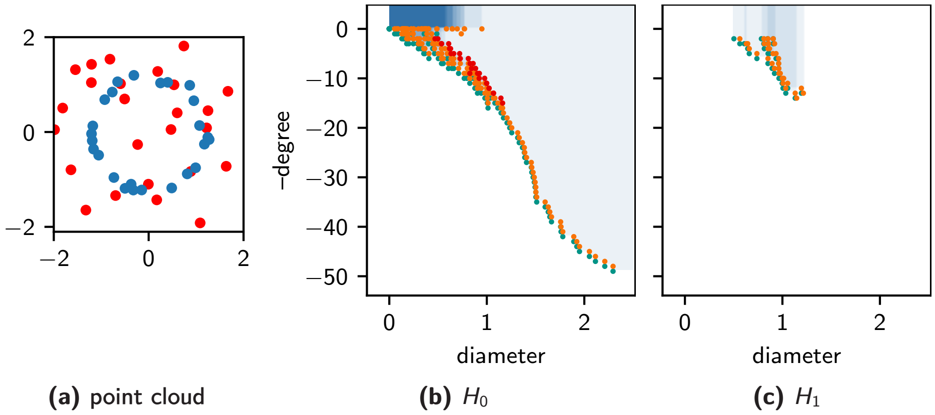


(a) filtered point cloud **(b)** H_0 **(c)** H_1

Figure 2.7: **(a)** Uniform sample (250 points) from a unit 1-sphere, superimposed with uniform sample (150 points) of the rectangle $[-1.5, 1.5]^2$. Each point is assigned the density from (2.13) with $\sigma = 0.2$. The right picture shows the (non-reduced) persistent homology of the associated one-parameter Vietoris–Rips complex. **(b)**, **(c)** Associated graded Betti numbers.

Figure 2.8: Barcode representing the horizontal slice through the diagrams in Figure 2.7 at density $\rho = 14$. This corresponds to taking the barcode of $H_\bullet(VR_*(S'))$, where S' consists of the points from Figure 2.7a that satisfy $\rho(-) \geq 14$.





(a) point cloud **(b)** H_0 **(c)** H_1

Figure 2.9: Graded Betti numbers and Hilbert function of (non-reduced) persistent homology of the degree-Rips filtration on the points from Figure 2.5a.

Degree-Rips filtrations Recall the point-cloud S from Figure 2.5a. Figure 2.9 shows the Hilbert function and graded Betti numbers of $H_\bullet(K_*)$ of the corresponding degree-Rips filtration.

2.3.3 Persistence modules as modules over a graded algebra

For explaining the theory of minimal free resolutions, it is convenient to talk about persistence modules using the language of graded algebras.

Definition 2.3.8. A $\mathbf{Z}^n$*-graded* k*-algebra* is a k-algebra A, such that the underlying k-vector space of A is a direct sum $A = \bigoplus_{z\in\mathbf{Z}^n} A_z$ of vector spaces, and multiplication satisfies $A_z A_{z'} \subseteq A_{z+z'}$ for all $z, z' \in \mathbf{Z}^n$. In particular, $1 \in A_0$.

A $\mathbf{Z}^n$*-graded* A*-module* is an A-module M, such that the underlying k-vector space of M is a direct sum $M = \bigoplus_{z\in\mathbf{Z}^n} M_j$ of vector spaces, and multiplication satisfies $A_z M_{z'} \subseteq M_{z+z'}$ for all $z, z' \in \mathbf{Z}^n$. A *morphism* of graded A-modules M and N is a morphism $f\colon M \to N$ of ordinary A-modules, with the additional property that $f(M_z) \subseteq N_z$ for all $z \in \mathbf{Z}^n$. We write A-$\mathrm{gMod}_{\mathbf{Z}^n}$ for the category of $\mathbf{Z}^n$-graded A-modules. The components A_z and M_z are called the *graded* or *homogeneous components* of A and M, respectively. An element of A or M is called *homogeneous* if it lies in a homogeneous component.

We require all graded algebras A we work with to be *non-negatively graded*, that is, $A_z = 0$ unless $z \geq 0$.

Example 2.3.9. The polynomial algebra $k[x_1, \ldots, x_n]$ is a (non-negatively) $\mathbf{Z}^n$-graded k-algebra, where for $z \in \mathbf{Z}^n$, the z-graded component is the one-dimensional vector space spanned by the monomial $x_1^{z_1} \cdots x_n^{z_n}$. It is also $\mathbf{Z}$-graded, where for $m \in \mathbf{Z}$, the vector space A_m is spanned by all monomials of total degree n. For convenience, we use the notation $x^z := x_1^{z_1} \cdots x_n^{z_n}$ for $z = (z_1, \ldots, z_n) \in \mathbf{Z}^n$.

Proposition 2.3.10 ([34, Theorem 1])**.** *The functor*

$$F\colon \mathrm{Vec}^{\mathbf{Z}^n} \to k[x_1, \ldots, x_n]\text{-}\mathrm{gMod}_{\mathbf{Z}^n}, \qquad F(M) = \textstyle\bigoplus_{z\in\mathbf{Z}^n} M_z, \tag{2.14}$$

is an equivalence of categories, where for $z, z' \in \mathbf{Z}^n$*, the element* x^z *acts on* $m \in F(M)_{z'}$ *by* $x^z m := M_{z+z',z'}(m)$*. Its quasi-inverse is the functor*

$$G\colon k[x_1, \ldots, x_n]\text{-}\mathrm{gMod}_{\mathbf{Z}^n} \to \mathrm{Vec}^{\mathbf{Z}^n}, \qquad G(M)_z = M_z, \tag{2.15}$$
$$G(M)_{z,z'} = x^{z-z'}|_{M_{z'}}.$$

Every equivalence of abelian categories is exact. In particular, limits and colimits in $\mathrm{Vec}^{\mathbf{Z}^n}$ correspond to limits and colimits in $k[x_1,\dots,x_n]\text{-gMod}_{\mathbf{Z}^n}$, and free (projective, injective) modules in $\mathrm{Vec}^{\mathbf{Z}^n}$ correspond to free (projective, injective) modules in $k[x_1,\dots,x_n]\text{-gMod}_{\mathbf{Z}^n}$.

Remark. One can define the functors F and G from Proposition 2.3.10 in greater generality. Namely, if S is a commutative partially ordered monoid and P a partially ordered set with a monotonic action $S \times P \to P$, then there is a fully faithful functor $F\colon \mathrm{Vec}^P \to k[S]\text{-gMod}_P$ defined analogously to (2.14), where $k[S]$ denotes the monoid algebra of S. See [50] for details.

Minimal free resolutions

Definition 2.3.11. A *(homological) d-ball* is a chain complex of the form

$$\cdots \to 0 \to F(z) \xrightarrow{\mathrm{id}} F(z) \to 0 \to \cdots$$

for some z, concentrated in degrees d, $d-1$. A chain complex of free modules is *trivial* if it is isomorphic to a direct sum of homological balls. It is *minimal* if it contains no homological ball as a direct summand.

Before we explore different equivalent ways of characterizing minimality of chain complexes, we establish the following lemma.

Lemma 2.3.12 (Eliminating balls). *Consider a chain complex*

$$\begin{tikzcd} C_\bullet\colon & \cdots \arrow[r] & K \arrow[r, "\left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right)"] & L \oplus M \arrow[r, "\left(\begin{smallmatrix} c & d \\ e & f \end{smallmatrix}\right)"] & N \oplus O \arrow[r, "\left(\begin{smallmatrix} g & h \end{smallmatrix}\right)"] & P \arrow[r] & \cdots . \end{tikzcd}$$

If c is invertible, then $C_\bullet \cong C'_\bullet \oplus B_\bullet$, where

$$\begin{tikzcd} C'_\bullet\colon & \cdots \arrow[r] & K \arrow[r, "b"] & M \arrow[r, "f-ec^{-1}d"] & O \arrow[r, "h"] & P \arrow[r] & \cdots \\ B_\bullet\colon & \cdots \arrow[r] & 0 \arrow[r] & L \arrow[r, "c"] & N \arrow[r] & 0 \arrow[r] & \cdots , \end{tikzcd}$$

and the projection $C_\bullet \to C'_\bullet$ is a quasi-isomorphism.

Remark 2.3.13. Every quasi-isomorphism of bounded below complexes of free modules is a chain homotopy equivalence. The same is true for bounded below complexes of projective or injective modules. This follows from [127, Corollary 1.5.4] and the lifting property of free (resp. projective, injective) modules.

Proof. Consider the automorphisms

$$\eta = \left(\begin{smallmatrix} 1 & c^{-1}d \\ 0 & 1 \end{smallmatrix}\right) \in \mathrm{Aut}(L \oplus M), \qquad \eta^{-1} = \left(\begin{smallmatrix} 1 & -c^{-1}d \\ 0 & 1 \end{smallmatrix}\right) \in \mathrm{Aut}(L \oplus M),$$
$$\theta = \left(\begin{smallmatrix} 1 & 0 \\ -ec^{-1} & 1 \end{smallmatrix}\right) \in \mathrm{Aut}(N \oplus O), \qquad \theta^{-1} = \left(\begin{smallmatrix} 1 & 0 \\ ec^{-1} & 1 \end{smallmatrix}\right) \in \mathrm{Aut}(N \oplus O).$$

Then $\theta\left(\begin{smallmatrix} c & d \\ e & f \end{smallmatrix}\right)\eta^{-1} = \left(\begin{smallmatrix} c & 0 \\ 0 & f-ec^{-1}d \end{smallmatrix}\right)$. Since $C_\bullet$ is a chain complex, $ca+db=0$ and $gc+he=0$. This implies that

$$\eta\left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right) = \left(\begin{smallmatrix} a+c^{-1}db \\ b \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right) \qquad (g,h)\theta^{-1} = (g+hec^{-1},h) = (0,h).$$

This shows that the diagram

$$\begin{tikzcd} C_\bullet\colon \arrow[d, shift right] & \cdots \arrow[r] & K \arrow[r, "\left(\begin{smallmatrix} a \\ b \end{smallmatrix}\right)"] \arrow[d, equal] & L \oplus M \arrow[r, "\left(\begin{smallmatrix} c & d \\ e & f \end{smallmatrix}\right)"] \arrow[d, shift right, "\eta"'] & N \oplus O \arrow[r, "(g\ h)"] \arrow[d, shift right, "\theta"'] & P \arrow[r] \arrow[d, equal] & \cdots \\ B_\bullet \oplus C'_\bullet\colon \arrow[u, shift right] & \cdots \arrow[r] & K \arrow[r, "\left(\begin{smallmatrix} 0 \\ b \end{smallmatrix}\right)"] & L \oplus M \arrow[r, "\left(\begin{smallmatrix} c & 0 \\ 0 & f-ec^{-1}d \end{smallmatrix}\right)"] \arrow[u, shift right, "\eta^{-1}"'] & N \oplus O \arrow[r, "(0{,}h)"] \arrow[u, shift right, "\theta^{-1}"'] & P \arrow[r] & \cdots . \end{tikzcd}$$

commutes and is a pair of mutually inverse isomorphisms of chain complexes. Since $B_\bullet = (\cdots 0 \to L \xrightarrow{c} N \to 0 \to \cdots)$ is trivial, the projection $C_\bullet \to C'_\bullet$ is a quasi-isomorphism. □

A graded k-algebra A is called *graded local* if it has a unique maximal homogeneous ideal $\mathfrak{m}$, and *connected* if $A_0 = k$. Every connected k-algebra A is graded local with $\mathfrak{m} = \bigoplus_{i>0} A_i$. As a $\mathbf{Z}$- or $\mathbf{Z}^n$-graded algebra, the polynomial algebra $k[x_1, \ldots, x_n]$ is connected and thus graded local, with the unique maximal homogeneous ideal $\mathfrak{m} = (x_1, \ldots, x_n)$.

We are now ready to collect some equivalent ways to characterize minimal chain complexes and minimal free resolutions. The idea behind these is that complex of free modules is non-minimal if and only if its boundary morphisms contain an invertible component. Since boundary morphisms are homogeneous and $\mathfrak{m}$ is the ideal of all homogeneous non-invertible elements $k[x_1, \ldots, x_n]$, we get that a complex is minimal if and only if all components of its boundary morphisms lie in $\mathfrak{m}$. The following makes this precise.

Lemma 2.3.14 (Characterizations of minimality)**.** *Let A be a connected graded algebra and $(F_\bullet, \partial_\bullet)$ be chain complex of graded free A-modules of finite rank. Then the following are equivalent:*

- (i) *$F_\bullet$ is minimal.*
- (ii) *For every d, we have $\operatorname{im} \partial_{d+1} \subseteq \mathfrak{m}F_d$.*
- (iii) *All boundary operators of the complex $F_\bullet/\mathfrak{m}F_\bullet$ are zero.*
- (iv) *All boundary operators of the complex $F_\bullet \otimes_{k[x_1,\ldots,x_n]} k$ are zero.*

In (iv), we regard k as an $k[x_1, \ldots, x_n]$-module with the only non-zero component k in grade 0, on which all indeterminates act by zero. The tensor product is the usual tensor product of graded modules.

Proof. Choose homogeneous bases $(e_j)_{j\in J}$ and $(e'_i)_{i\in I}$ of the free modules F_{d+1} and F_d, respectively, for indexing sets J and I. For every $i \in I$ and $j \in J$, let $f_{ij} \in k[x_1, \ldots, x_n]$ be such that $\partial_{d+1}(e_j) = \sum_{i\in I} f_{ij} e'_j$

$(i) \Rightarrow (ii)$ Assume that $\partial_{d+1}(a) \notin \mathfrak{m}F_d$ for some $a \in F_{d+1}$. Then there exist also $i_0 \in I$ and $j_0 \in J$ such that $f_{i_0,j_0} \in A \setminus \mathfrak{m} = k^*$. In particular, e_{i_0} and e'_{i_0} are homogeneous of the same degree p, and $f_{i_0,j_0} \colon F(p) \to F(q)$ is an isomorphism. Then Lemma 2.3.12 shows that the $d+1$-ball

$$\cdots \to 0 \to F(p) \xrightarrow{f_{i_0,j_0}} F(p) \to 0 \to \cdots$$

is a direct summand of $F_\bullet$, so $F_\bullet$ is not minimal.

$(ii) \Rightarrow (i)$ Assume that there are complexes $(B_\bullet, \partial^B_\bullet)$ and $(F'_\bullet, \partial^{F'}_\bullet)$ such that $F_\bullet \cong B_\bullet \oplus F'_\bullet$, where $B_\bullet$ is a homological $d+1$-ball. This means that $\operatorname{im} \partial^B_{d+1} = B_d$. Because B_d is finitely generated, $\operatorname{supp} B_d$ contains at least one minimal element g. Because all elements of $\mathfrak{m}$ have strictly positive grade, $(\mathfrak{m}B_d)_g = 0$. Therefore, $\operatorname{im} \partial^B_d \not\subseteq \mathfrak{m}B_{d-1}$.

$(ii) \Leftrightarrow (iv)$ If $B_\bullet = (\cdots \to 0 \to F(z) \xrightarrow{=} F(z) \to 0 \to \cdots)$ is a homological ball, then $B_\bullet \otimes k = (\cdots \to 0 \to k\langle -z\rangle \xrightarrow{=} k\langle -z\rangle \to 0 \to \cdots)$, where $k\langle -z\rangle$ is the persistence module with $(k\langle -z\rangle)_w = k$ if $w = z$, and zero otherwise. In particular, the boundary operator of $B_\bullet \otimes k$ is not zero. Thus, $F_\bullet$ contains no homological ball as a direct summand if and only if $F_\bullet \otimes k$ has no non-zero boundary operator.

$(iv) \Leftrightarrow (iii)$ This follows by recalling that $\mathfrak{m}$ is the ideal such that $k = k[x_1, \ldots, x_n]/\mathfrak{m}$. □

Definition 2.3.15. A free resolution is *minimal* if it is minimal as a chain complex. A free presentation of M is *minimal* if it extends to a minimal free resolution.

Example 2.3.16 (Barcodes). Let $M \in \mathrm{vec}^{\mathbf{Z}^2}$ be finitely generated with generating set $\{m_i \,|\, i \in I\}$ for some finite indexing set I, and let $\operatorname{barc} M = \{(b_i, d_i) \mid i \in I\}$. Then

$$\begin{array}{rcccl} 0 \to \bigoplus_{i \in I, d_i < \infty} F(d_i) & \longrightarrow & \bigoplus_{i \in I} F(b_i) & \xrightarrow{\varepsilon} & M, \\ e_i & \longmapsto & F(b_i)_{d_i, b_i}(e'_i) & & \\ & & e'_j & \longmapsto & g_j \end{array}$$

is a minimal free resolution of M, where e_i and e'_i denote the ith standard basis vector of the respective free module. See also Definition 2.2.6. A non-minimal free resolution of M would correspond to a barcode with $b_i = d_i$ for some $i \in I$. In this case, the corresponding generator m_i is not necessary.

Lemma 2.3.17. *Let A be connected and $(F_\bullet, \partial_\bullet)$ be a free resolution of an A-module M. Then $F_\bullet$ is minimal if and only if the augmented resolution $F_\bullet \xrightarrow{\partial_0} M \to 0$ (see Definition 2.3.4) has the property that for every $d \geq 0$, a basis of F_d is mapped to a minimal generating system of $\operatorname{im} \partial_d$.*

Proof. According to Lemma 2.3.14(*ii*), $F_\bullet$ is minimal if and only if $\operatorname{im} \partial_{d+1} \subseteq \mathfrak{m} F_d$ for all $d \geq 0$. Fix a $d \geq 0$, and let $(e_i)_{i \in I}$ be a homogeneous basis of F_d for some indexing set I.

Assume that $(\partial_d(e_i))_{i \in I}$ is a non-minimal generating set of $\operatorname{im} \partial_d$. Then there exists an $i_0 \in I$ and homogeneous polynomials $f_i \in k[x_1, \ldots, x_n]$ for all $i \neq i_0$ such that $\partial_d(e_{i_0}) = \sum_{i \neq i_0} f_i \partial(e_i)$. Then $g \coloneqq e_{i_0} - \sum_{i \neq i_0} f_i e_i$ satisfies $\partial_d(g) = 0$. By exactness, we get $g \in \operatorname{im} \partial_{d+1}$, but $g \notin \mathfrak{m} F_d$ because $e_{i_0} \in F_d \setminus \mathfrak{m} F_d$.

Conversely, assume $\operatorname{im} \partial_{d+1} \nsubseteq \mathfrak{m} F_d$. Then there exists a non-zero homogeneous element $g \in \operatorname{im} \partial_{d+1} \setminus \mathfrak{m} F_d$. For $i \in I$, let $f_i \in k[x_1, \ldots, x_n]$ be the homogeneous polynomials such that $g = \sum_{i \in I} f_i e_i$. Since $g \in F_d \setminus \mathfrak{m} F_d$, there exists an $i_0 \in I$ such that $f_{i_0} \in A \setminus \mathfrak{m} = k^*$. Since $\partial_g(g) = 0$, we get $\partial_d(e_{i_0}) = -f_{i_0}^{-1} \sum_{i \neq i_0} f_i \partial_d(e_i)$, so the generating system $(\partial_d(e_i))_{i \in I}$ of $\operatorname{im} \partial_d$ is not minimal. □

The algebra $k[x_1, \ldots, x_n]$ is Noetherian, which implies that every submodule of a finitely generated module is finitely generated again. Therefore, every finitely generated module has a minimal free resolution, obtained by choosing a minimal generating system $\partial_0 \colon F_0 \to M$ of M and a minimal generating system $\partial_{d+1} \colon F_{d+1} \to \ker \partial_d$ of $\ker \partial_d$ for every $d \geq 0$. The thus obtained resolution is essentially unique:

Theorem 2.3.18 (Uniqueness of minimal free resolutions [65, p. 491])**.** *Let $M \in k[x_1, \ldots, x_n]$-$\mathrm{gMod}_{\mathbf{Z}^n}$ be finitely generated and $F_\bullet$ be a minimal free resolution of M. For every other free resolution $F'_\bullet$ of M, there exists a trivial chain complex $F''_\bullet$ such that $F'_\bullet \cong F_\bullet \oplus F''_\bullet$. In particular, all minimal free resolutions of M are isomorphic.*

Theorem 2.3.19 (Hilbert's syzygy theorem [110, p. 56, 65, p. 474])**.** *Every minimal free resolution of a finitely generated module $M \in k[x_1, \ldots, x_n]$-$\mathrm{gMod}_{\mathbf{Z}^n}$ has length at most n.*

Proof (sketch). Let $R = k[x_1, \ldots, x_n]$. For $i = 1, \ldots, n$, consider the chain complexes $\Omega_i \colon 0 \to F(e_i) \xrightarrow{x_i} F(0) \to 0$ of $\mathbf{Z}^n$-graded free R-modules concentrated in degrees 1 and 0, where $e_i \in \mathbf{Z}^n$ is the ith unit vector. The *Koszul complex* $\Omega_\bullet \coloneqq \bigotimes_{i=1}^n \Omega_i$ is a $\mathbf{Z}^n$-graded minimal free resolution of length n of the graded R-module $k \cong F(0)/\mathfrak{m} F(0)$; here, $\otimes$ denotes the usual tensor product of chain complexes of modules. For a finitely generated module $M \in k[x_1, \ldots, x_n]$-$\mathrm{gMod}_{\mathbf{Z}^n}$ and a $\mathbf{Z}^n$-graded minimal free resolution $F_\bullet$ of M, we obtain that

$$F_i \otimes_R k \overset{2.3.14(iv)}{\cong} H_i(F_\bullet \otimes_R k) \overset{(*)}{=} \mathrm{Tor}_i^R(M, k) \overset{(*)}{=} H_i(M \otimes_R \Omega_\bullet) = 0$$

if $i > n$ because $\Omega_\bullet$ has length n and M (as a chain complex) has length 0, using the definition of Tor in $(*)$. Now each F_i is finitely generated, and $F_i \otimes_R k = 0$ for all $i > n$. This implies that $F_i = 0$ for all $i > n$, so $F_\bullet$ has length n. □

Corollary 2.3.20. *If $F_\bullet = (F_{n-1} \xrightarrow{\partial_{n-1}} F_{n-2} \to \cdots \xrightarrow{\partial_1} F_0)$ is an exact sequence of graded free $k[x_1, \dots, x_n]$-modules, then $\ker \partial_{n-1}$ is free.*

Proof. The sequence $0 \to \ker \partial_{n-1} \to F_{n-1} \to F_{n-2} \to \cdots \to F_0 \to \operatorname{coker} \partial_1 \to 0$ is exact. We show that it is a free resolution of $M := \operatorname{coker} \partial_1$. We extend $F_\bullet$ to a free resolution $F_\bullet \colon \cdots \to F_n \to F_{n-1} \to \cdots \to F_0$ of M. According to Theorem 2.3.18, $F_\bullet$ is isomorphic to a direct sum of a minimal free resolution $F'_\bullet$ of M and a trivial complex, so

$$F_\bullet \cong (\ldots \to \underbrace{B_3 \oplus F'_2 \oplus B_2}_{F_2} \to \underbrace{B_2 \oplus F'_1 \oplus B_1}_{F_1} \to \underbrace{B_1 \oplus F'_0}_{F_0} \to 0)$$

for free modules $B_1, B_2, \ldots$. Then

$$0 \to F'_n \oplus B_n \to \underbrace{F'_{n-1} \oplus B_n \oplus B_{n-1}}_{F_{n-1}} \to F_{n-2} \to \cdots \to F_0 \to M \to 0$$

is a free resolution of M. By the five lemma, $\ker \partial_{n-1} \cong F'_n \oplus B_n$, which is free. □

2.3.4 Computing minimal free resolutions

Let $C_\bullet$ be a chain complex of finite rank free $\mathbf{Z}^n$-persistence modules. We describe an algorithm that computes a minimal complex quasi-isomorphic to $C_\bullet$. Let $C_d = \bigoplus_{k=1}^{n_d} F(z_{dk})$ for every d, and let D_d be the graded matrix representing ∂_d.

Definition 2.3.21. Let D be a valid graded $m \times n$-matrix and $i \leq m$, and $j \leq n$. We call (i, j) a *local pair* of D if $[D]_{ij} \neq 0$ and $\mathrm{rg}_i^D = \mathrm{cg}_j^D$. In this case, we call i a local row index and j a local column index of D. We call a row or column *local* if it is part of a local pair. We call D *minimal* if it has no local pairs.

This definition is motivated by the following lemma:

Lemma 2.3.22. *The complex $C_\bullet$ is minimal if and only if D_d is minimal for every d.*

Proof. Let $D \colon F \to F'$ be a valid graded matrix representing a morphism of free modules F and F' with bases $(e_j)_{j \in J}$ and $(e'_i)_{i \in I}$ for index sets I and J, respectively. Then $D \subseteq \mathfrak{m}F'$ if and only if D has no local pairs. To see this, note that $De_j = \sum_i f_{ij} e'_i$ for all i, j, for polynomials $f_{ij} = x^{\mathrm{cg}_j^D - \mathrm{rg}_i^D} D_{ij} \in k[x_1, \ldots, x_n]$. If D has a local pair (i_0, j_0), then f_{ij} is invertible, so $\operatorname{im} D \nsubseteq \mathfrak{m}F'$. The converse is similar. The statement then follows from Lemma 2.3.14(*ii*). □

If D_d has a local pair (i_0, j_0), then we apply Lemma 2.3.12 with $c = [D_d]_{i_0, j_0}$. Define the graded matrices

$$\begin{aligned} D'_{d+1} &:= ([D_{d+1}]_{ij})_{i \neq i_0}, \\ D'_d &:= ([D_d]_{ij} - \tfrac{[D_d]_{i_0 j}[D_d]_{i j_0}}{[D_d]_{i_0 j_0}})_{i \neq i_0, j \neq j_0}, \\ D'_{d-1} &:= ([D_{d-1}]_{ij})_{j \neq j_0}, \end{aligned}$$

and $D'_d := D_d$ in all other cases. Lemma 2.3.12 shows that these represent a chain complex

$$\cdots \to C_{d+1} \xrightarrow{D'_{d+1}} \bigoplus_{k \neq j_0} F(z_{dk}) \xrightarrow{D'_d} \bigoplus_{k \neq i_0} F(z_{d-1,k}) \xrightarrow{D'_{d-1}} C_{d-2} \to \cdots \tag{2.16}$$

Algorithm 3: Minimization: removes all local pairs from a valid graded matrix M. [71, 86, 99].

Input: A valid graded $m \times n$-matrix M such that $\mathrm{rg}_i^M \ngeq \mathrm{rg}_{i'}^M$ and $\mathrm{cg}_j^M \ngeq \mathrm{cg}_{j'}^M$ for all $i < i'$ and $j < j'$.
Output: A graded matrix M', non-local row and column indices r, c of M.

```
function Minimize($M$):
    $p \leftarrow 0 \in \mathbf{N}^m$
    $c \leftarrow \emptyset$
    for $j = 1, \dots, n$ do                                          ▷ identify local pairs
        forever do
            $i \leftarrow \mathrm{piv}(M_j)$
            if $\mathrm{rg}_i^M \neq \mathrm{cg}_j^M$ then $c \leftarrow c \cup \{j\}$; break          ▷ column j non-local
            else if $p_i = 0$ then $p_i \leftarrow j$; break                ▷ column j local with pivot i
            else $[M]_j \leftarrow [M]_j - \frac{[M]_{ij}}{[M]_{ip_j}}[M]_{p_i}$
    for $j \in c$ do                                                  ▷ embarrassingly parallel
        while $\{i \mid D_{ij}$ and $p_i \neq 0\} \neq \emptyset$ do                 ▷ remove entries in local rows
            $i \leftarrow \max\{i \mid D_{ij} \neq 0$ and $p_i \neq 0\}$
            $[M]_j \leftarrow [M]_j - \frac{[M]_{ij}}{[M]_{ip_j}}[M]_{p_i}$
    $r \leftarrow \{i \mid p_i = 0\}$
    return $(M_{ij})_{i \in r, j \in c}, r, c$
```

quasi-isomorphic to $C_\bullet$. Repeating this procedure for every matrix D_d until there are no more local pairs in the chain complex yields a minimal complex $C'_\bullet$ quasi-isomorphic to $C_\bullet$.

For the following, assume that the columns of D_d are ordered non-decreasingly by grade; that is, D_d satisfies $\mathrm{rg}_i^{D_d} < \mathrm{rg}_{i'}^{D_d}$ implies $i < i'$ and $\mathrm{cg}_j^{D_d} < \mathrm{cg}_{j'}^{D_d}$ for all $j < j'$.

Proposition 2.3.23 (Elimination of homological d-balls [71, 86, 99])**.** *Let $C_\bullet$ be a chain complex of finite rank free modules, represented by graded matrices $D_\bullet$ that have the row and column grades in non-decreasing order. Then Algorithm 3 computes $(D'_d, r, c) \coloneqq \texttt{Minimize}(D_d)$, where D'_d is a minimal graded matrix and r and c are subsets of $\mathbf{N}$, such that the graded matrices*

$$\dots, D_{d+2}, \quad D_{d+3}, \quad ([D_{d+1}]_{ij})_{i \in c}, \quad D'_d, \quad ([D_{d-1}]_{ij})_{j \in r}, \quad D_{d-2}, \quad D_{d+3}, \dots \tag{2.17}$$

represent a chain complex $C'_\bullet$ quasi-isomorphic to $C_\bullet$.

Remark. Besides the minimal matrix D', the algorithm returns the set r of all row indices i and the set c of all column indices j that do not belong to a local pair $(i, -)$ or $(-, j)$. These sets are necessary to truncate the matrices adjacent to D in a chain complex as in (2.17).

Proof of Proposition 2.3.23 (idea). Since the algorithm only acts on D_d, we only write D for D_d as in the algorithm. That the rows columns of D are non-decreasingly ordered by grade ensures that a column $[D]_j$ is local if and only if $(\mathrm{piv}[D]_j, j)$ is a local pair. Namely, by validity, no non-zero entry in $[D]_j$ can have a row grade not smaller or equal than cg_j^D, and by the assumption, no non-zero entry in $[D]_j$ can have a row grade larger than the pivot grade.

The algorithm proceeds in two phases. In the first phase, the algorithm identifies a maximal set of pairwise disjoint local pairs, which corresponds to a maximal trivial direct summand of D. To do so, it reduces each column $[D]_j$ using column additions from columns $[D]_{j_0}$ that are already identified as local. Let $i = \mathrm{piv}[D]_j$. The reduction of $[D]_j$ stops either if $\mathrm{rg}_i^D \neq \mathrm{cg}_j^D$ (in this case, $[D]_j$ is non-local), or $\mathrm{rg}_i^D = \mathrm{cg}_j^D$ but no other local column has the same pivot (in this case, (i, j) is identified as a local pair). In the second phase, the algorithm performs column additions from the local to the non-local columns in order to remove all entries in the local rows.

To verify correctness, we note that the algorithm repeatedly performs the construction from (2.16). Lemma 2.3.12 implies that the computed chain complex $C'_\bullet$ is quasi-isomorphic and hence (by Remark 2.3.13) homotopy equivalent to $C_\bullet$. One verifies that $D' = \texttt{Minimize}(D)$ has no local pairs and thus is minimal. □

Corollary 2.3.24. (*i*) *Let $D_1, D_2, \dots, D_n$ be graded matrices that represent a chain complex $C_\bullet$ of free n-parameter persistence modules. Then Algorithm 4 computes graded matrices $D'_1, D'_2, \dots$ that represent a minimal chain complex homotopy equivalent to $C_\bullet$.*

(*ii*) *Let $D^1, D^2, \dots, D^n$ be graded matrices that represent a cochain complex $C^\bullet$ of free n-parameter persistence modules. Then Algorithm 5 computes graded matrices $D'^1, D'^2, \dots$ that represent a minimal cochain complex homotopy equivalent to $C^\bullet$.*

Proof. Algorithm 4 applies Algorithm 3 to repeatedly. According to Proposition 2.3.23, this produces a sequence of homotopy equivalent chain complexes

$$\begin{tikzcd}
\cdots \arrow[r] & C_3 \arrow[r, "D_3"] \arrow[d] & C_2 \arrow[r, "D_2"] \arrow[d] & C_1 \arrow[r, "D_1"{name=A1}] \arrow[d] & C_0 \arrow[d] \\
\cdots \arrow[r] & C_3 \arrow[r, "D_3"] \arrow[d] & C_2 \arrow[r, "D'_2"{name=A2}] \arrow[d] & C'_1 \arrow[r, "D''_1"{name=B1}] \arrow[d] & \tilde{C}_0 \arrow[d] \\
\cdots \arrow[r] & C_3 \arrow[r, "D'_3"{name=A3}] \arrow[d] & C'_2 \arrow[r, "D''_2"{name=B2}] \arrow[d] & \tilde{C}_1 \arrow[r, "\tilde{D}_1"] \arrow[d] & \tilde{C}_0 \arrow[d] \\
\cdots \arrow[r] & C_3 \arrow[r, "D''_3"{name=B3}] \arrow[d, dotted] & \tilde{C}_2 \arrow[r, "\tilde{D}_2"] \arrow[d, dotted] & \tilde{C}_1 \arrow[r, "\tilde{D}_1"] \arrow[d, dotted] & \tilde{C}_0 \arrow[d, dotted] \\
\cdots \arrow[r] & \tilde{C} \arrow[r, "\tilde{D}_3"] & \tilde{C}_2 \arrow[r, "\tilde{D}_2"] & \tilde{C}_1 \arrow[r, "\tilde{D}_1"] & \tilde{C}_0
\arrow[Rightarrow, from=A1, to=B1, "\texttt{Minimize()}"]
\arrow[Rightarrow, from=A2, to=B2, "\texttt{Minimize()}"]
\arrow[Rightarrow, from=A3, to=B3, "\texttt{Minimize()}"]
\end{tikzcd}$$

for free modules $C'_d = F(\mathrm{cg}^{D''_d})$ and $\tilde{C}_d = F(\mathrm{cg}^{\tilde{D}_d})$, where

$$\begin{aligned}
D'_d &:= \begin{cases} D_d & \text{if } d = 1, \\ ([D_d]_{ij})_{d \in r_{d-1}} & \text{if } d > 1 \end{cases}, \\
(D''_d, r_d, c_d) &:= \texttt{Minimize}(D'_d), \\
\tilde{D}_d &:= ([D'_d]_{ij})_{j \in r_{d+1}}
\end{aligned}$$

It follows from Proposition 2.3.23 that each of the vertical maps of chain complexes is a homotopy equivalence indeed, and that each computed matrix D''_d is minimal, in the sense of Definition 2.3.21. The matrices $\tilde{D}_d$ are obtained from D''_d by deleting certain columns. This cannot introduce new local pairs, so also all of the matrices $\tilde{D}_d$ are minimal. By Lemma 2.3.22, the matrices $\tilde{D}_d$ represent a minimal chain complex $\tilde{C}_\bullet$. For cochain complexes, the proof is analogous. □

Remark. A chain complex $C_\bullet$ of free modules is minimal if and only if its dual cochain complex $C^\bullet$ is. If $C_\bullet$ is a cochain complex, we two have two alternatives to produce a minimal chain complex homotopy equivalent to $C_\bullet$; namely, by applying Algorithm 4 to the boundary matrices D_d of $C_\bullet$ (which performs column operations on the matrices D_d), or by applying Algorithm 5 to the coboundary matrices $(D_d)^\top$ of $C^\bullet$ (which can be seen as performing row operations on the matrices D_d).

Algorithm 4: Minimization of chain complexes.

Input: Graded matrices $D_1, D_2, \ldots$ forming a chain complex $C_\bullet$.
Output: Graded matrices $D'_1, D'_2, \ldots$ forming a minimal chain complex $C'_\bullet$ quasi-isomorphic to $C_\bullet$.

```
$D'_1, r, c \leftarrow$ Minimize($D_1$)
for $d = 2, \ldots$ do
    $D'_d \leftarrow ([D_d]_{ij})_{i \in r}$
    $D'_d, r, c \leftarrow$ Minimize($D'_d$)
    $D'_{d-1} \leftarrow ([D_{d-1}]_{ij})_{j \in r}$
    yield $D'_{d-1}$
```

Algorithm 5: Minimization of cochain complexes.

Input: Graded matrices $D^1, D^2, \ldots$ forming a cochain complex $C^\bullet$
Output: Graded matrices $D'^1, D'^2, \ldots$ forming a minimal cochain complex $C'^\bullet$ quasi-isomorphic to $C^\bullet$.

```
$D'^1, r, c \leftarrow$ Minimize($D^1$)
for $d = 2, \ldots$ do
    $D'^d \leftarrow ([D^d]_{ij})_{j \in c}$
    $D'^d, r, c \leftarrow$ Minimize($D'^d$)
    $D'^{d-1} \leftarrow ([D_{d-1}]_{ij})_{i \in c}$
    yield $D'^{d-1}$
```

2.4 Computing two-parameter persistent homology

In this section, we explain how to compute graded matrices representing a minimal free resolution of $H_d(K_*)$ for each d, where $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}^n}$ is a finite two-parameter filtered simplicial complex. If K_* is one-critically filtered, $C_\bullet(K_*)$ is a chain complex of free modules in $\mathrm{vec}^{\mathbf{Z}^n}$. Now, the task is to compute a minimal free resolution of $H_d(C_\bullet)$ for each d, where $C_\bullet$ is a chain complex of free $\mathbf{Z}^n$-persistence modules.

This is commonly solved by Gröbner basis methods; for example, the algorithms [21, 67, 93] can be used for this task. Gröbner base methods have been previously studied in the context of persistent homology [33, 119]. A specific algorithm to compute a minimal free presentation or resolution of $H_d(C_\bullet)$ for the case $n = 2$ has been proposed in [99]. It relies on the fact that in this case, according to Corollary 2.3.20, $Z_d(C_\bullet)$ is free for every d. The algorithm [99], which is part of the software package `Rivet` [123], has been reported to outperform the implementation of the Gröbner base algorithm [67, 93] in Singular, Macaulay 2 and CoCoA [99]. The algorithm has been further improved in [73, 86] and implemented in the software `mpfree` [85]. We refer to this algorithm as the *Lesnick–Wright-* or *LW-algorithm.*

In the rest of this chapter, we explain how to compute a minimal free resolution of $H_d(C_\bullet)$, following [73, 86, 99].

Remark 2.4.1 (Multi-critical filtrations). If K_* is m-critically $\mathbf{Z}^2$-filtered for $m > 1$, then $C_\bullet(K_*)$ need not be a chain complex of free modules. However, one can define a chain complex $C'_\bullet$ of free modules that is quasi-isomorphic to $C_\bullet(K_*)$ [36], where each d-simplex of K_* contributes m generators of C'_d and $m - 1$ generators of C'_{d+1}.

An analogous complex can be constructed if K_* is m-critically $\mathbf{Z}^n$-filtered. Namely, assume the d-simplex σ enters the filtration at $g_{\sigma,1}, \ldots, g_{\sigma,m} \in \mathbf{Z}^n$. Define the *Koszul* complex

$$\Omega_\bullet(\sigma) \coloneqq \bigotimes_{i=1}^{m} (0 \to F(g_{\sigma,i}) \to F(0) \to 0),$$

of σ, where the two free modules inside the parentheses are placed in degrees zero and one. Here, the tensor product denotes the usual tensor product of chain complexes of graded modules.

For a chain complex $C_\bullet$, let $\operatorname{tr}_{\geq i} C_\bullet$ be the truncated chain complex with $(\operatorname{tr}_{\geq i} C_\bullet)_d := C_d$ if $d \geq i$, zero otherwise, and the boundary morphisms from $C_\bullet$. Then

$$\operatorname{tr}_{\geq 1}(\Omega_\bullet(\sigma))$$

is a chain complex of free modules quasi-isomorphic to the submodule of $C_d(K_*)$ generated by σ. Then one may define a double complex $\tilde{C}$ of persistence modules, whose rows are

$$\tilde{C}_p \bigoplus_{\sigma \in K_*^p} \operatorname{tr}_{\geq 1}(\Omega_\bullet(\sigma))$$

and whose vertical boundary maps come from the ones of $C_\bullet(K_*)$. Taking its total complex yields a free chain complex quasi-isomorphic to $C_\bullet(K_*)$.

2.4.1 The Lesnick–Wright algorithm

Let $C_\bullet$ be a chain complex of finite rank free $\mathbf{Z}^2$-persistence modules with boundary morphisms $\partial_d \colon C_d \to C_{d-1}$ represented by valid graded matrices D_d. For every d, let $i_d \colon Z_d(C_\bullet) \hookrightarrow C_d$ be the canonical inclusion. Because $C_\bullet$ is a chain complex, there exists a unique morphism $p_d \colon C_d \to Z_{d-1}(C_\bullet)$ such that $\partial_d = i_{d-1} p_d$. Corollary 2.3.20 implies that the kernel of a morphism of free modules of finite rank is free. In particular, the module $Z_d(C_\bullet) = \ker \partial_d$ is free for every d, so

$$0 \to Z_{d+1}(C_\bullet) \xrightarrow{i_{d+1}} C_{d+1} \xrightarrow{p_{d+1}} Z_d(C_\bullet) \qquad (2.18)$$

is a free resolution of $H_d(C_\bullet)$. We compute matrices I_{d+1} and P_{d+1} that represent the morphisms i_{d+1} and p_{d+1}.

Definition 2.4.2. The *lexicographic order* $\preceq_{\text{lex}}$ and the *colexicographic order* $\preceq_{\text{colex}}$ on $\mathbf{Z}^2$ are the total orders given by

$$\begin{aligned} z \preceq_{\text{lex}} z' \ &:\Leftrightarrow x \leq x' \text{ or } (x = x' \text{ and } y \leq y'), \\ z \preceq_{\text{colex}} z' \ &:\Leftrightarrow y \leq y' \text{ or } (y = y' \text{ and } x \leq x'). \end{aligned}$$

for $z = (x, y)$ and $z' = (x', y')$.

We have $z \leq z'$ if and only if $z \preceq_{\text{lex}} z'$ and $z \preceq_{\text{colex}} z'$.

Theorem 2.4.3 ([86, 99])**.** *Let $M \colon F \to G$ be a valid $\mathbf{Z}^2$ graded matrix, such that the column grades of M are colexicographically ordered. Then Algorithm 6 computes a graded matrix K that represents a basis of the free submodule* $\ker M$ *of F.*

Algorithm 6 computes the columns of K ordered lexicographically by grade. The role of the lexicographic and the colexicographic ordering in the algorithm can be exchanged.

Remark. Algorithm 6 does not consider the basis grades of G. This makes sense, because $\ker M = \ker JM$ for any free module H and every valid injective graded matrix $J \colon G \to H$.

We may compute the matrices I_{d+1} and P_{d+1} as follows. Consider the commutative diagram

$$\begin{tikzcd}
0 \arrow[r] & Z_{d+1}(C_\bullet) \arrow[r, "i_{d+1}", "\textcircled{1}"'] & C_{d+1} \arrow[r, "p_{d+1}", "\textcircled{2}"'] \arrow[rd, bend right, "\partial_{d+1}"'] & Z_d(C_\bullet) \arrow[r] \arrow[d, hook, "i_d \textcircled{1}"] & H_d(C_\bullet) \arrow[r] & 0 \\
 & & & C_d. & &
\end{tikzcd} \qquad (2.19)$$

Applying Algorithm 6 to D_d and D_{d+1} yields graded matrices $I_d = \texttt{Ker}(D_d)$ and $I_{d+1} = \texttt{Ker}(D_{d+1})$ representing i_d and i_{d+1}, respectively; see ①. We have $D_d D_{d+1} = 0$. Because I_{d+1} represents the inclusion $\ker D_d \to C_d$, there is a unique valid graded matrix $P_{d+1} \colon C_{d+1} \to C_d$ such that $D_{d+1} = I_d P_{d+1}$. This linear system can be solved for P_{d+1}, for example, using Algorithm 7; see ②. Then the matrices P_{d+1} and I_{d+1} represent the free resolution (2.18). Applying Algorithm 4 gives a minimal free resolution of $H_d(C_\bullet)$.

Remark. Although Algorithm 7 does not take the grading into account, it follows from existence and uniqueness of P_{d+1} that the result is a valid graded matrix.

Computing a minimal generating system of im ∂_{d+1} first It turns out that solving the linear system $D_{d+1} = I_d P_{d+1}$ for P_{d+1} does some unnecessary computation, as we will now see. Namely, if D_{d+1} is non-minimal (in the sense of Definition 2.3.21), then local pairs of D_{d+1} will give local pairs of P_{d+1}. Therefore, in this case, solving $D_{d+1} = I_d P_{d+1}$ computes columns of P_{d+1} that are discarded again when minimizing the resolution (Algorithm 4). [99] mentions the following construction that avoids this.

Consider again the commutative diagram in (2.19). We observe that $0 \to Z_{d+1}(C_\bullet) \to C_{d+1}$ is a free resolution of $\operatorname{im}\partial_{d+1}$. Using Algorithm 3, one may compute a minimal free resolution

$$0 \to Z'_{d+1} \xrightarrow{i'_{d_1}} C'_{d+1} \to \operatorname{im}\partial_{d+1} \tag{2.20}$$

of $\operatorname{im}\partial_{d+1}$ and let $\partial'_{d+1}\colon C'_{d+1} \to C_d$ such that $\operatorname{im}\partial'_{d+1} = \operatorname{im}\partial_{d+1}$; see ②′ in (2.21) below. Actually, computing i_{d+1} and passing to the minimal free resolution (2.20) of $\operatorname{im}\partial_{d+1}$ can be done in one step by a variant of Algorithm 6:

Proposition 2.4.4 ([86, 99])**.** *Let $M\colon F \to G$ be a valid graded matrix representing a morphism of free $\mathbf{Z}^2$-persistence modules. Then Algorithm 8 computes graded matrices M', K' such that M' represents a minimal generating system $M'\colon F' \to G$ of* $\operatorname{im} M$*, and K' represents a basis of* $\ker M'$.

Because $\operatorname{im}\partial'_{d+1} = \operatorname{im}\partial_{d+1}$, we have $\partial_d\partial'_{d+1} = 0$, so there exists a unique morphism p'_{d+1} such that $\partial'_{d+1} = i_d p'_{d+1}$; see ③′ in the following commutative diagram:

$$\begin{tikzcd}
 & 0 \arrow[r] & Z_{d+1}(C_\bullet) \arrow[rr, "i_{d+1}"] \arrow[dl] & & C_{d+1} \arrow[rr, "p_{d+1}"] \arrow[dl] \arrow[ddrr, bend right, "\partial_{d+1}"'] & & Z_d(C_\bullet) \arrow[rr] \arrow[dl, equal] \arrow[dd, hook, "i_d\ \text{①}"] & & H_d(C_\bullet) \arrow[r] \arrow[dl, equal] & 0 \\
0 \arrow[r] & Z'_{d+1} \arrow[rr, "\text{②′}\ i'_{d+1}"'] & & C'_{d+1} \arrow[rr, "\text{③′}\ p'_{d+1}"'] \arrow[ddrr, bend right, "\text{②′}\ \partial'_{d+1}"'] & & Z_d(C_\bullet) \arrow[rr] \arrow[dd, hook, "\text{①}\ i_d"'] & & H_d(C_\bullet) \arrow[r] & 0 & \\
 & & & & & & C_d \arrow[dl, equal] & & & \\
 & & & & & C_d. & & & &
\end{tikzcd} \tag{2.21}$$

Now, the sequence

$$0 \to Z'_{d+1} \xrightarrow{i'_{d+1}} C'_{d+1} \xrightarrow{p'_{d+1}} Z_d(C_\bullet) \tag{2.22}$$

is a free resolution of $H_d(C_\bullet)$.

Because (2.20) is a minimal resolution, (2.22) contains no homological 2-balls. Analogously to [98], we call such a resolution a *semi-minimal free resolution* of $H_d(C_\bullet)$. To obtain a minimal free resolution of $H_d(C_\bullet)$, it remains to eliminate all homological 1-balls by applying Algorithm 3 to P'_{d+1}. Together, we obtain the following strategy:

The following steps compute a free resolution of $H_d(C_\bullet)$:

1. *Compute a graded matrix $I_d :=$ `Ker`(D_d) (Algorithm 6) representing the kernel inclusion $i_d\colon Z_d(C_\bullet) \hookrightarrow C_d(C_\bullet)$; see ①.*
2. *Compute graded matrices $(D'_{d+1}, I'_{d+1}) :=$ `MGSWithKer`(D_{d+1}) (Algorithm 8) such that D'_{d+1} represents a minimal generating system of $B_d(C_\bullet)$, and I'_{d+1} represents* $\ker D'_{d+1}$*; see ②′.*
3. *Compute the unique graded matrix $P'_{d+1} :=$ `Factorize`(D'_{d+1}, I_d) (Algorithm 7) such that $I_d P'_{d+1} = D'_{d+1}$ that represents p'_{d+1}; see ③′.*

Algorithm 6: LW-algorithm. For a valid matrix $M\colon F \to G$ representing a morphism of free $\mathbf{Z}^2$-persistence modules, the algorithm computes a basis K of $\ker M \subseteq F$ [86, 99].

Input: A valid graded $m \times n$-matrix M with cg^M ordered colexicographically.
Output: A reduced graded matrix K representing a basis of $\ker M$ with cg^K ordered lexicographically.

```
function Ker($M$):
    $Q \leftarrow \{(\mathrm{cg}^M_j, j)\}$ as priority queue with $(z,j) \leq (z',j') :\Leftrightarrow z \preceq_{\mathrm{lex}} z' \vee (z = z' \wedge j \leq j')$
    $V \leftarrow E \in k^{n\times n}$                                      ▷ reduction matrix
    $K \leftarrow 0 \in k^{n\times 0}$ with $\mathrm{rg}^K = \mathrm{cg}^M$
    $p \leftarrow 0 \in \mathbf{N}^m$                                          ▷ if $p_i \neq 0$, then $i = \mathrm{piv}\, M_{p_i}$
    while $Q \neq \emptyset$ do
        $(z, j) \leftarrow$ PopMin($Q$)
        forever do
            $i \leftarrow \mathrm{piv}\, M_j$
            if $i = 0$ then append $V_j$ to $K$ and append $z$ to $\mathrm{cg}^K$; break
            else if $p_i = 0$ then $p_i \leftarrow j$; break
            else if $\mathrm{cg}^M_{p_i} \not\leq z$ then append $(\mathrm{cg}^M_{p_i} \vee z, j)$ to $Q$ and $p_i \leftarrow j$; break
            else
                $V_j \leftarrow V_j - M_{ij}/M_{ip_i} V_{p_i}$
                $M_j \leftarrow M_j - M_{ij}/M_{ip_i} M_{p_i}$
    return $K$
```

Algorithm 7: Factorization of matrices

Input: A matrix $L \in k^{\ell\times n}$, and a reduced matrix $M \in k^{\ell\times m}$
Output: The matrix $N \in k^{m\times n}$ such that $M = LN$, if it exists.

```
function Factorize($L$, $M$):
    $p \leftarrow 0 \in \mathbf{N}^\ell$
    $N \leftarrow 0 \in k^{m\times n}$
    for $j = 1, \ldots, m$ do $p_{\mathrm{piv}\, M_j} \leftarrow j$
    for $j = 1, \ldots, n$ do                                 ▷ embarrassingly parallel
        while $L_j \neq 0$ do
            $i \leftarrow \mathrm{piv}\, L_j$                                  ▷ if $p_i = 0$, then $\nexists N\colon L = MN$
            $N_j \leftarrow N_j - L_{ij}/M_{ip_i} e_{p_i}$
            $L_j \leftarrow L_j - L_{ij}/M_{ip_i} M_{p_i}$
    return $N$
```

Then I'_{d+1}, P'_{d+1} represent the free resolution of $H_{d+1}(C_\bullet)$ from (2.22)*. It contains no homological 2-balls as direct summands.*

4. *To obtain a minimal free resolution, split off all homological* 1*-balls using* `Minimize()` *(Algorithm 3).*

The procedures `Ker()` and `MGSWithKer()` (Algorithms 6 and 8) can be combined into a single algorithm `KerAndMgsWithKer()`. Given a valid graded matrix M, it computes a matrix K representing $\ker M$, a matrix M' representing a minimal generating system of $\mathrm{im}\, M$, and a matrix K' representing $\ker M'$. This can be used to compute minimal free resolutions for $H_0(C_\bullet), H_1(C_\bullet), \ldots$; see Algorithm 9.

Algorithm 8: A variant of Algorithm 6. For a morphism $f\colon F \to G$ of free modules, computes minimal generating system $f'\colon F' \to G$ of $\operatorname{im} f$, together with $\ker f'$.

Input: A graded $m \times n$-matrix M representing f.
Output: Graded matrices M' representing f' and K' representing a basis of $\ker f'$.

```
function MGSWithKer($M$):
    $Q \leftarrow \{(\mathrm{cg}^M_j, j)\}$ as priority queue with $(z,j) \leq (z',j')$ if $z \preceq_{\mathrm{lex}} z'$ or $z = z'$ and $j \leq j'$
    $M' \leftarrow$ graded $m \times 0$-matrix with $\mathrm{rg}^{M'} = \mathrm{rg}^M$
    $V' \leftarrow$ empty matrix
    $K' \leftarrow$ graded $0 \times 0$-matrix with $\mathrm{rg}^{K'} = \mathrm{cg}^{M'}$
    $m \leftarrow 0 \in \mathbf{N}^n$                                          ▷ index of $M_j$ in $M'$
    $p \leftarrow 0 \in \mathbf{N}^m$                                          ▷ pivot row to column
    while $Q \neq \emptyset$ do                                                  assignment
        $(z,j) \leftarrow$ PopMin($Q$)
        forever do
            $i \leftarrow \operatorname{piv} M_j$
            if $i = 0$ then
                if $m_j \neq 0$ then append $V'_{m_j}$ to $K'$ and $z$ to $\mathrm{cg}^{K'}$     ▷ $K_{m_j}$ represents a basis
                break                                                            element of $\ker M'$
            else if $p_i = 0$ then $p_i \leftarrow j$; break
            else if $\mathrm{cg}^M_{p_i} \not\leq z$ then
                add $(\mathrm{cg}^M_{p_i} \vee z, j)$ to $Q$
                $p_i \leftarrow j$
                break
            else
                if $z \neq \mathrm{cg}^M_j$ then $V'_{m_j} \leftarrow V'_{m_j} - M_{ij}/M_{ip_i} V'_{m_{p_i}}$
                $M_j \leftarrow M_j - M_{ij}/M_{ip_i} M_{p_i}$
        if $M_j \neq 0$ and $z = \mathrm{cg}^M_j$ then                              ▷ $M_j$ cannot be reduced to
            $m_j \leftarrow \#$ columns in $M' + 1$                                   zero at $\mathrm{cg}^M_j$ and therefore is
            append $M_j$ to $M'_{m_j}$                                             an element of the min. gen.
            $V' \leftarrow \left(\begin{smallmatrix} V' & 0 \\ 0 & 1 \end{smallmatrix}\right)$                                              system.
            $K' \leftarrow \left(\begin{smallmatrix} K' \\ 0 \end{smallmatrix}\right)$
            append $z$ to $\mathrm{cg}^{M'}$ and $\mathrm{rg}^{K'}_{m_j}$
    return $M'$, $K'$
```

Algorithm 9: Homology algorithm. Given valid graded matrices $D_1, \ldots, D_{\ell+1}$ representing a chain complex $C_\bullet$ of free $\mathbf{Z}^2$-modules, computes matrices $F_{d,1}, F_{d,0}$ representing a minimal free resolution of $H_d(C_\bullet)$ for each $d = 0, \ldots, \ell$.

```
$I_1, D'_1, I'_1 \leftarrow$ KerAndMgsWithKer($D_1$)
$F_{0,1}, F_{0,0} \leftarrow$ MinimizeCpx($I'_1, D'_1$)                                  ▷ see Algorithm 4
for $d = 1, \ldots, \ell$ do
    $I_{d+1}, D'_{d+1}, I'_{d+1} \leftarrow$ KerAndMgsWithKer($D_{d+1}$)
    $P_{d+1}$                   $\leftarrow$ Factorize($D'_{d+1}, I_d$)
    $F_{d,1}, F_{d,0}$             $\leftarrow$ MinimizeCpx($I'_{d+1}, P_{d+1}$)
```

2.4.2 Notes on implementations

Currently, the fastest implementation of the computation of minimal free presentations or resolutions of two-parameter persistent homology is `mpfree` [85]. This software applies chunk preprocessing (see below) to its input, before computing a minimal free presentation or resolution using the above approach [73, 86].

Chunk preprocessing The procedure `MinimizeCpx`() (Algorithm 4) can be used as an efficient preprocessing step to the actual computation of a resolution of persistent homology. This preprocessing is called *chunk preprocessing*. Its benefit lies in the fact that many complexes, such as Vietoris–Rips complexes or function-Rips complexes, contain many local pairs. The procedure `Minimize`() (Algorithm 3), which does the heavy lifting inside `MinimizeCpx`(), can be efficiently parallelized (see below), which is not the case for the LW-algorithm. In two parameter persistence, this preprocessing considerably increases the efficiency of the entire computation [72, 73]. In [72, Table 1], chunk preprocessing has been observed to outperform an older approach (see [117]) to decreasing the input size based on discrete Morse theory.

We propose and evaluate an analogous preprocessing scheme based on Algorithm 5; see the procedure `MinimizeCpx*`() in Section 5.1.6. Recently, other preprocessing steps that simplify the input simplicial complex have been proposed [2].

Parallelization The second phase in `Minimize`() (Algorithm 3) can be implemented in an embarrassingly parallel way since no two non-local columns affect each other. The first phase only performs column operations $[D]_j \leftarrow [D_j] + [D_{p_i}]$ if $\mathrm{cg}^D_j = \mathrm{cg}^D_{p_i}$. This implies that if the column indices of D are partitioned into *chunks* of common column grade, the chunks can be processed in parallel in the first phase, hence the name *chunk preprocessing* [11]. In practice, however, the first phase of `Minimize`() has been observed to account for only a small fraction of the total run time of `Minimize`(), and the benefit from parallelizing it has been observed to be relatively limited. The procedure `Factorize`() (Algorithm 7) can be implemented in an embarrassingly parallel way, too, although it only accounts for a small fraction of the total run time of the entire homology resolution computation (Algorithm 9). There is no known way to parallelize `Ker`() or `KerAndMgsWithKer`() (Algorithms 6 and 8), which dominate the rumtime of Algorithm 9.

Minimal free resolutions vs. presentations Often (e.g., in [73, 86, 98]), one is interested in computing a minimal free presentation of persistent homology instead of a minimal free resolution. However, for $\mathbf{Z}^2$-persistence modules, it is not less work to compute the latter than to compute the former. To see this, consider the free resolution (2.18) of $H_d(C_\bullet)$. A free presentation of $H_d(C_\bullet)$ is given by the morphism $p_{d+1} \colon C_{d+1} \to Z_d(C_\bullet)$. To remove all direct summands of p_{d+1} of the form $G \to 0$, it is necessary to compute $\ker p_{d+1} = Z_{d+1}(C_\bullet)$ and use this to remove certain columns of the matrix representing p_{d+1} (which is what `KerAndMgsWithKer`() (Algorithm 8) does). Therefore, to compute a minimal free presentation of a the persistent homology of $\mathbf{Z}^2$-filtered complex one has to carry out the same computations as for computing a minimal free resolution.

Row and column orders As mentioned above (see Proposition 2.3.23), `Minimize`() (Algorithm 3) requires the input graded matrix M to have row and column grades in non-descending order. This requirement is satisfied if rg^M and cg^M are ordered in a total order refining the partial order $\leq$ on $\mathbf{Z}^2$. In the homology resolution computation (Algorithm 9), `Minimize`() is applied to the matrix P'_d. which in turn is computed from I_d and D'_{d+1} using `Factorize`() (Algorithm 7). The matrices I_d and D'_{d+1} are computed using `Ker`() and `KerAndMgsWithKer`() (Algorithms 6 and 8). `Ker`() ensures that I_d has columns in lexicographic order by grade. `Ker`() and `KerAndMgsWithKer`() do not require any specific order on the row and column grades of its input D_d and D_{d+1}. We may thus assume that all boundary matrices have their rows and and columns ordered colexicographically by grade. Then `KerAndMgsWithKer`() computes D'_{d+1} with columns in colexicographic order, too, which together implies that `Factorize`() computes

P'_d with columns in colexicographic order (from D'_{d+1}) and rows in lexicographic order (from I_d). Thus, `Minimize`() can be applied directly to P'_d; i.e., there is no re-ordering needed in Algorithm 9.

Two-parameter persistent cohomology

In the following two chapters, we introduce different approaches to computing the persistent cohomology of a finite one-critical two-parameter filtered simplicial complex K_*. Specifically, we explore different ways to compute a minimal free resolution of $H_\bullet(K_*)$, using cohomology as an intermediate step. Our goal is to apply this to two-parameter clique filtrations such as function-Rips complexes. Using cohomology is motivated by the effectiveness of clearing in the computation of one-parameter persistent cohomology of Vietoris–Rips complexes.

Chapter 3 presents an approach that computes two-parameter persistent (co)homology by computing the cohomology of a certain cochain complex $N^\bullet(K_*)$ of free modules. In general, $N^\bullet(K_*)$ is different from $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$, but relates to these by a certain property of $\mathrm{vec}^{\mathbf{Z}^n}$ called the *Calabi–Yau property*, see Section 3.2. This property is crucial both for an efficient computation of a minimal free resolution of $H^\bullet(N^\bullet(K_*))$, and for obtaining a minimal free resolution of $H_\bullet(K_*)$ from it. Parts of this approach can be generalized to an arbitrary number of parameters. In particular, for one parameter, this approach gives back the usual computation of relative persistent cohomology. The approach can be implemented efficiently; see Section 5.1.

In Chapter 4, we explain how minimal free resolutions of $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ can be computed directly from the cochain complexes $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$. For two and more parameters, these are no complexes of free modules, and the main part of Chapter 4 lies in remedying this. While this approach arguably is conceptually simpler than the aforementioned one, when it comes to implementing it, it comes with certain drawbacks that we explain in Section 4.5.

Chapter 3

Persistent cohomology using the Calabi–Yau property

Let K_* be a finite one-critical $\mathbf{Z}^n$-filtered simplicial complex, for any $n \geq 1$. Recall from Section 2.2.1 that in one-parameter persistence, the relative cochain complex $C^\bullet(K, K_*)$ is a cochain complex of free modules. In this section, we construct a cochain complex $N^\bullet(K_*)$ of free persistence modules such that $n = 1$ yields $N^\bullet(K_*) = C^\bullet(K, K_*)$. The construction of $N^\bullet(K_*)$ can thus be seen as a way to generalize the definition of $C^\bullet(K, K_*)$ to more than one parameter.

We prove that if $H_d(K_*)$ is finitely supported for all d, then $H_d(K_*) \cong H^{d+n}(N^\bullet(K_*))^*$; see Theorem A. This implies that duality gives a correspondence between injective resolutions of $H^{d+n}(N^\bullet(K_*))$ and free resolutions of $H_d(K_*)$. We show that under the same conditions, there is a direct correspondence between injective and free resolutions of $H^{d+n}(N^\bullet(K_*))$ (or any other finite dimensional module); see Corollary 3.2.11 and Theorem E. For $n = 2$, we devise an efficient method to compute a minimal free resolution of $H^{d+2}(N_\bullet)$ that involves a scheme akin to clearing, see Section 3.5. This method underlies the implementation presented in Section 5.1.

Parts of the results presented in this chapter are joint work with Ulrich Bauer and Michael Lesnick, and have appeared in [14].

3.1 The free cochain complex $N^\bullet(K_*)$

Definition 3.1.1. For a module M and $z \in \mathbf{Z}^n$, let $M\langle z\rangle$ be the module with graded components $M\langle z\rangle_w = M_{z+w}$. For $z \geq 0$, the structure maps of M give a morphism $M \to M\langle z\rangle$. Note that $M\langle z\rangle^* = M^*\langle -z\rangle$. For a graded matrix A, let $A\langle z\rangle$ be the graded matrix with $A\langle z\rangle_{ij} = A_{ij}$, $\mathrm{rg}_i^{A\langle z\rangle} = \mathrm{rg}_i^A - z$ and $\mathrm{cg}_j^{A\langle z\rangle} = \mathrm{rg}_j^A - z$ for all i, j.

This ensures that if $A\colon F \to F'$ is a valid graded matrix representing a morphism of free modules, then $A\langle z\rangle\colon F\langle z\rangle \to F'\langle z'\rangle$. Let K_* be a one-critical $\mathbf{Z}^n$-filtered simplicial complex. We fix an ordering on the simplices of K_* and endow $C_\bullet(K_*) = \bigoplus_{\sigma \in K_*} F(g(\sigma))$ with the standard basis. Let the graded matrix $D_\bullet$ represent the boundary operator $\partial_\bullet$. Let $\epsilon = (1, \ldots, 1) \in \mathbf{Z}^n$.

Definition 3.1.2. Let $N^\bullet(K_*)$ be the cochain complex of free modules

$$N^d(K_*) = \bigoplus_{\sigma \in K_*^d} F(\epsilon - g(\sigma)),$$

such that the coboundary operator $\delta^d\colon N^{d-1}(K_*) \to N^d(K_*)$ is represented by the graded matrix $D^d \coloneqq D_d^\top$ for every d.

Example 3.1.3. If $n = 1$, then $N^\bullet(K_*) = C^\bullet(K, K_*)$.

According to Remark 2.1.19, D^d is valid, so δ^d is a well-defined morphism for every d. Our first goal is to prove the following statement:

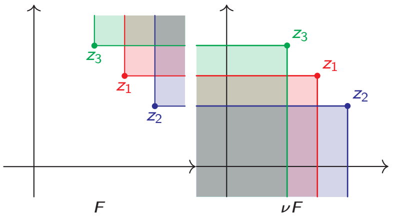


Figure 3.1: A free module $F = F(z_1) \oplus F(z_2) \oplus F(z_3)$ (left) and its image $\nu F = I(z_1) \oplus I(z_2) \oplus I(z_3)$ under the Nakayama functor (right)

Theorem A. *A If $H_\bullet(K_*)$ has finite support, then there is a natural isomorphism $H_d(K_*) \cong H^{d+n}(N^\bullet(K_*))^*$ for all d.*

We defer the proof of Theorem A to the next subsection; see page 45. The condition that $H_\bullet(K_*)$ has finite total dimension equivalently means that $H_d(K_*)$ is pointwise finite dimensional for all d and has finite support in $\mathbf{Z}^n$. Once we have proven Theorem A, we obtain:

Corollary 3.1.4. *If $H_\bullet(K_*)$ has finite support and $F_\bullet$ is a free resolution of $H^{d+n}(N^\bullet(K_*))$, then $(F_\bullet)^*$ is an injective resolution of $H_d(K_*)$.*

Proof. Since the duality $(-)^*\colon \mathrm{Vec}^{\mathbf{Z}^n} \to \mathrm{Vec}^{\mathbf{Z}^n}$ is contravariant exact, it maps free resolutions to injective resolutions and vice versa. □

3.2 The Calabi-Yau-property of persistence modules

Assuming for now that Theorem A has been established, Corollary 3.1.4 gives us an injective resolution of $H_d(K_*)$. It remains to compute from this the *free* resolution of $H_d(K_*)$ we are interested in. Both Theorem A and a simple correspondence between free and injective resolutions of the same module will follow from Theorem 3.2.8, which is closely related to a property known in some areas of algebra as the *Calabi–Yau property* of a triangulated category [77].

Definition 3.2.1. For persistence modules M, N, let $\mathsf{Hom}(M,N)$ be the persistence module with components $\mathsf{Hom}(M,N)_z = \mathrm{Hom}(M, N\langle z\rangle)$. The structure maps of $\mathsf{Hom}(M,N)$ are induced by $N \to N\langle z\rangle$.

For $z \in \mathbf{Z}^n$, we define the injective module $I(z) = F(-z)^*$; that is,

$$I(z)_w = \begin{cases} k \text{ if } w \le z, \\ 0 \text{ otherwise,} \end{cases} \qquad I(z)_{w \le w'} = \begin{cases} \text{id if } w' \le z, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{P}_{\mathrm{Vec}^{\mathbf{Z}^n}}$ and $\mathcal{I}_{\mathrm{Vec}^{\mathbf{Z}^n}}$ be the full subcategories of $\mathrm{Vec}^{\mathbf{Z}^n}$ consisting of finite rank free modules and their dual modules, respectively. Note that $\mathcal{P}_{\mathrm{Vec}^{\mathbf{Z}^n}}$ contains all point-wise finite dimensional projective modules modules. Note also that all modules in $\mathcal{I}_{\mathrm{Vec}^{\mathbf{Z}^n}}$ are injective, but $\mathcal{I}_{\mathrm{Vec}^{\mathbf{Z}^n}}$ does not comprise all injective modules, not even all point-wise finite dimensional ones. Both categories are closed under finite direct sums.

Definition 3.2.2. The functors

$$\begin{aligned} \nu &= \mathsf{Hom}(-, F(0))^*\colon \mathcal{P}_{\mathrm{Vec}^{\mathbf{Z}^n}} \longrightarrow \mathcal{I}_{\mathrm{Vec}^{\mathbf{Z}^n}}, \\ \nu' &= \mathsf{Hom}(I(0)^*, -)\colon\ \mathcal{I}_{\mathrm{Vec}^{\mathbf{Z}^n}} \longrightarrow \mathcal{P}_{\mathrm{Vec}^{\mathbf{Z}^n}} \end{aligned}$$

are called the *Nakayama* and the *inverse Nakayama functor.*

Lemma 3.2.3 ([3, p. 84])**.** *The functors ν and ν' are mutually quasi-inverse equivalences of categories.*

This follows from the following:

Lemma 3.2.4. *We have $\nu F(z) \cong I(z)$ and $\nu^{-1} I(z) \cong F(z)$, cf. Figure 3.1.*

Proof. We note

$$(\mathsf{Hom}(F(z), F(0))^*)_w = (\mathsf{Hom}(F(z), F(0))_{-w})^* = \mathsf{Hom}(F(z), F(w))^* \cong \{ \begin{smallmatrix} k \text{ if } z \geq w, \\ 0 \text{ otherwise.} \end{smallmatrix}$$

For $z \geq w$, let $\iota_{wz} \colon F(z) \hookrightarrow F(w)$ be the morphism with $(\iota_{wz})_v = \mathrm{id}_k$ for all $v \geq z$. Then the isomorphism $\nu F(z) \cong I(z)$ is given in graded components by

$$\begin{aligned} \phi \colon \mathsf{Hom}(F(z), F(w))^* &\longrightarrow I(z)_w = k, \\ \psi &\longmapsto \psi(\iota_{wz}). \end{aligned}$$

The statement $\nu^{-1} I(z) \cong F(z)$ follows analogously. □

In particular, we obtain

$$N^\bullet(K_*) = (\nu C_\bullet(K_*)\langle \epsilon \rangle)^*.$$

Remark. If $n = 1$ and $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}}$ is a finite filtered complex, then $C^\bullet(K_*, K) = \nu C_\bullet(K_*)$.

Definition 3.2.5. For a chain complex $C_\bullet$ and $i \in \mathbf{Z}$, let $C_\bullet[i]$ be the chain complex with d-dimensional component $(C_\bullet[i])_d = C_{i+d}$. Analogously, for a cochain complex $C^\bullet$, let $C^\bullet[i]$ be the cochain complex with $(C^\bullet[i])^d = C^{i+d}$. We have $C_\bullet[i]^* = (C_\bullet)^*[i]$.

For $z \in \mathbf{Z}^n$, we write $z = (z_1, \ldots, z_n)$. For $n \in \mathbf{N}$, let $[n] := \{1, \ldots, n\}$ and let $\binom{[n]}{k} := \{S \subseteq [n] \mid |S| = k\}$.

Definition 3.2.6. For $S = \{s_1 < \ldots < s_k\} \in \binom{[n]}{k}$, we define the functors

$$\begin{aligned} \mathrm{colim}_S \colon \mathrm{Vec}^{\mathbf{Z}^n} &\to \mathrm{Vec}^{\mathbf{Z}^{n-k}}, & (\mathrm{colim}_S M)_{(z_1, \ldots, \hat{z}_{s_1}, \ldots, \hat{z}_{s_k}, \ldots, z_k)} &= \mathrm{colim}_{(z_{s_1}, \ldots, z_{s_k}) \in \mathbf{Z}^k} M_z, \\ \Delta_S \colon \mathrm{Vec}^{\mathbf{Z}^{n-k}} &\to \mathrm{Vec}^{\mathbf{Z}^n}, & (\Delta_S M)_{(z_1, \ldots, z_n)} &= M_{(z_1, \ldots, \hat{z}_{s_1}, \ldots, \hat{z}_{s_{n-k}}, \ldots, z_n)}, \\ \lim_S \colon \mathrm{Vec}^{\mathbf{Z}^n} &\to \mathrm{Vec}^{\mathbf{Z}^{n-k}}, & (\lim_S M)_{(z_1, \ldots, \hat{z}_{s_1}, \ldots, \hat{z}_{s_k}, \ldots, z_k)} &= \lim_{(z_{s_1}, \ldots, z_{s_k}) \in \mathbf{Z}^k} M_z, \end{aligned}$$

where $\hat{z}_{s_1}, \ldots, \hat{z}_{s_k}$ are omitted. We also define $\mathrm{Colim}_S := \Delta_S \,\mathrm{colim}_S$ and $\mathrm{Lim}_S := \Delta_S \,\mathrm{Lim}_S$.

That is, $\mathrm{colim}_S M$ is the $(n - |S|)$-parameter module obtained by taking the colimit of M with respect to the axes indexed by S. For example, if $n = 3$ and $S = \{1, 3\}$, then $(\mathrm{colim}_{\{1,3\}} M)_z = \mathrm{colim}_{(z_1, z_3) \in \mathbf{Z}^2} M_{(z_1, z, z_3)}$. The module $\Delta_S M$ (and thus also $\mathrm{Colim}_S M$ and $\mathrm{Lim}_S M$) is constant along the coordinate axes specified by S. The functor Δ_S, called the *S-diagonal*, is exact, right adjoint to colim_S and left adjoint to $\lim_S$.

Definition 3.2.7. A chain complex $F_\bullet$ in $\mathrm{Vec}^{\mathbf{Z}^n}$ is called *eventually acyclic* if $\mathrm{colim}_S F_\bullet$ is acyclic for all non-empty $S \subseteq [n]$. A $\mathbf{Z}^n$-filtered simplicial simplicial complex K_* is *eventually acyclic* if $\mathrm{colim}_S K_*$ is acyclic for all non-empty $S \subseteq [n]$.

A chain complex of free persistence modules is eventually acyclic if and only if $\mathrm{supp}\, H_d(F_\bullet)$ is bounded for all d.

Theorem 3.2.8 (Calabi–Yau property of persistence modules)**.** *If $F_\bullet$ is an eventually acyclic complex of free $\mathbf{Z}^n$-persistence modules, then $F_\bullet$ and $\nu F_\bullet[n]\langle \epsilon \rangle$ are naturally quasi-isomorphic.*

Before proving Theorem 3.2.8, we introduce some additional terminology that we will also use later.

For a module M, we define the modules

$$\Omega_k M = \bigoplus_{S \in \binom{[n]}{n-k}} \mathrm{Colim}_S M$$

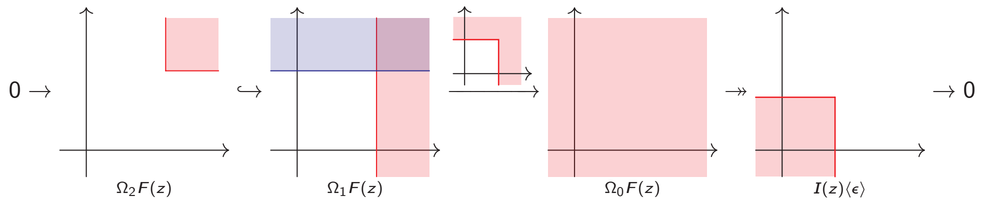


Figure 3.2: The Koszul complex from (3.2) for $n = 2$. The sequences are exact, and the module above the arrow denotes the image of that morphism.

for each $0 \le k \le n$. If $S' \subseteq S$, then there is a canonical morphism $c_{S,S'} \operatorname{Colim}_{S'} M \to \operatorname{Colim}_S M$. These give rise to morphisms

$$\Omega_k M = \bigoplus_{S'\in\binom{[n]}{n-k}} \operatorname{Colim}_{S'} M \xrightarrow{\kappa_k} \bigoplus_{S\in\binom{[n]}{n-k+1}} \operatorname{Colim}_S M = \Omega_{k-1} M$$

that are defined by their non-zero components

$$(-1)^j c_{S,S\setminus\{s_j\}} \colon \operatorname{Colim}_{S\setminus\{s_j\}} M \to \operatorname{Colim}_S M$$

for $S = (s_1 < \cdots < s_j < \cdots < s_{n-k})$. The sign rule ensures that $(\Omega_\bullet M, \kappa_\bullet)$ is a chain complex.

Lemma 3.2.9. *The assignment $M \mapsto \Omega_\bullet M$ is a functor.*

Definition 3.2.10. We call the functor $\Omega_\bullet$ the *Koszul complex functor*.

Remark. The complex from Remark 2.4.1 is not an instance of this functor, but follows the same combinatorics. There exist several different but closely related notions of "the" Koszul complex.

Proof of Theorem 3.2.8. For a free module $F(z)$, the sequence

$$0 \to \Omega_n F(z) \to \Omega_{n-1} F(z) \to \cdots \to \Omega_0 F(z) \to \nu F(z)\langle\epsilon\rangle. \tag{3.1}$$

is exact. The last morphism is the canonical morphism

$$\Omega_0 F(z) = \operatorname{Colim} F(z) = \operatorname{Lim} I(z)\langle\epsilon\rangle \to I(z)\langle\epsilon\rangle = \nu F(z)\langle\epsilon\rangle.$$

One may check (e.g., using [65, Corollary 6.6 or Theorem A6.6]) that all modules $\Omega_k F(z)$ are flat[1], so (3.1) sequence is a flat resolution of $\nu F(z)\langle\epsilon\rangle$. see Figure 3.2. Let $F_\bullet$ be a bounded complex of free modules. Using functoriality of $\Omega_\bullet$ and ν, we get an exact sequence

$$\Omega_\bullet F_\bullet \colon \quad 0 \to \underbrace{\Omega_n F_\bullet}_{F_\bullet} \to \Omega_{n-1} F_\bullet \to \cdots \to \Omega_0 F_\bullet \to \nu F_\bullet\langle\epsilon\rangle \to 0 \tag{3.2}$$

of chain complexes, given by taking a shifted copy of (3.1) for every summand $F(z)$ in a direct sum decomposition of $F_\bullet$. We unsplice (3.2) into short exact sequences

$$\begin{tikzcd}
 & & & 0 \arrow[dr] & & 0 & 0 \arrow[dr] & & & & & \\
 & & & & U_\bullet^{(n-1)} \arrow[ur] \arrow[dr] & & & U_\bullet^{(1)} \arrow[dr] & & & & \\
0 \arrow[r] & F_\bullet \arrow[rr] \arrow[dr, equal] & & \Omega_{n-1}F_\bullet \arrow[ur] \arrow[rr] & & \Omega_{n-2}F_\bullet \arrow[rrr, "\cdots" description] \arrow[dr] & & & \Omega_0 F_\bullet \arrow[rr] \arrow[dr] & & \nu F_\bullet\langle\epsilon\rangle \arrow[r] & 0 \\
 & & F_\bullet \arrow[ur] & & & & U_\bullet^{(n-2)} \arrow[dr] & & & \nu F_\bullet\langle\epsilon\rangle \arrow[ur, equal] \arrow[dr] & & \\
 & 0 \arrow[ur] & & & & & & 0 & & & 0 &
\end{tikzcd} \tag{3.3}$$

[1] A graded module F is *flat* if the graded module tensor product functor $- \otimes F$ is exact.

with chain complexes $U_\bullet^{(k)}$ for each k. In the derived category $\mathcal{D}^{\mathrm{b}}(\mathrm{Vec}^{\mathbf{Z}^n})$ (see Section 3.2.1 below for the notion of the derived category), each of these short exact sequences gives a triangle [127, §10.4.9]. We obtain connecting homomorphisms

$$\partial^{(n-1)}\colon U_\bullet^{(n-1)}[1] \to F_\bullet, \qquad \partial^{(n-2)}\colon U_\bullet^{(n-2)}[1] \to U_\bullet^{(n-1)}, \qquad \ldots \qquad \partial^{(0)}\colon \nu F_\bullet\langle\epsilon\rangle[1] \to U_\bullet^{(1)}$$

in $\mathcal{D}^{\mathrm{b}}(\mathrm{Vec}^{\mathbf{Z}^n})$. These descend to maps fitting into the long exact sequences

$$\begin{gathered}
\cdots \longrightarrow H_{d+1}(\Omega_{n-1}F_\bullet) \longrightarrow H_{d+1}(U_\bullet^{(n-1)}) \xrightarrow{\partial^{(n-1)}} H_d(F_\bullet) \longrightarrow H_d(\Omega_{n-1}F_\bullet) \longrightarrow \cdots, \\
\cdots \longrightarrow H_{d+2}(\Omega_{n-2}F_\bullet) \longrightarrow H_{d+2}(U_\bullet^{(n-2)}) \xrightarrow{\partial^{(n-2)}} H_{d+1}(U_\bullet^{(n-1)}) \longrightarrow H_{d+1}(\Omega_{n-2}F_\bullet) \longrightarrow \cdots, \\
\vdots \\
\cdots \longrightarrow H_{d+n}(\Omega_0 F_\bullet) \longrightarrow H_{d+n}(\nu F_\bullet\langle\epsilon\rangle) \xrightarrow{\partial^{(0)}} H_{d+n-1}(U_\bullet^{(1)}) \longrightarrow H_{d+n-1}(\Omega_0 F_\bullet) \longrightarrow \cdots,
\end{gathered} \tag{3.4}$$

induced by the short exact sequences (3.3). By assumption, $\operatorname{Colim}_S H_d(F_\bullet) = 0$ for all d and for all S with $|S| > 0$. The functor Colim_S is exact for all S because it is a directed colimit. In particular,

$$H_d(\Omega_k F_\bullet) = \textstyle\bigoplus_{|S|=k} H_d(\operatorname{Colim}_S F_\bullet) = 0$$

for all $k < n$. Therefore, the long exact sequences (3.4) show that all connecting homomorphisms $\partial^{(k)}$ are quasi-isomorphisms. Thus,

$$\partial^{(n-1)} \circ \cdots \circ \partial^{(0)}\colon \nu F_\bullet[n]\langle\epsilon\rangle \longrightarrow F_\bullet$$

is a quasi-isomorphism. □

A proof in a more general context can be found in [84, Lemma 4.1].

Proof of Theorem A. With $N^\bullet(K_*) = (\nu C_\bullet(K_*)\langle\epsilon\rangle)^*$, Theorem 3.2.8 gives

$$H^{d+n}(N^\bullet(K_*))^* \cong H_d(N^\bullet(K_*)^*[n]) \cong H_d(\nu C_\bullet(K_*)\langle\epsilon\rangle[n]) \cong H_d(C_\bullet(K_*)) = H_d(K_*). \qquad □$$

Corollary 3.2.11. *Let $M \in \mathrm{Vec}^{\mathbf{Z}^n}$ be finitely supported.*

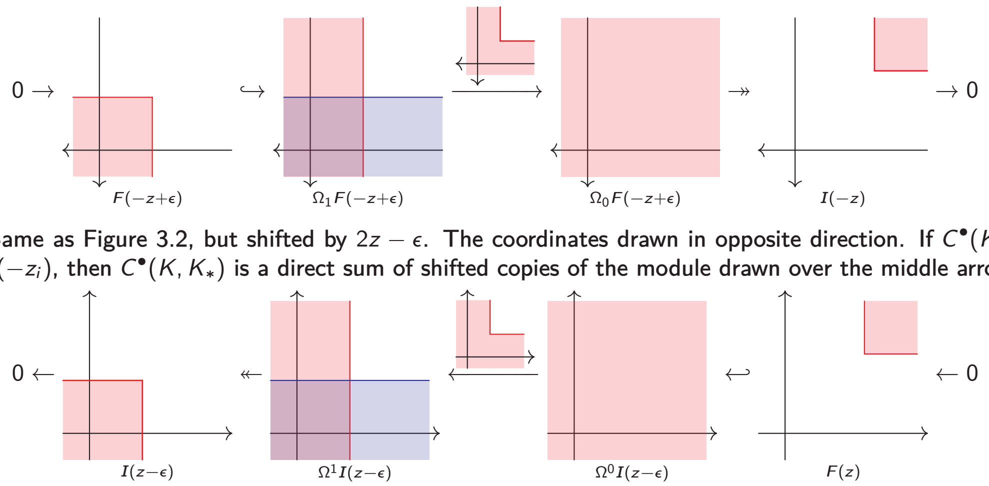
(i) If $F_\bullet$ is a free resolution of M, then $\nu F_\bullet[n]\langle\epsilon\rangle$ is an injective resolution of M.
(ii) If $I^\bullet$ is an injective resolution of M, then $\nu^{-1}I^\bullet[-n]\langle-\epsilon\rangle$ is a free resolution of M.
(iii) If $F_\bullet$ is a free resolution of M, then $(\nu F_\bullet[n]\langle\epsilon\rangle)^*$ is a free resolution of M^*.

3.2.1 Remarks on Theorem 3.2.8

One-parameter persistence For $n = 1$, we have $C_\bullet(K, K_*) \cong \nu C_\bullet(K_*)\langle 1\rangle$. If K is acyclic, then Theorem 3.2.8 gives back the isomorphism $H_d(K_*) \cong H_{d+1}(K, K_*)$ we already obtained from (2.2).

Relation to derived categories Recall that the *derived category* $\mathcal{D}(\mathcal{C})$ of an abelian category $\mathcal{C}$ is the triangulated category whose objects are chain complexes of persistence modules, and whose morphisms are $\mathrm{Hom}_{\mathcal{D}(\mathcal{C})}(X, Z) = \{X_\bullet \xleftarrow{\simeq} Y_\bullet \to Z_\bullet\}/{\sim}$, where $X_\bullet \xleftarrow{\simeq} Y_\bullet$ is a quasi-isomorphism and $\sim$ is a certain equivalence relation. That is, the derived category is obtained via formally inverting quasi-isomorphisms by imposing a calculus of fractions [124], analogous to the construction of localization of rings; see [127, §10] for details.

The *bounded derived category* $\mathcal{D}^{\mathrm{b}}(\mathrm{Vec}^{\mathbf{Z}^n})$ is the full subcategory of $\mathcal{D}(\mathrm{Vec}^{\mathbf{Z}^n})$ of chain complexes $C_\bullet$ whose homology has finite total dimension [84, §4.1]; that is, $H_d(C_\bullet)_z = 0$ for all but finitely many d and z. Let $\mathcal{K}^{\mathrm{b}}(\mathcal{P}_{\mathrm{Vec}^{\mathcal{C}}})$ and $\mathcal{K}^{\mathrm{b}}(\mathcal{I}_{\mathrm{Vec}^{\mathcal{C}}})$ denote the respective bounded homotopy



(a) Same as Figure 3.2, but shifted by $2z - \epsilon$. The coordinates drawn in opposite direction. If $C^\bullet(K_*) = \bigoplus_i I(-z_i)$, then $C^\bullet(K, K_*)$ is a direct sum of shifted copies of the module drawn over the middle arrow.

(b) The Koszul complex from (3.6). Dual of **(a)**. If $C_\bullet(K_*) = \bigoplus_i F(z_i)$, then $C_\bullet(K, K_*)$ is a direct sum of shifted copies of the module drawn over the middle arrow.

Figure 3.3: Illustration of the variations of the Koszul complex for $n = 2$. The sequences are exact, and the module above the arrow denotes the image of that morphism.

categories; that is, $\mathcal{K}^{\mathrm{b}}(\mathcal{P}_{\mathrm{Vec}^{\mathcal{C}}})$ (and analogously, $\mathcal{K}^{\mathrm{b}}(\mathcal{I}_{\mathrm{Vec}^{\mathcal{C}}})$) is the triangulated category whose objects are chain complexes of projective (resp., injective) persistence modules with homology of finite total dimension, and whose morphisms are chain homotopy classes of morphisms of chain complexes. Recall that $\mathcal{D}^{\mathrm{b}}(\mathcal{C})$, $\mathcal{K}^{\mathrm{b}}(\mathcal{P}_{\mathrm{Vec}^{\mathcal{C}}})$ and $\mathcal{K}^{\mathrm{b}}(\mathcal{I}_{\mathrm{Vec}^{\mathcal{C}}})$ are equivalent as triangulated categories. We note:

1. Theorem 3.2.8 states that $\nu \cong [-n]\langle -\epsilon \rangle$ as endo-functors of $\mathcal{D}^{\mathrm{b}}(\mathrm{Vec}^{\mathbf{Z}^n})$.

2. The Koszul complex $\Omega_\bullet F(0)$ concentrated in degrees $n, \dots, 0$ satisfies $- \otimes \Omega_\bullet F(0) \cong \nu\langle \epsilon \rangle$ as endo-functors of $\mathcal{D}^{\mathrm{b}}(\mathrm{Vec}^{\mathbf{Z}^n})$. Note that there exist different but similar definitions of "the" Koszul complex, which all follow the same combinatorics, e.g., the complex defined in Remark 2.4.1 or the one from [84, §4.2].

3. A triangulated category $\mathcal{T}$ is *n-Calabi–Yau* if $[n]$ is a *Serre functor* [47, §2, 77]; that is, if $\mathrm{Hom}_{\mathcal{T}}(A_\bullet, B_\bullet)^* \cong \mathrm{Hom}_{\mathcal{T}}(B_\bullet, A_\bullet[n])$ for all $A_\bullet, B_\bullet$. One can show that $[n]$ is a Serre functor if $\nu \cong [-n]$; see [84, Lemma 4.1]. In [84, §4.2], it is shown that $\mathcal{D}^{\mathrm{b}}(k[x_1, \dots, x_n])$ is n-Calabi–Yau, using a slightly different Koszul complex than we do. Theorem 3.2.8 can be viewed as a $\mathbf{Z}^n$-graded version of this.

Relation to relative cochains We can also state a dual version of Theorem 3.2.8:

Theorem 3.2.12. *If $F^\bullet$ is a cochain complex of free modules such that $H^\bullet(F^\bullet)$ is finitely supported, then $F^\bullet \simeq \nu F^\bullet\langle \epsilon \rangle[-n]$.*

Proof. The proof is analogous to the proof of Theorem 3.2.8, with the commutative diagram

$$\begin{tikzcd}
 & & & 0 \arrow[dr] & & 0 & 0 \arrow[dr] & & & & & \\
 & & & & U^\bullet_{(n-1)} \arrow[ur] \arrow[dr] & & & U^\bullet_{(1)} \arrow[dr] & & & & \\
0 \arrow[r] & F^\bullet \arrow[rr] \arrow[dr, equal] & & \Omega_{n-1}F^\bullet \arrow[rr] \arrow[ur] & & \Omega_{n-2}F^\bullet \arrow[rrr, "\cdots" description] \arrow[dr] & & & \Omega_0 F^\bullet \arrow[rr] \arrow[dr] & & \nu F^\bullet\langle\epsilon\rangle \arrow[r] & 0 \\
 & & F^\bullet \arrow[ur] & & & & U^\bullet_{(n-2)} \arrow[dr] & & & \nu F^\bullet\langle\epsilon\rangle \arrow[ur, equal] \arrow[dr] & & \\
 & 0 \arrow[ur] & & & & & & 0 & & & 0 &
\end{tikzcd} \tag{3.5}$$

of cochain complexes. □

Corollary 3.2.13. *If* $\operatorname{Colim}_S K_*$ *is acyclic for all* $S \in \binom{[n]}{k}$ *for* $0 < k < n$, *then*

$$H^{d+n}(N^\bullet(K_*)) \cong H^{d+1}(K, K_*).$$

Proof. With $F^\bullet = N^\bullet(K_*)$ and $\nu N^\bullet\langle\epsilon\rangle = C^\bullet(K_*)$, we obtain $U^\bullet_{(1)} = C^\bullet(K, K_*)$; see Figure 3.3a. From (3.5), we obtain the sequence

$$H^{d+n}(N^\bullet(K_*)) \longrightarrow H^{d+n-1}(U^\bullet_{(n-1)}) \longrightarrow \cdots \longrightarrow H^{d+1}(K, K_*) \longrightarrow H^d(K_*)$$

of connecting homomorphisms. If $\operatorname{Colim}_S K_*$ is acyclic for all $S \in \binom{[n]}{k}$ for $0 < k < n$, then $\Omega_k F^\bullet$ is acyclic for all $0 < k < n$, so this is a sequence of isomorphisms. □

Proof using limits Alternatively, Theorem 3.2.8 can be proven using limits instead of colimits. For $1 \leq k \leq n$, let $\Omega^k M\colon \bigoplus_{S \in \binom{[n]}{n-k}} \operatorname{Lim}_S M$. Then there is an exact sequence

$$0 \to \nu^{-1} I(z)\langle\epsilon\rangle \to \Omega^0 I(z)\langle\epsilon\rangle \to \cdots \to \Omega^{n-1} I(z)\langle\epsilon\rangle \to \underbrace{\Omega^n I(z)\langle\epsilon\rangle}_{I(z)\langle\epsilon\rangle} \to 0; \tag{3.6}$$

see Figure 3.3b. Let $F_\bullet$ be a chain complex of free modules, and let $I_\bullet = \nu F_\bullet[n]\langle\epsilon\rangle$ such that $I_\bullet \simeq F_\bullet$. Taking shifted copies of (3.6) and unsplicing the sequence into short exact sequence analogously to (3.4) yields a commutative diagram

$$\begin{tikzcd}
 & & & 0 \arrow[dr] & & 0 & 0 \arrow[dr] & & & & & \\
 & & & & U'^{(1)}_\bullet \arrow[ur] \arrow[dr] & & & U'^{(n-1)}_\bullet \arrow[dr] & & & & \\
0 \arrow[r] & F_\bullet \arrow[rr] \arrow[dr, equal] & & \Omega^0 I_\bullet \arrow[rr] \arrow[ur] & & \Omega^1 I_\bullet \arrow[rrr, "\cdots" description] \arrow[dr] & & & \Omega^{n-1} I_\bullet \arrow[rr] \arrow[dr] & & I_\bullet \arrow[r] & 0 \\
 & & I_\bullet \arrow[ur] & & & & U'^{(2)}_\bullet \arrow[dr] & & & I_\bullet \arrow[ur, equal] \arrow[dr] & & \\
 & 0 \arrow[ur] & & & & & & 0 & & & 0 &
\end{tikzcd} \tag{3.7}$$

for suitable chain complexes $U'^{(k)}_\bullet$. Note that (3.7) is the dual of (3.5).

3.3 The Nakayama functor and matrices

It remains to describe the injective resolution $\nu F_\bullet[n]\langle\epsilon\rangle$ of M in terms of matrices representing $F_\bullet$. To that end, we use matrices also to describe morphisms of *injective* modules. We say an injective module I is of *finite type* if $I \in \mathcal{I}_{\mathrm{Vec}^{\mathrm{Vec}}\mathbf{Z}^n}$. Note that such a module is of the form $I \cong \bigoplus_{i \in I} I(z_i)$ for some finite indexing set I.

Recall that a basis of a free module F can be identified with the choice of an isomorphism $\bigoplus_{z \in \operatorname{rk} F} F(z) \to F$.

Definition 3.3.1. An *injective basis* of an injective module I of finite type is an isomorphism $I \cong \bigoplus_{i \in I} I(z_i)$.

Lemma 3.3.2. *A graded matrix M represents a morphism $\bigoplus_j I(\mathrm{cg}_j^M) \to \bigoplus_i I(\mathrm{rg}_i^M)$ if and only if M is valid.*

Proof. This follows from $\operatorname{Hom}(I(z), I(z')) = \left\{ \begin{smallmatrix} k \text{ if } z' \leq z \\ 0 \text{ otherwise,} \end{smallmatrix} \right.$. □

In particular, a graded matrix represents a morphism of free modules if and only if it represents a morphism of injective modules.

Lemma 3.3.3. *A graded matrix represents a morphism $f \colon \bigoplus_{j=1}^n F(z_j) \to \bigoplus_{i=1}^m F(z_i')$ if and only if it represents the morphism $\nu f \colon \bigoplus_{j=1}^n I(z_j) \to \bigoplus_{i=1}^m I(z_i')$.*

Proof. The functor ν is additive. Therefore, it suffices to check the statements for morphisms $f \colon F(z) \to F(z')$. If $z' \not\leq z$, then $\operatorname{Hom}(F(z), F(z') = \operatorname{Hom}(I(z), I(z')) = 0$, so let $z' \leq z$. Any morphism $f \colon F(z) \to F(z')$ is of the form $f = \lambda \iota_{z'z}$ for some $\lambda \in k$, where $\iota_{z'z}$ $\iota_{z'z} \colon F(z) \hookrightarrow F(z')$ is the injective morphism with $(\iota_{z'z})_v = \mathrm{id}_k$ for all $v \geq z$, and zero otherwise. Note that $\iota_{z''z'}\iota_{z'z} = \iota_{z''z}$ for $z'' \leq z' \leq z$. Analogously, any morphism $g \colon I(z) \to I(z')$ is of the form $g = \gamma \pi_{z'z}$ for some $\gamma \in k$, where $\pi_{z'z} \colon I(z) \to I(z')$ is the injective morphism with $(\pi_{z'z})_v = \mathrm{id}_k$ for all $v \leq z'$, and zero otherwise. Given a morphism $f = \lambda \iota_{z',z}$ for some $\lambda \in k$, we calculate

$$\begin{array}{cccc} I(z)_w \xrightarrow{\phi} \nu F(z)_w = \mathsf{Hom}(F(z), F(w))^* & \xrightarrow{\nu f} & \mathsf{Hom}(F(z'), F(w))^* = \nu F(z') & \xrightarrow{\phi^{-1}} I(z'), \\ \mu \longmapsto \qquad [\phi(\mu) \colon \iota_{wz} \mapsto \mu] & \longmapsto & [\iota_{wz'} \mapsto \phi(\mu)(\iota_{wz'} f) = \lambda\mu] & \longmapsto \lambda\mu, \end{array}$$

where we use the isomorphism $\phi \colon I(z) \to \nu F(z)$ from Lemma 3.2.4. Hence, $\nu f = \lambda \pi_{z',z}$. □

Theorem E. *Let M be a finitely generated n-parameter persistence module with bounded support. For graded matrices $U_1, \ldots, U_n$, the following are equivalent:*

(*i*) *$U_1, \ldots, U_n$ represent a free resolution $0 \to F_n \xrightarrow{U_n} \cdots \xrightarrow{U_1} F_0$ of M,*

(*ii*) *$U_1, \ldots, U_n$ represent an injective resolution of $I^0 \xrightarrow{U_n} \cdots \xrightarrow{U_1} I^n \to 0$ of $M\langle -\epsilon \rangle$,*

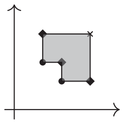
(*iii*) *the graded transposes $U_1^\top, \ldots, U_n^\top$ represent a free resolution $0 \to G_0 \xrightarrow{U_n^\top} \cdots \xrightarrow{U_1^\top} G_0$ of $M^*\langle \epsilon \rangle$.*

In this case $I^q = \nu F_{n-q} = G_q^*$.

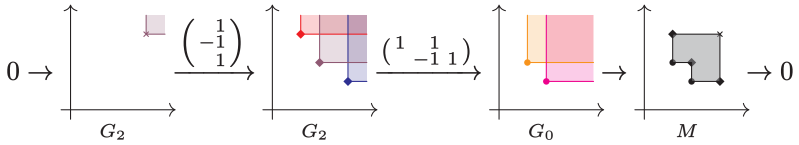
Proof. With Lemma 3.3.3, this follows from Theorem 3.2.8. □

Example 3.3.4. Let $M \in \mathrm{Vec}^{\mathbf{Z}^2}$ be the module

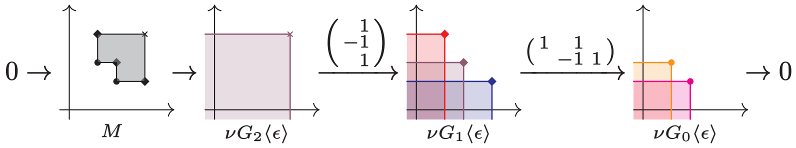
$$M =$$



with components $M_z = k$ if z lies in the shaded region in the following picture, and $M_z = 0$ otherwise. All structure morphisms between non-zero components of M are identities. Here, the symbols •, ⬩ and × denote the grades of the 0-, 1- and 2-syzygies of M, respectively. The sequence


$0 \to$
G_2
$\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$
G_2
$\begin{pmatrix} 1 & 1 & \\ & -1 & 1 \end{pmatrix}$
G_0
$\to$
M
$\to 0$

is exact, so $G_\bullet$ is a free resolution of M. With the same matrices, the sequence


$0 \to$
M
$\to$
$\nu G_2\langle\epsilon\rangle$
$\begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}$
$\nu G_1\langle\epsilon\rangle$
$\begin{pmatrix} 1 & 1 & \\ & -1 & 1 \end{pmatrix}$
$\nu G_0\langle\epsilon\rangle$
$\to 0$

is exact, so $\nu G_\bullet[2]\langle\epsilon\rangle$ is an injective resolution of M.

Lemma 3.3.5. *A chain complex $C_\bullet$ of free modules is minimal if and only if $(\nu C_\bullet)^*$ is.*

Proof. This follows from the additivity of ν and $(-)^*$. Namely, if $C_\bullet \cong C'_\bullet \oplus B_\bullet$ for a homological d-ball $B = (\cdots \to 0 \to F(z) \xrightarrow{\cong} F(z) \to 0 \to \cdots)$, then we obtain the cochain complex $(\nu C_\bullet)^* \cong (\nu C'_\bullet)^* \oplus (\nu B_\bullet)^*$ with the cohomological $d-1$-ball $(\nu B_\bullet)^* = (\cdots \to 0 \to F(-z) \xrightarrow{\cong} F(-z) \to 0 \to \cdots)$. □

3.4 Pulling back modules from the colimit

From now on, we consider $\mathbf{Z}^2$-persistence modules only. In this and the following section, we explain how we actually compute $H^{d+2}(N^\bullet(K_*))$. In principle, this could be done by a procedure analogous to the one described in Section 2.4. Namely, the dashed sequence in the commutative diagram

$$\begin{tikzcd}
N^d(K_*) \arrow[d, "p^d"'] \arrow[dr, "\delta^{d+1}"] & & & \\
Z^{d+1}(N^\bullet(K_*)) \arrow[r, hook, dashed, "i^{d+1}"'] & N^{d+1}(K_*) \arrow[r, dashed, "p^{d+1}"] \arrow[dr, "\delta^{d+2}"'] & Z^{d+2}(N^\bullet(K_*)) \arrow[r, dashed] \arrow[d, hook, "i^{d+2}"] & H^{d+2}(N^\bullet(K_*)) \\
& & N^{d+2}(K_*) \arrow[dr, "\delta^{d+3}"] & \\
& & & N^{d+3}(K_*)
\end{tikzcd} \tag{3.8}$$

is a free resolution of $H^{d+2}(N^\bullet(K_*))$. We can obtain matrices I^{d+1}, P^{d+1} representing the morphisms i^{d+1}, p^{d+1} of this resolution with the method described in Section 2.4. However, this would involve the coboundary maps δ^{d+2} and δ^{d+3}, which is not feasible for clique complexes. Instead, we propose a method that computes a free resolution of $H^{d+2}(N^\bullet(K_*))$ from δ^{d+1} only.

At the core of this approach lies the observation that the kernel of a map $f\colon M \to N$ of free persistence modules is determined by the kernel of the morphism $\operatorname{colim} f$ of vector spaces, such that $\ker f$ can be reconstructed by “pulling back” $\operatorname{colim} f$ along the canonical morphism to the colimit. The following makes this precise.

Recall the Functors $\operatorname{Colim} = \Delta \operatorname{colim}$ and $\operatorname{Lim} = \Delta \lim\colon \mathrm{Vec}^{\mathbf{Z}^n} \to \mathrm{Vec}^{\mathbf{Z}^n}$. For a persistence module M, let $\operatorname{Colim} M = \Delta \operatorname{colim} M$, and let $\eta_M\colon M \to \operatorname{Colim} M$ be the canonical morphism. Note that η_M is induced by the unit of the adjunction $\operatorname{colim} \dashv \Delta$.

Definition 3.4.1. For a module $M \in \mathrm{Vec}^{\mathbf{Z}^2}$ and a vector space $V \subseteq \operatorname{colim} M$, let $[V]_M := \eta_M^{-1}(\Delta V) \subseteq M$.

The submodule $[V]_M$ is the unique largest submodule of M whose colimit is V:

Lemma 3.4.2. *For $M \in \mathrm{Vec}^{\mathbf{Z}^2}$ and $V \subseteq \operatorname{colim} M$, we have*

$$[V]_M = \sum\{N \subseteq M \mid \operatorname{colim} N \subseteq V\}.$$

Proof. Let $\mathcal{U} := \{N \subseteq M \mid \operatorname{colim} N \subseteq V\}$ If $x \in [V]_M$, then $\eta_M(x) \in V$. Therefore, the submodule $\langle x\rangle_M$ of M spanned by x satisfies $\operatorname{colim}\langle x\rangle_M = \langle \eta_M(x)\rangle_V \subseteq V$, so $\langle x\rangle_M \in \mathcal{U}$ and thus $x \in \sum\mathcal{U}$. This shows $[V]_M \subseteq \sum\mathcal{U}$. Conversely, let $x \in N \subseteq M$ such that $\operatorname{colim} N = \eta_M(N) \subseteq V$. Then $\eta_M(x) \in V$, so $x \in \eta_M^{-1}(V)$. Because V is closed under addition, this shows $\sum\mathcal{U} \subseteq [V]_M$. □

A persistence module is called *torsion free* if all its structure maps are injective. A free module is torsion free.

Theorem F. *If $f\colon M \to N$ is a morphism and N is torsion free, then* $\ker f = [\operatorname{colim}\ker f]_M$.

Proof. If N is torsion free, then $\eta_N\colon N \to \operatorname{Colim} N$ is injective. Every submodule $L \subseteq M$ satisfies $L \subseteq [\operatorname{colim} L]_M$, so in particular $\ker f \subseteq [\operatorname{colim}\ker f]_M$. It remains to show the other inclusion $[\operatorname{colim}\ker f]_M \subseteq \ker f$. Consider the commutative diagram

$$\begin{tikzcd}
{[\operatorname{colim}\ker f]_M} \arrow[drr, hook, bend left, "j"] \arrow[ddr, "\eta_{[\operatorname{colim}\ker f]_M}"'] & & & \\
& \ker f \arrow[ul, hook'] \arrow[r, hook, "i"'] \arrow[d, "\eta_{\ker f}"] & M \arrow[r, "f"] \arrow[d, "\eta_M"] & N \arrow[d, hook, "\eta_N"] \\
& \operatorname{Colim}\ker f \arrow[r, hook, "\operatorname{Colim} i"'] & \operatorname{Colim} M \arrow[r, "\operatorname{Colim} f"'] & \operatorname{Colim} N.
\end{tikzcd}$$

The functor Colim is a directed colimit and thus exact. Therefore, $\operatorname{Colim}\ker f = \ker\operatorname{Colim} f$. This implies

$$\eta_N f j = (\operatorname{Colim} f)(\operatorname{Colim} i)\eta_{[\operatorname{colim}\ker f]_M} = 0.$$

Since η_N is injective, we obtain $fj = 0$, so the injective morphism j factors uniquely through $\ker f$. This proves the claim. □

The lemma shows that $\ker f$ is the largest submodule of M whose colimit is $\operatorname{colim}\ker f$. In particular, one can reconstruct $\ker f$ from $\operatorname{colim}\ker f$.

Corollary 3.4.3. *If M is free of finite rank and $V \subseteq \operatorname{colim} M$, then $[V]_M$ is free.*

Proof. Consider the canonical projection $\tilde{p}\colon \operatorname{colim} M \to \operatorname{colim} M/V$. Let $z_0 \leq z$ for all $z \in \operatorname{rk} M$, and let N be the persistence module with $N_z = \begin{cases}\operatorname{colim} M/V & \text{if } z_0 \leq z, \\ 0 & \text{otherwise}\end{cases}$ and $N_{z',z} = \mathrm{id}$ for all $z_0 \leq z \leq z'$. Then N is free of graded rank $\{z_0 \mid z \in \operatorname{rk} M\}$ and satisfies $\operatorname{colim} N = \operatorname{colim} M/V$. The morphism $p := (\Delta\tilde{p})\eta_M\colon M \to N$ satisfies $\operatorname{colim} p = \tilde{p}$. From Theorem F, we get that $[V]_M = [\ker\tilde{p}]_M = \ker p$, which is free because it is a kernel of free modules in $\mathrm{Vec}^{\mathbf{Z}^2}$; see Corollary 2.3.20. □

In more than two parameters, Corollary 3.4.3 may fail; that is, the pullback $[V]_M$ of $V \subseteq \operatorname{colim} F$ may not be free even if M is. If $[V]_M$ is not free, it has a minimal generating system of larger cardinality than $\dim V$.

Recall the lexicographic and colexicographic ordering $\preceq_{\mathrm{lex}}$ and $\preceq_{\mathrm{colex}}$ on $\mathbf{Z}^2$ from Definition 2.4.2. Fix a tuple $\vec{z} \in (\mathbf{Z}^2)^m$ and let $z_i = (x_i, y_i)$ for all i.

Definition 3.4.4. With respect to $\vec{z}$, the *lex pivot* of a non-zero vector $b \in k^m$, denoted by $\text{l-piv}_{\vec{z}}(b)$, is the largest index in $\{i \,|\, b_i \neq 0\}$ for which z_i attains its $\preceq_{\text{lex}}$-maximal value. The *colex pivot*, denoted by $\text{c-piv}_{\vec{z}}(b)$, is defined analogously. For $0 \in k^m$, we let $\text{l-piv}_{\vec{z}}(0) = \text{c-piv}_{\vec{z}}(0) = 0$. A $m \times n$-matrix is *bireduced* with respect to $\vec{z}$ if all its non-zero columns have distinct lex pivots and distinct colex pivots with respect to $\vec{z}$.

Notation 3.4.5. For a $m \times n$-matrix B such that $B_j \neq 0$ for all j, let $[B]_{\vec{z}}$ denote the graded matrix with entries $\mathsf{u}([B]_{\vec{z}}) = B$, row grades $\text{rg}^{[B]_{\vec{z}}} = \vec{z}$ and column grades $\text{cg}_j^{[B]_{\vec{z}}} = \bigvee_{B_{ij}\neq 0} z_i$.

The graded matrix $[B]_{\vec{z}}$ is valid, and any valid graded matrix $\tilde{B}$ with $\mathsf{u}(\tilde{B}) = B$ and $\text{rg}^{\tilde{B}} = \vec{z}$ satisfies $\text{cg}_j^{\tilde{B}} \geq \text{cg}_j^{[B]_{\vec{z}}}$ for all j. Recall that for any z, $z' \in \mathbf{Z}^2$, we have $z \leq z'$ if and only if $z \preceq_{\text{lex}} z'$ and $z \preceq_{\text{colex}} z'$. This implies that $\text{cg}_j^{[B]_{\vec{z}}}$ is determined only by the lex and colex pivot of B_j:

$$\text{cg}_j^{[B]_{\vec{z}}} = \bigvee_{B_{ij}\neq 0} z_i = (x_{\text{l-piv}\, B_j}, y_{\text{c-piv}\, B_j}). \tag{3.9}$$

Let $\vec{z} = (z_i)_i \in (\mathbf{Z}^2)^n$ be a tuple of grades, let $M = \bigoplus_{i=1}^m F(z_i)$ be a free module, let $(e_i)_i$ denote the standard basis of M, let $V \subseteq \operatorname{colim} M$ be a vector subspace, and let B be a $m \times n$-matrix that represents a basis of V with respect to the basis $(\eta_M(e_i))_i$ of $\operatorname{colim} M$.

Lemma 3.4.6. *If B is bireduced, then $[B]_{\vec{z}}$ represents a basis of $[V]_M$.*

Proof. The graded matrix $[B]_{\vec{z}}$ represents a basis of a free submodule $N \subseteq M$ with $\eta_M(N) = V$. We have to show that $N = [V]_M$. Since B is a basis of V, we have $N \subseteq [V]_M$.

For the converse inclusion, let $v \in [V]_M$. We identify v with the unique valid graded column vector that represents v with respect to the basis $(e_i)_i$ of M and $\bar{v} := \eta_M(v) \in V$ with the ungraded column vector that represents $\bar{v}$ with respect to the basis $(\eta_M(e_i))_i$ of $\operatorname{colim} M$. Because B is a basis of V, there exists a unique column vector $\bar{w}$ such that $\bar{v} = B\bar{w}$. Let w be the graded column vector with $\mathsf{u}(w) = \bar{w}$, row grades $\text{rg}^w = \text{cg}^{[B]_{\vec{z}}}$ and column grade $\text{cg}^w = g(v)$. Then $v = [B]_{\vec{z}} w$. To show that $v \in N$, we have to show that w is valid. Because all columns B have pairwise distinct lex and colex pivots, we have

$$\begin{aligned} \text{l-piv}_{\vec{z}}\, v = \text{l-piv}_{\vec{z}}\, B\bar{w} &= \max\{\text{l-piv}_{\vec{z}}\, B_j \mid w_j \neq 0\}, \\ \text{c-piv}_{\vec{z}}\, v = \text{c-piv}_{\vec{z}}\, B\bar{w} &= \max\{\text{c-piv}_{\vec{z}}\, B_j \mid w_j \neq 0\}. \end{aligned}$$

Therefore, if j is an index such that $w_j \neq 0$, then $\text{l-piv}_{\vec{z}}\, \bar{v} \geq \text{l-piv}_{\vec{z}}\, B_j$ and $\text{c-piv}_{\vec{z}}\, \bar{v} \geq \text{c-piv}_{\vec{z}}\, B_j$. Because v is a valid graded column vector, we have

$$\text{cg}^w = \text{cg}^v \geq \bigvee_{v_i \neq 0} z_i \overset{(3.9)}{=} (x_{\text{l-piv}\, v}, y_{\text{c-piv}\, v}) \geq (x_{\text{l-piv}\, B_j}, y_{\text{c-piv}\, B_j}) \overset{(3.9)}{=} \text{cg}_j^{[B]_{\vec{z}}} = \text{rg}_j^w$$

for every j with $w_j \neq 0$, so w is valid. □

Theorem 3.4.7. *Let M be a free $\mathbf{Z}^2$-persistence module of finite rank with a fixed basis, let $V \subseteq \operatorname{colim} M$ be a subspace, and let B be a matrix representing a generating set of V. Then Algorithm 10 calculates a graded matrix representing a basis of $[V]_M$.*

Proof. In each iteration of the first for-loop, the colex-pivot index of one column decreases, so the loop terminates. When it does, all columns have distinct colex-pivots. Line (a) ensures that no column with a larger lex-pivot is added to a column with a smaller one. Therefore, the lex-pivots of the nonzero columns of B do not increase during the first for-loop. Analogously, during each iteration of the second for-loop, the lex-pivot of a column decreases. When the loop terminates, all nonzero columns have distinct lex-pivots. Line (c) ensures that no column

Algorithm 10: Computes a basis of $[V]_M$, where M is free of rank $\vec{z}$, and $V \subseteq \operatorname{colim} M$.

Input: An $m \times n$-matrix B representing a generating set of V, $\vec{z} = (z_i)_i \in (\mathbf{Z}^2)^m$.
Output: A graded $m \times n$-matrix whose nonzero columns represent a basis of $[V]_M$.

```
    function Bireduce($B$):
        $p \leftarrow 0 \in \mathbf{N}^n$
        for $j' = 1, \ldots, n$ do
            $j \leftarrow j'$
            while $i \leftarrow$ c-piv$_{\vec{z}}(B_j) \neq 0$ do
                if $p_i = 0$ then $p_i = j$; break
(a)             if l-piv$_{\vec{z}}(B_j) <$ l-piv$_{\vec{z}}(B_{p_i})$ then swap $p_i$ and $j$
                $B_j \leftarrow B_j - B_{ij}/B_{ip_i} B_{p_i}$
(b)     $p \leftarrow 0 \in \mathbf{N}^n$
        for $j' = 1, \ldots, n$ do
            $j \leftarrow j'$
            while $i \leftarrow$ l-piv$_{\vec{z}}(B_j) \neq 0$ do
                if $p_i = 0$ then $p_i = j$; break
(c)             if c-piv$_{\vec{z}}(B_j) <$ c-piv$_{\vec{z}}(B_{p_i})$ then swap $p_i$ and $j$
                $B_j \leftarrow B_j - B_{ij}/B_{ip_i} B_{p_i}$
        return $[B]_{\vec{z}}$
```

with a larger colex-pivot is added to a column with a smaller one. Since all nonzero columns have distinct colex-pivots after the first for-loop, the colex-pivots of the columns do not change during the second for-loop. Therefore, when the algorithm terminates, all nonzero columns of B have pairwise distinct lex- and colex-pivots, so B is bireduced. The statement then follows from Lemma 3.4.6. □

Remark. Let $B \in k^{l \times m}$ and $\vec{z} \in (\mathbf{Z}^2)^l$ as in Notation 3.4.5. The procedure `Bireduce()` (Algorithm 10) uses the following particularity of $\mathbf{Z}^2$: if two columns B_j and B_k have the same lex- or colex-pivot w.r.t. $\vec{z}$, then the respective columns of $[B]_{\vec{z}}$ have comparable grades. This need not be true for $\vec{z} \in (\mathbf{Z}^n)^l$ for $n > 2$.

3.5 Computing a free resolution of $H^\bullet(N^\bullet(K_*))$

In this section, we explain how to use `Bireduce()` (Algorithm 10) to compute a minimal free resolution of $H^{d+2}(N^\bullet)$, where $N^\bullet$ is an eventually acyclic cochain complex of free two-parameter persistence modules. For example, $N^\bullet = N^\bullet(K_*)$ if K_* is eventually acyclic. The dashed sequence in the commutative diagram

$$\begin{tikzcd}
 & N^d \arrow[d, "p^d"'] \arrow[dr, "\delta^{d+1}"] & & & \\
0 \arrow[r, dashed] & Z^{d+1}(N^\bullet) \arrow[r, dashed, "i^{d+1}"'] & N^{d+1} \arrow[r, dashed, "p^{d+1}"] \arrow[dr, "\delta^{d+2}"'] & Z^{d+2}(N^\bullet) \arrow[r, dashed] \arrow[d, hook, "i^{d+2}"] & H^{d+2}(N^\bullet) \\
 & & & N^{d+2} &
\end{tikzcd} \tag{3.10}$$

is a free resolution of $H^{d+2}(N^\bullet)$. Chose a basis of $N^\bullet$, and let $D^\bullet$ be the graded matrix representing $\delta^\bullet$. Because $N^\bullet$ is eventually acyclic, the morphism $\operatorname{colim} p^d \colon \operatorname{colim} N^d \to \operatorname{colim} Z^{d+1}(N^\bullet)$ is surjective. In other words, the ungraded matrix underlying D^{d+1} represents a generating system of $\operatorname{colim} Z^{d+1}(N^\bullet)$. When applied to D^{d+1}, Algorithm 10 thus computes a graded matrix

I^{d+1} that represents a basis of the free submodule

$$[\operatorname{colim} Z^{d+1}(N^\bullet)]_{N^{d+1}} \subseteq N^{d+1}.$$

According to Theorem F, we have

$$[\operatorname{colim} Z^{d+1}(N^\bullet)]_{N^{d+1}} = Z^{d+1}(N^\bullet).$$

Therefore, `Bireduce`() (Algorithm 10) allows us to compute a basis

$$I^{d+1} := \texttt{Bireduce}(D^{d+1})$$

of $Z^{d+1}(N^\bullet)$ from D^{d+1}; i.e., a matrix representing the morphism i^{d+1}. It remains to compute a graded matrix P^{d+1} representing the morphism p^{d+1}. This can be done using the following consequence of Corollary 3.2.11:

Corollary 3.5.1. *If $0 \to F_n \xrightarrow{f_n} \cdots \xrightarrow{f_1} F_0 \to M$ is a free resolution of a finitely supported $\mathbf{Z}^n$-persistence module M, then $(\nu F_0)^* = \ker(\nu f_2)^*$.*

Proof. Theorem 3.2.8 If

$$0 \longrightarrow F_n \xrightarrow{f_n} \cdots \xrightarrow{f_2} F_1 \xrightarrow{f_1} F_0 \longrightarrow M$$

is a free resolution of M, then by Corollary 3.2.11,

$$M \longrightarrow \nu F_n \xrightarrow{\nu f_n} \cdots \xrightarrow{\nu f_2} \nu F_1 \xrightarrow{\nu f_1} \nu F_0 \longrightarrow 0$$

is an injective resolution of M and the dual sequence

$$0 \longrightarrow (\nu F_0)^* \xrightarrow{(\nu f_1)^*} (\nu F_1)^* \xrightarrow{(\nu f_2)^*} \cdots \xrightarrow{(\nu f_n)^*} (\nu F_n)^* \longrightarrow M^*$$

is a free resolution of M^*. In particular, $(\nu F_0)^* = \ker((\nu f_2)^*)$. □

This shows that in the situation of (3.10), we have

$$(\nu Z^{d+2}(N^\bullet))^* = \ker(\nu i^{d+1})^*,$$

where $(\nu i^{d+1})^*$ is a morphism of free modules. The morphism $(\nu p^{d+1})^*$ is the inclusion

$$(\nu p^{d+1})^* \colon \ker(\nu i^{d+1})^* \hookrightarrow (\nu N^{d+1})^*.$$

If i^{d+1} is represented by the graded matrix I^{d+1}, then according to Lemma 3.3.3, the morphism $(\nu i^{d+1})^*$ is represented by the graded matrix $(I^{d+1})^\top$, and $(\nu p^{d+1})^*$ is represented by some matrix $(P^{d+1})^\top$, such that P^{d+1} represents p^{d+1}. Because $(\nu p^{d+1})^*$ is the kernel inclusion of a morphism of free modules, the matrix $(P^{d+1})^\top$ can be computed by using `Ker`() (Algorithm 6).

Theorem 3.5.2. *Let $N^\bullet$ be an eventually acyclic cochain complex of free modules. Then Algorithm 11 computes free resolutions of $H^d(N^\bullet)$.*

Remark 3.5.3 (Clearing). In general, D^{d+1} is not injective. As in one-parameter persistent cohomology, the first loop in Algorithm 10 would spend some time on reducing the columns of D^{d+1} that are eventually reduced to zero if Algorithm 10 was applied to D^{d+1}. The clearing scheme in Algorithm 11, line (a) avoids this.

Proof of Theorem 3.5.2. For legibility, we omit the dimension superscripts in the following; every matrix I and P and every morphism i and p has a superscript $d+1$.

Because $N^\bullet$ is eventually acyclic, we have that

$$\operatorname{colim} B^d(N^\bullet) = B^d(\operatorname{colim} N^\bullet) = Z^d(\operatorname{colim} N^\bullet) = \operatorname{colim} Z^d(N^\bullet)$$

for all d. The underlying matrix $\mathtt{u}(D^{d+1})$ of D^{d+1} thus represents a generating system of $\operatorname{colim} Z^d(C_\bullet)$.

Claim. After the clearing step in line (a), we still have that $\mathtt{u}(D^{d+1})$ represents a generating system of $\operatorname{colim} Z^d(C_\bullet)$.

Proof of claim. Let $\hat{I}$ denote the matrix $\hat{I} \coloneqq \mathtt{Bireduce}(D^d)$ obtained as I in the previous iteration. Because $D^{d+1}D^d = 0$ and $\mathtt{Bireduce}(D^d)$ performs only column operations, we have $D^{d+1}\hat{I} = 0$. Then every column $\hat{I}_j$ with pivot i represents a linear relation $[D^{d+1}]_i = \sum_{i'<i} \hat{I}_{i'j}[D^{d+1}]_{i'}$ of the columns of D^{d+1}. Setting the columns $[D^{d+1}]_i$ to zero for every pivot $i \in q$ of $\hat{I}_j$ thus does not change the image of $\mathtt{u}(D^{d+1})$.

It follows from Theorem 3.4.7 and Corollary 3.5.1 that the matrices

$$\begin{aligned} I &\coloneqq \mathtt{Bireduce}(D^{d+1}) \\ P &\coloneqq (\mathtt{Ker}(I^\top))^\top \end{aligned}$$

represent the free resolution

$$0 \to Z^{d+1}(N^\bullet) \xrightarrow{i} N^{d+1} \xrightarrow{p} Z^{d+2}(N^\bullet) \to H^{d+2}(N^\bullet)$$

of $H^{d+2}(N^\bullet)$. By Proposition 2.3.23, replacing I by the minimal (in the sense of Definition 2.3.21) graded matrix $I' \coloneqq \mathtt{Minimize}(I)$ fits into a free resolution

$$0 \to Z'^{d+1}(N^\bullet) \xrightarrow{I'} C'^{d+1} \xrightarrow{P'} Z^{d+2}(N^\bullet) \to H^{d+2}(N^\bullet)$$

of $H^{d+2}(N^\bullet)$ for some matrix P'. We could compute P' by

$$P' \coloneqq (\mathtt{Ker}((I')^\top))^\top$$

However, the thus computed matrix P' need not be minimal. To address this, recall that $\mathtt{MGSWithKer}()$ (Algorithm 8) can be used to compute matrices

$$((\tilde{I})^\top, (\tilde{P})^\top) \coloneqq \mathtt{MGSWithKer}((I')^\top),$$

where $\operatorname{im} \tilde{I}^\top = \operatorname{im}(I)^\top$, and $\tilde{P}^\top$ is a minimal matrix (in the sense of Definition 2.3.21) represents a basis of $\ker \tilde{I}^\top$. These fit into a free resolution

$$0 \to \tilde{Z}^{d+1} \xrightarrow{\tilde{I}} \tilde{C}^{d+1} \xrightarrow{\tilde{P}} \tilde{Z}^{d+2} \to H^{d+1}(N^\bullet)^*$$

of $H^{d+1}(N^\bullet)^*$ for some free modules $\tilde{Z}^{d+1}$, $\tilde{C}^{d+1}$ and $\tilde{Z}^{d+2}$. The matrix $\tilde{P}$ is minimal because $\tilde{P}^\top$ is, and $\tilde{I}$ is minimal because already I' was minimal. Hence, this is a minimal free resolution of $H^{d+1}(N^\bullet)^*$. □

Theorem 3.5.2 also shows:

Theorem C. *Let K_* be an eventually acyclic one-critically filtered simplicial complex. Then $H^{d+2}(N^\bullet(K_*))$ can be computed from the coboundary map $\delta^{d+1}\colon N^d(K_*) \to N^{d+1}(K_*)$.*

Algorithm 11: Cohomology algorithm. Computes a minimal free resolution of $H^\bullet(N^\bullet)$ for a cochain complex $N^\bullet$ of free $\mathbf{Z}^2$-modules, using clearing.

Input: Graded matrices $D^\bullet$ representing $N^\bullet$.
Output: Pairs of graded matrices representing a free resolution of $H^d(N^\bullet)$ for $d = 0, 1, \ldots$.

```
    $q \leftarrow \emptyset$                                        ▷ pivots for clearing
    for $d = 0, 1, \ldots$ do
(a)     for $j \in q$ do $[D^{d+1}]_j \leftarrow 0$                 ▷ clearing
(b)     $I \leftarrow$ Bireduce($D^{d+1}$)                          ▷ see Algorithm 10
        $q \leftarrow \{\operatorname{piv} I_j \mid I_j \neq 0\}$   ▷ pivots for next iteration
(c)     $I', r, c \leftarrow$ Minimize($I$)                         ▷ see Algorithm 3
        $\tilde{I}^\top, \tilde{P}^\top \leftarrow$ MGSWithKer($(I')^\top$)   ▷ see Algorithm 8
        yield $\tilde{I}, \tilde{P}$                                ▷ min. free res. of $H^{d+2}(N^\bullet)$
```

Remark. Recall that a matrix is minimal if and only if its graded transpose is. Since we need $(I')^\top$ anyway, it makes sense to replace line (c) by

$$((I')^\top, r, c) \leftarrow \texttt{Minimize}(I^\top).$$

In other words, we can either first minimize I and then take the graded transpose, or first take the transpose and then minimize. Experiments have shown that the version in line (c) is slightly faster.

Remark (Base change). It is possible to change the basis of N^d before or after line (b). In this case, the algorithm computes a different but equivalent free resolution. In particular, one may choose a basis of N^d that makes later computations less expensive. We will elaborate on this idea in Section 5.1.5.

3.6 Eventual acyclicity of K_* is necessary

Recall that in one-parameter persistence (where $N^\bullet(K_*) = C^\bullet(K, K_*)$), the absolute and relative cohomology $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ determine each other uniquely even if K_* is not eventually acyclic, see Proposition 2.2.10 and Corollary 2.2.11. In two and more parameters, this is not true anymore, as we will see in this section. Specifically, we show:

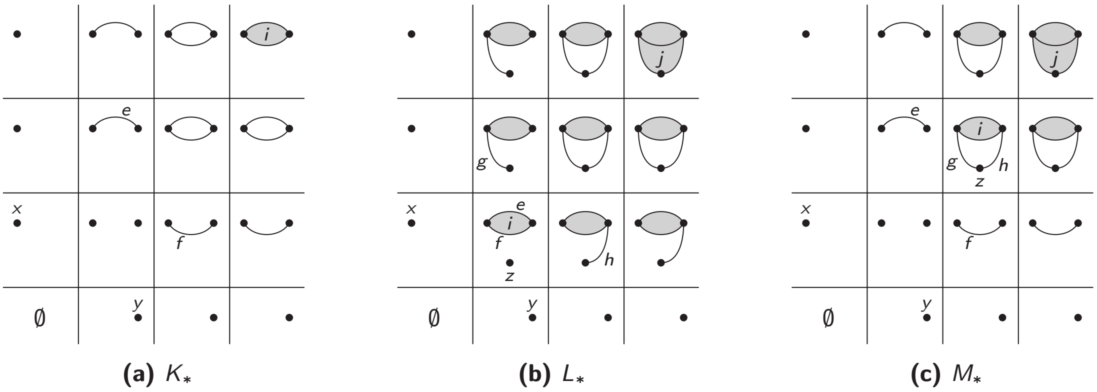
Theorem B(a). *There exist one-critically two-parameter filtered simplicial complexes K_*, L_* and M_* that are not eventually acyclic, such that*

(i) $H^\bullet(K_*) \ncong H^\bullet(L_*)$, but $H^\bullet(N^\bullet(K_*)) \cong H^\bullet(N^\bullet(L_*))$, and
(ii) $H^\bullet(K_*) \cong H^\bullet(M_*)$, but $H^\bullet(N^\bullet(K_*)) \ncong H^\bullet(N^\bullet(M_*))$.

Proof. Let K_*, L_* and M_* be the one-critically filtered simplicial complexes in Figure 3.4. These are not eventually acyclic. We compute $H_\bullet(-)$ and $H^\bullet(N^\bullet(-))$ for each of these complexes. For simplicity, we compute non-reduced (co)homology; an example using reduced homology can be obtained by adding one additional vertex.

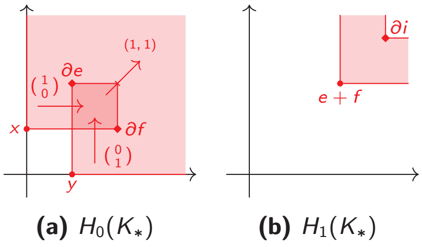
The homology and cohomology modules are shown in Figures 3.5 to 3.10. One sees that $H_d(K_*) \cong H_d(M_*)$ (and thus also $H^d(K_*) \cong H^d(M_*)$) and $H^d(N^\bullet(K_*)) \cong H^d(N^\bullet(L_*))$ for all d. The module $H_0(K_*)$ is indecomposable because it contains an indecomposable quiver representation, while $H_0(L_*)$ is a direct sum of nonzero submodules, so $H_0(K_*) \ncong H_0(L_*)$ (and thus also $H^0(K_*) \ncong H^0(L_*)$). Similarly, $H^2(N^\bullet(K_*))$ is indecomposable, while $H^2(N^\bullet(M_*))$ is a direct sum of nonzero submodules, so $H^2(N^\bullet(K_*)) \ncong H^2(N^\bullet(M_*))$. This establishes the desired example. □

This example also shows that $H^\bullet(K_*)$ and $H^\bullet(N^\bullet(K_*))$ need not determine each other uniquely even if $\operatorname{colim} K_*$ is acyclic (but $\operatorname{colim}_{\{1\}} K_*$ or $\operatorname{colim}_{\{2\}} K_*$ is not).



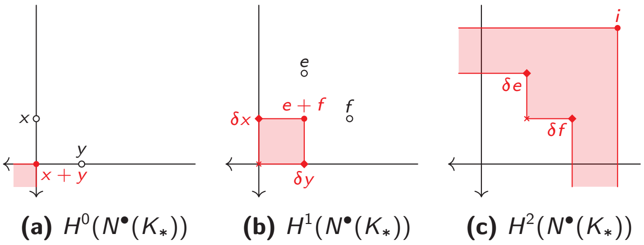
(a) K_* **(b)** L_* **(c)** M_*

Figure 3.4: The filtered simplicial complexes used in Theorem B(a). The filtration goes left to right and bottom to top.



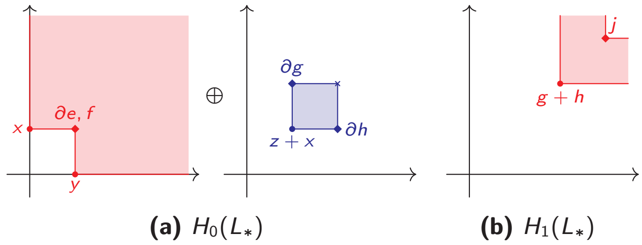
(a) $H_0(K_*)$ **(b)** $H_1(K_*)$

Figure 3.5: Homology of the complex K_* from Figure 3.4a. The module $H_0(K_*)$ is generated by x, y. These are subject to the relations $\partial e = \partial f = x + y$. Hence, the generators x and y are identified when e or y are present in the filtration. Analogously for $H_1(K_*)$. The module $H_0(K_*)$ is indecomposable, because it contains an indecomposable quiver representation.



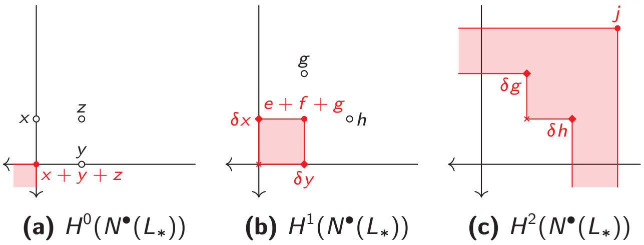
(a) $H^0(N^\bullet(K_*))$ **(b)** $H^1(N^\bullet(K_*))$ **(c)** $H^2(N^\bullet(K_*))$

Figure 3.6: Cohomology of $N^\bullet(K_*)$ for the complex K_* from Figure 3.4a. The coordinate axes are drawn pointing downward. Therefore, the generator labeled, e.g., "i" really means the basis element of $N^\bullet(K_*)$ of grade $-g(i)$ corresponding to the cell i. The symbols $\circ$ denote the negated grades of the respective cells of K_*. That is, the label "$x + y$" is placed at $-g(x + y) = -g(x) \vee -g(y)$.



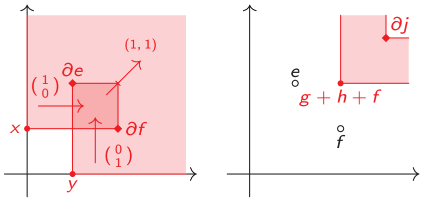
(a) $H_0(L_*)$ **(b)** $H_1(L_*)$

Figure 3.7: Homology of the complex L_* from Figure 3.4b. The generators x, y of $H_0(K_*)$ are identified by $\partial e = \partial f = x + f$ at $g(x) \vee g(y)$, and $z + x$ generates a direct summand "killed" by ∂g and ∂h.



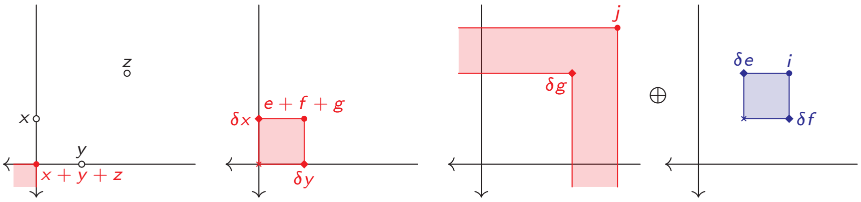
(a) $H^0(N^\bullet(L_*))$ **(b)** $H^1(N^\bullet(L_*))$ **(c)** $H^2(N^\bullet(L_*))$

Figure 3.8: Cohomology of $N^\bullet(L_*)$ for the complex L_* from Figure 3.4b. Note that $e + f + g \equiv e + h + f \pmod{\delta z}$.



(a) $H_0(M_*)$ **(b)** $H_1(M_*)$

Figure 3.9: Homology of the complex M_* from Figure 3.4c. The generator $g + h + f$ of $H_1(M_*)$ is equivalent (mod ∂i) to $g + h + e$. The module $H_0(M_*)$ is indecomposable.



(a) $H^0(N^\bullet(M_*))$ **(b)** $H^1(N^\bullet(M_*))$ **(c)** $H^2(N^\bullet(M_*))$

Figure 3.10: Cohomology of $N^\bullet(M_*)$ for the complex M_* from Figure 3.4c.

3.7 Making K_* eventually acyclic

For a given filtered complex K_* and $z \in \mathbf{Z}^n$, we consider two ways to construct an eventually acyclic chain complex $\tilde{C}_\bullet$ such that $(\tilde{C}_\bullet)_w = C_\bullet(K_w)$ for all $w \leq z$. Inspired by the first way presented in Section 3.7.1, we refer to this as *coning off* K_*, although we do not build an actual cone in all cases. The motivating situation for this is that we choose $z > g(\sigma)$ for all $\sigma \in K_*$, so that everying "interesting" in K_* happens below z.

3.7.1 Simplicially coning off K_*

Assume that $z = (z_1, \dots, z_n) > g(\sigma)$ for all $\sigma \in K_*$, and let $1 \leq k \leq n$. For $w = (w_1, \dots, w_n) \in \mathbf{Z}^n$ let $w|_z^k \in \mathbf{Z}^n$ be obtained by replacing the kth coordinate of w by z_k. We define the one-critically filtered simplicial complex $\tilde{K}_*$ with

$$K_*^{(d)} = K_*^{(0)} \cup \begin{cases} \{a\} & \text{if } d = 0, \\ \{\tilde{\sigma} \mid \sigma \in K_*^{(d-1)}\} & \text{if } d > 0, \end{cases}$$

for $\tilde{\sigma} := \sigma \cup \{a\}$, equipped with the grading $g(a) := \bigwedge_{\sigma \in K_*} g(\sigma)|_z^k$ and $g(\tilde{\sigma}) := g(\sigma)|_z^k$. If $w_k \geq z_k$, then $\tilde{K}_w$ is a cone with apex a and thus contractible. If $w_k < z_k$, then $\tilde{K}_w = K_w$. The same construction can be applied on the level of chains. For $w \in \mathbf{Z}^n$ let $F(z)|_z^k := F(w|_z^k)$. Then $\tilde{C}_\bullet := C_\bullet(\tilde{K}_*)$ is the chain complex with

$$\tilde{C}_d := C_d(K_*) \oplus C_{d-1}(K_*)|_z^k, \qquad \partial_d := \begin{pmatrix} \partial_d & (-1)^d \\ & \partial_{d-1} \end{pmatrix}.$$

Repeatedly applying this construction to $\tilde{C}_\bullet$ for all $1 \leq k \leq n$ gives an eventually acyclic chain complex $\tilde{C}_\bullet$ such that $(\tilde{C}_\bullet)_w = (C_\bullet)_w$ if $w < z$.

3.7.2 Coning off K_* using homology resolutions

Alternatively, if $C_\bullet$ is a chain complex of free modules, one can obtain a new complex $\hat{C}_\bullet$ by adjoining only as many new basis elements to $C_\bullet$ as necessary to turn $\hat{C}_\bullet$ eventually acyclic. Let $z \in \mathbf{Z}^n$ with $z > w$ for all $w \in \operatorname{rk} C_d$ for all d. For $k \leq n$, recall the exact functors

$\operatorname{colim}_{\{k\}} \colon \mathrm{Vec}^{\mathbf{Z}^n} \to \mathrm{Vec}^{\mathbf{Z}^{n-1}}$ from Definition 3.2.6. We also define the exact functor

$$\begin{aligned} E_k \colon \mathrm{Vec}^{\mathbf{Z}^{n-1}} &\longrightarrow \mathrm{Vec}^{\mathbf{Z}^n}, \\ M &\longmapsto (w_1, \dots, w_n) \mapsto \left\{ \begin{smallmatrix} M_{(w_1,\dots,w_{k-1},w_{k+1},\dots,w_n)} & \text{if } w_k \geq z_k, \\ 0 & \text{otherwise.} \end{smallmatrix} \right. \end{aligned}$$

A module $F \in \mathrm{Vec}^{\mathbf{Z}^{n-1}}$ is free if and only if $E_k F$ is. In this case,

$$E_k F((w_1, \dots, w_{k-1}, w_{k+1}, \dots, w_n)) = F((w_1, \dots, w_{k-1}, z_k, w_{k+1}, \dots, w_n)).$$

Two parameters We first describe the construction for the case $n = 2$. Let $z \in \mathbf{Z}^2$ such that $z > w$ for all $z \in \operatorname{rk} C_d$ for all d as above, and choose $k \in \{1, 2\}$. W.l.o.g., we assume that $k = 1$. Consider the chain complex $C'_\bullet \coloneqq \operatorname{colim}_{\{k\}} C_\bullet$ of free one-parameter persistence modules. For each d, choose a free resolution $F_{d,\bullet}$ of $H_d(C'_\bullet) \cong \operatorname{colim}_{\{k\}} H_d(C_\bullet)$ of length 1, and let $f_{d,0} \colon F_{d,0} \to H_d(C'_\bullet)$ be the augmentation morphism. Consider the diagram

$$\begin{tikzcd} 0 \arrow[r] & F_{d,1} \arrow[r, "f_{d,1}"] \arrow[d, dashed, "\phi_{d,1}"] & F_{d,0} \arrow[r, "f_{d,0}"] \arrow[d, dashed, "\phi_{d,0}"] & H_d(C'_\bullet) \arrow[r] \arrow[d, equal] & 0 \\ & C'_{d+1} \arrow[r, "\partial_{d+1}"'] & Z_d(C'_\bullet) \arrow[r, "q_d"'] & H_d(C'_\bullet) \arrow[r] & 0 \end{tikzcd} \tag{3.11}$$

of one-parameter persistence modules. Each of the two rows is exact and thus forms (in the case of the lower row, the beginning of a) free resolution of $H_d(C_\bullet)$. We construct the morphisms $\phi_{d,0}$ and $\phi_{d,1}$ that make the diagram commute. We will use these to construct a chain complex $\tilde{C}_\bullet$ with $(\tilde{C}_\bullet)_w = (C_\bullet)_w$ if $w_k < z_k$ and $(\tilde{C}_\bullet)_w = (\tilde{C}'_\bullet)_w$ otherwise.

Since q_d is surjective and $F_{d,0}$ free, there exists a morphism $\phi_{d,0}$ such that $q_d \phi_{d,0} = f_{d,0}$. This implies $q_d \phi_{d,0} f_{d,1} = 0$. Because the lower row is exact, ∂_{d+1} surjects onto $B_d(C'_\bullet) = \ker q_d$. Since $F_{d,1}$ is free, there exists a morphism $\phi_{d,1}$ such that $\partial_{d+1} \phi_{d,1} = \phi_{d,0} f_{d,1}$. It follows from commutativity of (3.11) than

$$\tilde{C}'_d \coloneqq C'_d \oplus F_{d-1,0} \oplus F_{d-2,1}, \qquad \tilde{\partial}_d \coloneqq \left(\begin{smallmatrix} \partial_d & \phi_{d-1,0} & -\phi_{d-2,1} \\ 0 & 0 & f_{d-2,1} \\ 0 & 0 & 0 \end{smallmatrix} \right) \colon \tilde{C}'_d \longrightarrow \tilde{C}'_{d-1}. \tag{3.12}$$

defines a chain complex.

Lemma 3.7.1. *For any number n of parameters, the morphism*

$$\alpha_{d,2} \coloneqq (\partial_{d+1}, \phi_{d,0}) \colon C'_{d+1} \oplus F_{d,0} \to Z_d(C'_\bullet). \tag{3.13}$$

is surjective.

Proof. To see this, note that for $c \in Z_d(C_\bullet)$, surjectivity of $f_{d,0}$ allows us to choose a $b \in F_{d,0}$ such that $f_{d,0}(b) = q_d(c)$. Then $q_d(c - \phi_{d,0}(b)) = 0$, so by exactness of the lower row, we obtain that $c = \partial_{d+1}(a) + \phi_{d,0}(b) = \alpha_{d,2}\left(\begin{smallmatrix} a \\ b \end{smallmatrix} \right)$ for some $a \in C'_{d+1}$. □

Lemma 3.7.2. *The complex $\tilde{C}'_\bullet$ from* (3.12) *is acyclic.*

Proof. Let $(a, b, c) \in \ker \tilde{\partial}_d$. Then $c = 0$, because $f_{d-2,1}$ is injective. Then $\partial_d(a) + \phi_{d-1,0}(b) = 0$, which gives

$$f_{d-1,0}(b) = q_{d-1} \partial_d(a) = 0.$$

By exactness of the upper row, we may choose $b' \in F_{d-1,1}$ such that $b = f_{d-1,1}(b')$. This implies $\partial_d(a + \phi_{d-1,1}(b')) = 0$. By surjectivity of $\alpha_{d,2}$, we may choose $(a', b'') \in C_{d+1} \oplus F_{d,0}$ such that

$$a + \phi_{d-1,1}(b') = \partial_{d+1}(a') + \phi_{d,0}(b'').$$

This shows that $(a, b, c) = \tilde{\partial}_{d+1}(a', b'', b')$, so $\tilde{C}'_\bullet$ is acyclic. □

Let $\tilde{C}_\bullet$ be the chain complex of free modules with

$$\begin{aligned}\tilde{C}_d &:= C_d \oplus E_k F_{d-1,0} \oplus E_k F_{d-2,1},\\ \tilde{\partial}_d &:= \begin{pmatrix} \partial_d & E_k\phi_{d-1,0} & -E_k\phi_{d-2,1} \\ 0 & 0 & E_k f_{d-2,1} \\ 0 & 0 & 0 \end{pmatrix} : \tilde{C}_d \longrightarrow \tilde{C}_{d-1}.\end{aligned}$$

Then $(\tilde{C}_\bullet)_w = (C_\bullet)_w$ if $w_k < z_k$ and $(\tilde{C}_\bullet)_w = (\tilde{C}'_\bullet)_w$ otherwise. In particular, $\operatorname{colim}_k \tilde{C}_\bullet = C'_\bullet$ is acyclic. If $F_{d,\bullet}$ is a minimal free resolution for each d, then $\tilde{C}_\bullet$ is the smallest chain complex with these properties.

Recall that $C'_\bullet$ is a chain complex of finite rank free one-parameter persistence modules, so $B_d(C'_\bullet)$ and $Z_d(C'_\bullet)$ are free and $F_{d,\bullet} = (0 \to B_d(C'_\bullet) \to Z_d(C'_\bullet))$ is a free resolution of $H_d(C'_\bullet)$. In this case, $\tilde{C}_\bullet$ is isomorphic to the cone construction described in Section 3.7.1. Note that since $H_d(C'_\bullet)$ is a one-parameter persistence module, one can choose a basis of $F'_\bullet$ that exhibits the barcode of $H_d(C'_\bullet)$ as described in Section 2.2, which can be computed efficiently. We give an algorithmic treatment of this approach in Section 5.1.4.

Applying the above construction to $\tilde{C}_\bullet$ again, now for $k = 2$, gives an eventually acyclic complex $\tilde{C}_\bullet$ $(\tilde{C}_\bullet)_w = C_\bullet(K_w)$ for all $w \leq z$.

More than two parameters The above construction can be generalized inductively to more than two parameters. Let $C_\bullet$ be a chain complex of free n-parameter persistence modules. As before, let $C'_\bullet := \operatorname{colim}_{\{k\}} C'_\bullet$

$$0 \to F_{d,n-1} \xrightarrow{f_{d,n-1}} F_{d,n-2} \to \cdots \to F_{d,0} \xrightarrow{f_{d,0}} H_d(C_\bullet) \to 0.$$

be a free resolution of $H_\bullet(C'_\bullet)$ of length $n-1$. For each d, we construct two sequences of morphisms $\phi_{d,0}, \phi_{d,1}, \ldots, \phi_{d,n-1}$ and $\alpha_{d+2,2}, \ldots, \alpha_{d+n-1,n-1}$ that make the diagram

$$\begin{tikzcd}
\cdots \arrow[r] & F_{d,3} \arrow[r, "f_{d,3}"] \arrow[d, dashed, "\phi_{d,3}"] & F_{d,2} \arrow[r, "f_{d,2}"] \arrow[d, dashed, "\phi_{d,2}"] \arrow[dr, phantom, "②"] & F_{d,1} \arrow[r, "f_{d,0}"] \arrow[d, dashed, "\phi_{d,1}"] \arrow[dr, phantom, "①"] & F_{d,0} \arrow[r, "f_{d,0}"] \arrow[d, dashed, "\phi_{d,0}"] \arrow[dr, phantom, "⓪"] & H_d(C'_\bullet) \arrow[r] \arrow[d, equal] & 0 \\
\cdots \arrow[r] & A_{d,3} \arrow[r, "\alpha_{d,3}"'] & A_{d,2} \arrow[r, "\alpha_{d,2}"'] & A_{d,1} \arrow[r, "\alpha_{d,1}"'] & A_{d,0} \arrow[r, "q_d"'] & H_d(C'_\bullet) \arrow[r] & 0.
\end{tikzcd} \tag{3.14}$$

commute and render the lower row exact, where

$$\begin{aligned}
A_{d,0} &= Z_d(C'_\bullet) & & \\
A_{d,1} &= C'_{d+1} & \alpha_{d,1} &= \partial_{d+1}, \\
A_{d,2} &= C'_{d+2} \oplus F_{d+1,0} & \alpha_{d,2} &= (\partial_{d+2}, \phi_{d+1,0}), \\
A_{d,i} &= A_{d+1,i-1} \oplus F_{d+1,i-2} & \alpha_{d,i} &= \begin{pmatrix} \alpha_{d+1,i-1} & (-1)^i\phi_{d+1,i-2} \\ 0 & f_{d+1,i-2} \end{pmatrix}
\end{aligned} \tag{3.15}$$

for $i > 1$. We will then construct a chain complex $\tilde{C}_\bullet$ such that $(\tilde{C}_\bullet)_w = (C_\bullet)_w$ if $w_k < z_k$ and $(\tilde{C}_\bullet)_w$ is acyclic otherwise.

Lemma 3.7.3. *There exists morphisms $\phi_{d,i}$, such that the diagram* (3.14) *commutes, and such that the morphisms $\alpha_{d,i}$ make $A_{d,\bullet}$ a free resolution of $H_d(C'_\bullet)$. that is exact at $A_{d,i}$ for all $i > 0$.*

Proof. We construct the morphisms $\phi_{d,i}$ inductively. As induction base, we note that q_d surjects onto $H_d(C')$ and $F_{d,0}$ is free, so there exists a morphism $\phi_{d,0}$ such that the square ⓪ commutes. This establishes the construction of $\phi_{d,0}$. We next construct the morphism $\phi_{d,1}$. Commutativity of ⓪ gives $q_d\phi_{d,0}f_{d,1} = 0$, so $\phi_{d,0}f_{d,1}$ factors through a unique morphism $F_{d,1} \to \ker q_d$. The lower row is exact at $A_{d,0} = Z_d(C'_\bullet)$. Therefore, the morphism $\alpha_{d,1}$ surjects onto $\operatorname{im}\alpha_{d,1} = \ker q_d$. The universal property of the free (and thus projective) module $F_{d,1}$ gives a morphism $\phi_{d,1}$ such that ① commutes. We next show that the $A_{d,\bullet}$ form an exact chain complex at $A_{d,1}$. We have

$$\alpha_{d,1}\alpha_{d,2} = \partial_{d+1}(\partial_{d+2}, \phi_{d+1,0}) = (0, \partial_{d+1}\phi_{d+1,0}) = 0,$$

because $\operatorname{im}\phi_{d+1,0} \subseteq A_{d+1,0} = Z_{d+1}(C'_\bullet)$. This shows that $A_{d,\bullet}$ forms a chain complex at $A_{d,1}$. Exactness at $A_{d,1}$ was already shown in Lemma 3.7.1. With the same argument as for $\phi_{d,1}$, we get the morphism $\phi_{d,2}$ that makes the square ② commute.

As induction hypothesis, we assume for $i \geq 2$ that

$$\alpha_{d+1,i-1}\phi_{d+1,i-1} = \phi_{d+1,i-2}f_{d+1,i-1} \tag{3.16}$$

$$\alpha_{d,i}\phi_{d,i} = \phi_{d,i-1}f_{d,i} \tag{3.17}$$

and that the chain complex $(A_{d+1,\bullet}, \alpha_{d+1,\bullet})$ is exact at $A_{d+1,i-1}$. For the induction step, we first show that $A_{d,\bullet}$ forms an exact complex at $A_{d,i}$. We have

$$\begin{aligned}\alpha_{d,i}\alpha_{d,i+1} &= \begin{pmatrix}\alpha_{d+1,i-1} & (-1)^i\phi_{d+1,i-1} \\ 0 & f_{d+1,i-1}\end{pmatrix}\begin{pmatrix}\alpha_{d+1,i} & (-1)^{i+1}\phi_{d+1,i} \\ 0 & f_{d+1,i}\end{pmatrix} \\ &= \begin{pmatrix}\alpha_{d+1,i-1}\alpha_{d+1,i} & \pm\alpha_{d+1,i-1}\phi_{d+1,i} \mp \phi_{d+1,i-1}f_{d+1,i} \\ & f_{d+1,i-1}f_{d+1,i}\end{pmatrix} \overset{(3.16)}{=} 0.\end{aligned}$$

To show acyclicity, let

$$\begin{pmatrix}a \\ b\end{pmatrix} \in \ker\alpha_{d,i} = \ker\begin{pmatrix}\alpha_{d+1,i-1} & (-1)^i\phi_{d+1,i-2} \\ & f_{d+1,i-2}\end{pmatrix} \subseteq A_{d+1,i-1} \oplus F_{d+1,i-2} = A_{d,i}. \tag{3.18}$$

Then $f_{d+1,i-2}(b) = 0$. By exactness of $F_{d+1,\bullet}$, there exists a $c \in F_{d+1,i-1}$ such that $b = f_{d+1,i-1}(c)$. Then

$$\begin{aligned}\alpha_{d+1,i-1}(a + (-1)^i\phi_{d+1,i-1}(c)) &\overset{(3.16)}{=} \alpha_{d+1,i-1}(a) + (-1)^i\phi_{d+1,i-2}f_{d+1,i-1}(c) \\ &\overset{(3.18)}{=} \alpha_{d+1,i-1}(a) + (-1)^i\phi_{d+1,i-2}(b) = 0.\end{aligned}$$

Because $A_{d+1,\bullet}$ is exact at $A_{d+1,i-1}$ by the induction hypothesis, we get an $m \in A_{d+1,i}$ such that $\alpha_{d+1,i}(m) = a + (-1)^i\phi_{d+1,i-1}(c)$. Now one verifies that

$$\begin{aligned}\alpha_{d,i+1}\begin{pmatrix}m \\ c\end{pmatrix} &\overset{(3.15)}{=} \begin{pmatrix}\alpha_{d+1,i} & (-1)^{i+1}\phi_{d+1,i-1} \\ & f_{d+1,i-1}\end{pmatrix}\begin{pmatrix}m \\ c\end{pmatrix} = \begin{pmatrix}\alpha_{d+1,i}(m) + (-1)^{i+1}\phi_{d+1,i-1}(c) \\ f_{d+1,i-1}(c)\end{pmatrix} \\ &= \begin{pmatrix}a + (-1)^i\phi_{d+1,i-1}(c) + (-1)^{i+1}\phi_{d+1,i-1}(c) \\ f_{d+1,i-1}(c)\end{pmatrix} = \begin{pmatrix}a \\ b\end{pmatrix},\end{aligned}$$

using the definition of m and c in the last two steps. This shows exactness at $A_{d,i}$.

To construct $\phi_{d,i+1}$, we note that

$$\alpha_{d,i}\phi_{d,i}f_{d,i+1} \overset{(3.17)}{=} \phi_{d,i-1}f_{d,i}f_{d,i+1} = 0,$$

because $F_{d,\bullet}$ is a chain complex. By the universal property of $\ker\alpha_{d,i}$, the morphism $\phi_{d,i}f_{d,i+1}$ factors through a morphism $F_{d,i+1} \to \ker\alpha_{d,i}$. By exactness at $A_{d,i+1}$, the morphism $\alpha_{d,i+1}$ surjects onto this kernel. The universal property of the free (and, in particular, projective) module $F_{d,i+1}$ gives a morphism $\phi_{d,i+1}$ such that

$$\alpha_{d,i+1}\phi_{d,i+1} = \phi_{d,i}f_{d,i+1}.$$

□

Recall that n is the number of parameters, so $C'_\bullet$ is a chain complex of $n-1$-parameter modules, and $F_{d,\bullet}$ has length $n-1$. Let $\tilde{C}'_\bullet$ be the chain complex defined by

$$\tilde{C}'_d = A_{d-n-1,n+1} = C'_d \oplus F_{d-1,0} \oplus \cdots \oplus F_{d-n,n-1}$$

with the boundary morphism

$$\tilde{\partial}'_{d+n+1} \colon \tilde{C}'_{d+n+1} = A_{d,n+1} \xrightarrow{\alpha_{d,n+1}} A_{d,n} \xrightarrow{\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)} A_{d,n} \oplus F_{d,n-1} = A_{d-1,n+1} = \tilde{C}'_{d+n}.$$

Lemma 3.7.4. *The objects $\tilde{C}'_\bullet$ and morphisms $\tilde{\partial}'_\bullet$ define an acyclic chain complex.*

Proof. This is a chain complex indeed, because

$$\tilde{\partial}'_{d+n+1}\tilde{\partial}'_{d+n+2} = \begin{pmatrix}\alpha_{d,n+1}\\0\end{pmatrix}\begin{pmatrix}\alpha_{d+1,n+1}\\0\end{pmatrix} \overset{(3.15)}{=} \begin{pmatrix}\alpha_{d+1,n} & \pm\phi_{d+1,n-1}\\ & f_{d+1,n-1}\end{pmatrix}\begin{pmatrix}\alpha_{d+1,n+1}\\0\end{pmatrix} = 0$$

because $A_{d+1,\bullet}$ is a chain complex. To show that it is acyclic, let

$$a \in \ker \tilde{\partial}'_{d+n+1} = \ker \begin{pmatrix}\alpha_{d,n+1}\\0\end{pmatrix} \subseteq A_{d,n+1} = \tilde{C}'_{d+n+1}.$$

Because $A_{d,\bullet}$ is acyclic, we get a $b \in A_{d,n+2}$ such that $a = \alpha_{d,n+2}(b)$. There is a commutative diagram

$$\begin{tikzcd}
A_{d,i+1} \arrow[rr, equal] \arrow[dr, "\alpha_{d,i+1}"'] & & A_{d+1,i} \oplus F_{d+1,i-1} \arrow[rr, "{(\mathrm{id},0)}"] \arrow[dr, "\left(\begin{smallmatrix}\alpha_{d+1,i} & \pm\phi_{d+1,i-1}\\ & f_{d+1,i-1}\end{smallmatrix}\right)"'] & & A_{d+1,i} \arrow[dl, "\left(\begin{smallmatrix}\alpha_{d+1,n+1}\\0\end{smallmatrix}\right)"] \\
& A_{d,i} \arrow[rr, equal] & & A_{d+1,i-1} \oplus F_{d+1,i-2}, &
\end{tikzcd}$$

for all $i \geq 2$. For $i = n+1$, we have $F_{d+1,n} = 0$ because $F_{d+1,\bullet}$ has length $n-1$. Therefore, we get $A_{d,n+2} = A_{d+1,n+1}$ and $\alpha_{d,n+2} = \left(\begin{smallmatrix}\alpha_{d+1,n+1}\\0\end{smallmatrix}\right) = \tilde{\partial}'_{d+n+2}$. In particular, $a = \alpha_{d,n+2}(b) = \tilde{\partial}'_{d+n+2}(b)$. This shows acyclicity. □

Now,

$$\begin{aligned}
\tilde{C}_d &= C_d \oplus E_kF_{d-1,0} \oplus \cdots \oplus E_kF_{d-n,n-1},\\
\tilde{\partial}_d &= \begin{pmatrix}
\partial_d & E_k\phi_{d-1,0} & -E_k\phi_{d-2,1} & \cdots & \pm E_k\phi_{d-n,n-1}\\
0 & E_kf_{d-1,0} & & & \\
\vdots & \ddots & E_kf_{d-1,1} & \ddots & \\
\vdots & & \ddots & \ddots & E_kf_{d-n,n-1}\\
0 & \cdots & \cdots & \cdots & 0
\end{pmatrix}
\end{aligned}$$

defines a chain complex, where we use that $E_kC'_\bullet \subseteq C_\bullet$. The chain complex $\tilde{C}_\bullet$ is a complex of free modules such that $(\tilde{C}_\bullet)_w = (C_\bullet)_w$ if $w_k < z_k$ and such that $(\tilde{C}_\bullet)_w$ is acyclic if $w_k \geq z_k$. Repeating this for all $1 \leq k \leq n$ gives an eventually acyclic chain complex $\hat{C}_\bullet$ such that $(\hat{C}_\bullet)_w = (C_\bullet)_w$ if $w < z$. This completes the construction.

3.8 Fringe presentations

Let M be a finitely supported persistence module. The close connection between free and injective resolutions of M from Theorem 3.2.8 can be used to compute a fringe presentation of M, in the sense of [104].

Definition 3.8.1. A *fringe presentation* of a persistence module M is a morphism $f\colon F \to I$ such that F is free, I is injective, and $M \cong \operatorname{im} f$.

Recall the Koszul complex functor

$$\Omega_\bullet\colon \quad 0 \to \overbrace{\mathrm{id}}^{\Omega_n} \xrightarrow{\kappa_n} \overbrace{\textstyle\bigoplus_{S\in\binom{[n]}{1}} \mathrm{Colim}_S}^{\Omega_{n-1}} \to \cdots \to \underbrace{\textstyle\bigoplus_{S\in\binom{[n]}{n-1}} \mathrm{Colim}_S}_{\Omega_1} \xrightarrow{\kappa_1} \underbrace{\mathrm{Colim}}_{\Omega_0} \to 0$$

from Definition 3.2.10. Recall that $\Omega_\bullet F(z)$ is a flat resolution of the injective module $I(z)\langle\epsilon\rangle \cong \nu F(z)\langle\epsilon\rangle$; that is, there is a quasi-isomorphism $\Omega_\bullet F(z) \to I(z)\langle\epsilon\rangle$. Since $\Omega_n F(z) = F(z)$, there is also a morphism $\Omega_\bullet F(z)[n] \to F(z)$ of chain complexes. Let $(F_\bullet, \partial^F_\bullet)$ be a chain complex of free modules, and let $I_\bullet := \nu F_\bullet\langle\epsilon\rangle$. Then $\Omega_\bullet F_\bullet$ is a double complex, whose differentials we denote by

$$\kappa_{ij} := \kappa_i F_j \colon \Omega_i F_j \to \Omega_{i-1} F_j, \qquad \partial_{ij} := \Omega_i \partial^F_j \colon \Omega_i F_j \to \Omega_i F_{j-1}.$$

Let $\tilde{\Omega}_\bullet$ be the total complex of $\Omega_\bullet F_\bullet$. Explicitly, $\tilde{\Omega}_\bullet$ is the chain complex with components and boundary morphisms

$$\tilde{\Omega}_q = \bigoplus_{i+j=q} \Omega_i F_j, \quad \partial^{\tilde{\Omega}}_q = \sum_{i+j=q} ((-1)^q \kappa_{ij} + \partial_{ij}).$$

Seeing $F(z)$ and $I(z)\langle\epsilon\rangle$ as chain complexes concentrated in degree zero, we obtain morphisms

$$\begin{tikzcd}
F(z)\colon & F(z) \arrow[d, equal] & & & \\
\Omega_\bullet F(z)[n]\colon \arrow[u] \arrow[d] & \Omega_n F(z) \arrow[r] & \Omega_{n-1} F(z) \arrow[r] & \cdots \arrow[r] & \Omega_0 F(z) \arrow[d] \\
I(z)\langle\epsilon\rangle[n]\colon & & & & I(z)\langle\epsilon\rangle
\end{tikzcd} \tag{3.19}$$

that give rise to the morphisms

$$I_\bullet[n] \xleftarrow{f[n]} \tilde{\Omega} \xrightarrow{g} F_\bullet$$

of chain complexes, where f is a chain homotopy equivalence. If $F_\bullet$ is eventually acyclic, then according to Theorem 3.2.8, the morphism g is a quasi-isomorphism. We show that g is a chain homotopy equivalence by constructing a homotopy inverse h of g, so that $f[n]h$ will be the desired fringe presentation.

The idea behind the following construction is to use the fact that by the assumption that $F_\bullet$ is eventually acyclic, the chain complexes $\operatorname{colim}_S F_\bullet$ of free $(n - |S|)$-parameter modules are acyclic and hence contractible for every S with $|S| > 0$. We may choose chain contractions of $\operatorname{colim}_S F_\bullet$ for every S, which give rise to chain contractions of $\Omega_i F_\bullet$ for every $0 \leq i < n$, and use these to construct the desired homotopy inverse h of g.

Let $0 \leq i < n$ and $S \in \binom{[n]}{i-k}$. Because $F_\bullet$ is eventually acyclic, $\operatorname{colim}_S F_\bullet$ is an acyclic and hence contractible chain complex. Let $(s^S_j)_{j\in\mathbf{Z}}$ be a chain contraction of $\operatorname{colim}_S F_\bullet$. Then the maps $s_{ij} := \bigoplus_{S\in\binom{[n]}{n-i}} \Delta_S s^S_j$ form a chain contraction $(s_{ij})_{j\in\mathbf{Z}}$ of $\Omega_i F_\bullet$, where $s_{ij}\colon \Omega_i F_j \to \Omega_i F_{j+1}$. Explicitly, this means that

$$\partial_{i,j+1} s_{ij} + s_{i,j-1} \partial_{ij} = \mathrm{id}_{\Omega_i F_j} \tag{3.20}$$

for all $j \in \mathbf{Z}$ and $0 \leq i < n$. For $k \geq 0$, define the morphism

$$t_{ijk} = \begin{Bmatrix} \mathrm{id}_{\Omega_i F_j} & \text{if } k = 0, \\ s_{i-k,j+k-1}\, \kappa_{i-k+1,j+k-1}\, t_{i,j,k-1} & \text{if } k > 0 \end{Bmatrix} \colon \Omega_i F_j \longrightarrow \Omega_{i-k} F_{j+k}. \tag{3.21}$$

Proposition 3.8.2. *Let $F_\bullet$ be an eventually acyclic chain complex of free modules, and let $\tilde{\Omega}_\bullet$ be as above.*

(*i*) *There is a morphism $h\colon F_\bullet \to \tilde{\Omega}_\bullet[n]$ of chain complexes. Its component*

$$h_q\colon F_q \longrightarrow \underbrace{\Omega_n F_q \oplus \cdots \oplus \Omega_0 F_{q+n}}_{\tilde{\Omega}_{q+n}}$$

is induced by the morphisms $(-1)^{j(q+1)} t_{nqj}\colon F_q \to \Omega_{n-j} F_{q_j}$ for $j = 0, \ldots, n$.

(*ii*) *The morphism h is a chain homotopy inverse of the morphism $g\colon \tilde{\Omega}_\bullet[n] \to F_\bullet$.*

Proof. We first show two statements about the morphisms t_{ijk}. For simplicity of notation, we only write the indices i and j for the rightmost morphism; for the others, these indices then are determined.

Claim. For all $i < n$, j and k, we have

$$\kappa \partial t_{ijk} = (-1)^k \kappa t_k \partial_{ij}. \tag{3.22}$$

Proof of claim. By induction over k (with the induction step at $(*)$), we get

$$\begin{aligned} \kappa \partial t_{ijk} &\overset{(3.21)}{=} \kappa \partial s \kappa t_{i,j,k-1} \overset{(3.20)}{=} \kappa (1 - s\partial) \kappa t_{i,j,k-1} \overset{(\dagger)}{=} -\kappa s \partial \kappa t_{i,j,k-1} \\ &\overset{(\dagger)}{=} -\kappa s \kappa \partial t_{i,j,k-1} \overset{(*)}{=} (-1)^k \kappa s \kappa t_{k-1} \partial_{ij} \overset{(3.21)}{=} (-1)^k \kappa t_k \partial_{ij}, \end{aligned}$$

using that $\partial_{\bullet\bullet}$ and $\kappa_{\bullet\bullet}$ form a double complex in $(\dagger)$.

The claim implies that

$$\begin{aligned} \partial t_{ijk} - \kappa t_{i,j,k-1} &\overset{(3.21)}{=} (\partial s - 1) \kappa t_{i,j,k-1} \\ &\overset{(3.20)}{=} -s \partial \kappa t_{i,j,k-1} \\ &\overset{(3.22)}{=} -(-1)^{k-1} s \kappa t_{k-1} \partial_{ij} \\ &\overset{(3.21)}{=} (-1)^k t_k \partial_{ij}. \end{aligned} \tag{3.23}$$

for all $i < n$, j and k. This allows us to drag symbols ∂ past the symbols t_{ijk}. Let ± stand for $(-1)^q$ and ∓ for $(-1)^{q+1}$. We obtain that

$$\begin{aligned} \partial_q^{\tilde{\Omega}} h_q &= \begin{pmatrix} \partial_{nq} & & & & \\ \pm\kappa_{nq} & \partial_{n-1,q+1} & & & \\ & \pm\kappa_{n-1,q+1} & \partial_{n-2,q+2} & & \\ & & \ddots & \ddots & \\ & & & \pm\kappa_{1,q+n-1} & \partial_{0,q+n} \end{pmatrix} \begin{pmatrix} \mathrm{id}_{\Omega_n F_q} \\ \mp t_{nq1} \\ t_{nq2} \\ \vdots \\ (\mp)^{n-1} t_{n,q,n-1} \\ (\mp)^n t_{nqn} \end{pmatrix} \\ &= \begin{pmatrix} \partial_{nq} \\ \mp(\partial t_{nq1} - \kappa_{nq}) \\ -(\partial t_{nq2} - \kappa t_{nq1}) \\ \vdots \\ (\mp)^n (\partial t_{nqn} - \kappa t_{n,q,n-1}) \end{pmatrix} \overset{(3.23)}{=} \begin{pmatrix} \partial_{nq} \\ \pm t_1 \partial_{nq} \\ t_2 \partial_{nq} \\ \vdots \\ (\pm)^n t_n \partial_{nq} \end{pmatrix} = h_{q-1} \partial_q^F. \end{aligned}$$

Therefore, h is a chain map. It remains to show that h is a homotopy inverse of g. We clearly have $gh = \mathrm{id}_{F_\bullet}$. It remains to show that $\mathrm{id}_{\tilde{\Omega}_\bullet}$ and hg are chain homotopic. For every j, let

$$\sigma_j := \begin{pmatrix} 0 & & & & \\ 0 & s_{n-1,q+1} & & & \\ 0 & \mp t_1 s_{n-1,q+1} & s_{n-2,q+2} & & \\ 0 & t_2 s_{n-1,q+1} & \mp t_1 s_{n-2,q+2} & s_{n-3,q+3} & \\ 0 & \mp t_3 s_{n-1,q+1} & t_2 s_{n-2,q+2} & \mp t_1 s_{n-3,q+3} & s_{n-4,q+4} \\ \vdots & & & & & \cdots \end{pmatrix} : \tilde{\Omega}_{q+n} \to \tilde{\Omega}_{q+n+1}.$$

We remark that we have the following two recursive relationship of the symbols t_{ijk}, which both follow directly from the definition:

$$t_{i,j,k+1} \overset{(3.21)}{=} s_{i-k-1,j+k}\kappa_{i-k,j+k}t_{ijk} = t_{i-1,j+1,k}s_{i-1,j}\kappa_{i,j}. \tag{3.24}$$

From this, it follows that for all $k \geq 0$, we have

$$\begin{aligned} & (\kappa t_k s_{ij} - \partial t_k s_{ij}) + (-1)^{k+1}(t_{k+1}s\partial_{ij} - t_k s\kappa_{ij}) \\ \overset{(3.24)}{=} & (1-\partial s)\kappa t_k s_{ij} + (-1)^k t_k s\kappa(s\partial_{ij} - 1) \\ \overset{(3.20)}{=} & s\partial\kappa t_k s_{ij} - (-1)^k t_k s\kappa\partial s_{ij} \\ \overset{(3.24)}{=} & s\partial\kappa t_k s_{ij} - (-1)^k s\kappa t_k \partial s_{ij} \\ \overset{(3.22)}{=} & s\partial\kappa t_k s_{ij} - s\partial\kappa t_k s_{ij} \\ = & \ 0. \end{aligned} \tag{3.25}$$

We will use this to show that the σ_j form a chain homotopy between $\mathrm{id}\,\bar{\Omega}_\bullet$ and hg. In favor of legibility, we leave out the indices i and j of all morphisms s_{ij}, κ_{ij}, ∂_{ij} and t_{ijk} in the following. We obtain

$$\begin{aligned} & \partial^{\tilde{\Omega}}_{j+1}\sigma_j + \sigma_{j-1}\partial^{\tilde{\Omega}}_j \\ = & \begin{pmatrix} \partial & & & & \\ \mp\kappa & \partial & & & \\ & \mp\kappa & \partial & & \\ & & \mp\kappa & \partial & \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} 0 & & & & \\ 0 & s & & & \\ 0 & \pm t_1 s & s & & \\ 0 & t_2 s & \pm t_1 s & s & \\ \vdots & & & & \ddots \end{pmatrix} + \begin{pmatrix} 0 & & & & \\ 0 & s & & & \\ 0 & \mp t_1 s & s & & \\ 0 & t_2 s & \mp t_1 s & s & \\ \vdots & & & & \ddots \end{pmatrix} \begin{pmatrix} \partial & & & & \\ \pm\kappa & \partial & & & \\ & \pm\kappa & \partial & & \\ & & \pm\kappa & \partial & \\ & & & & \ddots \end{pmatrix} \\ = & \begin{pmatrix} 0 & & & & \\ \pm s\kappa & s\partial + \partial s & & & \\ -t_1 s\kappa & \mp\kappa s \pm \partial t_1 s \mp t_1 s\partial \pm s\kappa & s\partial + \partial s & & \\ \pm t_2 s\kappa & -\kappa t_1 s + \partial t_2 s - t_1 s\kappa & \mp\kappa s \pm \partial t_1 s \mp t_1 s\partial \pm s\kappa & s\partial + \partial s & \\ \vdots & & & & \cdots \end{pmatrix} \\ \overset{(3.25)}{=} & \begin{pmatrix} 0 & & & & \\ \pm t_1 & \mathrm{id}_{\Omega_{n-1}F_{q+1}} & & & \\ -t_2 & 0 & \mathrm{id}_{\Omega_{n-2}F_{q+2}} & & \\ \pm t_3 & 0 & 0 & \mathrm{id}_{\Omega_{n-3}F_{q+3}} & \\ \vdots & & & & \cdots \end{pmatrix} \\ = & \ \mathrm{id}_{\tilde{\Omega}_{n+q}} - h_j g_j. \end{aligned}$$

This shows that hg and $\mathrm{id}_{\tilde{\Omega}_\bullet}$ are chain homotopic. □

Corollary 3.8.3. *Let $M \in \mathrm{Vec}^{\mathbf{Z}^n}$ be finitely supported, $F_\bullet$ be a free resolution of M of length n, let t_{ijk} be as above, and let $a\colon \Omega_0 F_n \to \nu F_n\langle\epsilon\rangle$ be the augmentation map of the free resolution $\Omega_\bullet F_n$ of $\nu F_n\langle\epsilon\rangle$. Then the composite morphism*

$$F_0 = \Omega_n F_0 \xrightarrow{t_{n,0,n}} \Omega_0 F_n \xrightarrow{a} \nu F_n\langle\epsilon\rangle$$

is a fringe presentation of M.

Proof. Because $F_\bullet$ is a free resolution of M, there is a quasi-isomorphism $F_\bullet \to M$. According to Proposition 3.8.2, we obtain the two homotopy equivalences

$$\begin{tikzcd}
F_\bullet\colon \arrow[d, "h", "\simeq"'] & \cdots \arrow[r] & F_1 \arrow[r] \arrow[d] & F_0 \arrow[r] \arrow[d, "h_0"] & 0 \arrow[r] \arrow[d] & \cdots \\
\tilde{\Omega}_\bullet[n]\colon \arrow[d, "f[n]", "\simeq"'] & \cdots \arrow[r] & \tilde{\Omega}_{n+1} \arrow[r] \arrow[d] & \tilde{\Omega}_n \arrow[r] \arrow[d, "f_n"] & \tilde{\Omega}_{n-1} \arrow[r] \arrow[d] & \cdots \\
I_\bullet[n]\colon & \cdots \arrow[r] & 0 \arrow[r] & I_n \arrow[r] & I_{n-1} \arrow[r] & \cdots,
\end{tikzcd}$$

of chain complexes, where f and h are as above, $I_n = \nu F_n\langle\epsilon\rangle$, and $\tilde{\Omega}_n = \Omega_n F_0 \oplus \cdots \oplus \Omega_0 F_n$. The image of the homotopy equivalence $f[n]h$ is quasi-isomorphic to M. The only non-zero component of $f[n]h$ is

$$f_n h_0 = (0, \ldots, 0, a) \begin{pmatrix} \mathrm{id}_{\Omega_n F_0} \\ -t_{n01} \\ \vdots \\ (-1)^n t_{n0n} \end{pmatrix} = (-1)^n a t_{n0n}\colon F_0 \to \Omega_0 F_n \to I_n.$$

This shows that $\operatorname{im} a t_{n0n} = \operatorname{im} f_n h_0 = M$. □

Two-parameter modules Recall that $\Omega_i F_\bullet = \bigoplus_{S \in \binom{[n]}{n-k}} \Delta_S \operatorname{colim}_S F_\bullet$, where $\operatorname{colim}_S F_\bullet$ is a chain complex of $(n - |S|)$-parameter modules. If $n = 2$ and $F_\bullet$ is a free resolution of a finitely supported module M, then

$$\Omega_1 F_\bullet = \Delta_{\{1\}} \operatorname{colim}_{\{1\}} F_\bullet \oplus \Delta_{\{2\}} \operatorname{colim}_{\{2\}} F_\bullet,$$

where $\operatorname{colim}_{\{l\}} F_\bullet$ is a contractible chain complex of free one-parameter modules for $l \in \{1, 2\}$. In this case, choosing a chain contraction of $\Omega_1 F_\bullet$ can be done as follows.

Let $C_\bullet$ be a chain complex of free one-parameter persistence modules, and fix a basis of $C_\bullet$. Applying the Standard Algorithm (Algorithm 1) to the boundary matrices of $C_\bullet$ gives a persistence basis of $C_\bullet$; cf. Example 2.2.7. If $\operatorname{colim} C_\bullet$ is acyclic, then a persistence basis of $C_\bullet$ is a collection $(c_i)_{i\in J} \subseteq C_\bullet$ for some index set J such that $(\partial c_i)_{i\in J} \cup (c_i)_{i\in J}$ is a basis of $C_\bullet$. In this case, each pair $(\partial c_i, c_i)$ represents a bar $(g(\partial c_i), g(c_i)) \in \operatorname{barc} H_\bullet(C_\bullet)$. If $C_\bullet$ is acyclic, then $\operatorname{barc} H_\bullet(C_\bullet)$ contains no bars of non-zero length; which means that $g(\partial c_i) = g(c_i)$ for all $i \in J$. In particular, the assignment

$$C_\bullet \to C_{\bullet+1}, \quad \partial c_i \mapsto c_i, \quad c_i \mapsto 0$$

is a well-defined chain contraction of $C_\bullet$. Applying this to the contractible $\operatorname{colim}_{\{l\}} F_\bullet$ complexes of one-parameter modules for $l \in \{1, 2\}$ and (by an analogous construction) to the contractible complex $\Omega_0 F_\bullet$ of vector spaces gives the desired contractions s_1 of $\Omega_1 F_\bullet$ and s_0 of $\Omega_0 F_\bullet$. In the light of Corollary 3.8.3, these can be used to construct a fringe presentation of M.

Chapter 4

Persistent cohomology of freely resolved cochain complexes

In this section, we introduce a different approach to computing a minimal free resolution of the two-parameter persistent cohomology of a one-critically $\mathbf{Z}^2$-filtered simplicial complex K_*. Namely, we compute minimal free resolutions of $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ directly from the cochain complexes $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$, respectively. Using Corollary 3.2.11, these can be used to obtain minimal free resolutions of $H_\bullet(K_*)$ and $H_\bullet(K, K_*)$. If $K = \operatorname{colim}_{\mathbf{Z}^2} K_*$ is acyclic, then the long exact sequence (2.3) shows that $H^d(K_*) \cong H^{d+1}(K, K_*)$, which is why we study both absolute and relative cohomology here.

Arguably, computing $H^d(K_*)$ or $H^{d+1}(K, K_*)$ suggests itself more easily than computing $H^{d+2}(N^\bullet(K_*))$, given that one is ultimately interested in computing $H_d(K_*)$. The computational challenge lies in the fact that as mentioned earlier, for two and more parameters, $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$ are no cochain complexes of free modules. Actually, these are not even complexes of finitely generated modules.

To address this, we choose a $z > g(\sigma)$ for all $\sigma \in K_*$, and replace $C^\bullet(K_*)$ and $C^\bullet(K, K_*)$ by complexes $R_z C^\bullet(K_*)$ and $R_z C^\bullet(K, K_*)$ of finitely generated modules. We choose a free resolution $C^d_\bullet$ of $R_z C^d(K_*)$ for each d, and use these to compute minimal free resolutions of $H^d(R_z C^\bullet(K_*))$ and $H^{d+1}(R_z C^\bullet(K, K_*))$.

We start in Section 4.1 by deriving explicit formulas for free resolutions of the cocycles, coboundaries and cohomology of an arbitrary cochain complex $C^\bullet$ of modules C^d, for which a free resolution $F^d_\bullet \to C^d$ are known for each d; see Theorem 4.1.7. In Section 4.2, we apply this to the finitely generated modules $R_z C^\bullet(K_*)$ and $R^z C^\bullet(K, K_*)$. This gives minimal free resolutions of $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$, see Propositions 4.2.5 and 4.2.7. In Section 4.4, we devise further algorithms to efficiently compute a minimal free resolution of $H^{d+1}(K, K_*)$. In particular, we develop an optimization scheme analogous to clearing; see Proposition 4.4.6. We also show in Theorem D (page 90) that if K is acyclic, then a minimal free resolution of $H^{d+1}(K, K_*)$ can be computed from the coboundary morphism $\delta^{d+1} \colon C^d(K, K_*) \to C^{d+1}(K, K_*)$; that is, it is not necessary to know the coboundary morphism δ^{d+2} explicitly.

Recall from Corollary 2.2.11 that in one-parameter persistence, $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ determine each other uniquely up to isomorphism even if K is not acyclic. We show in Theorem B(b) (page 98) that this is not the case for two or more parameters. This is analogous to the example presented in Section 3.6.

Parts of the results presented in this chapter are joint work with Ulrich Bauer and Michael Lesnick.

4.1 Free resolutions of (co)kernels, images and subquotients

Let $A = k[x_1, x_2]$, Let L, M and N be finitely generated A-modules, let $L \xrightarrow{f} M \xrightarrow{g} N$ be morphisms such that $gf = 0$. let $L_\bullet, M_\bullet$ and $N_\bullet$ be free resolutions of L, M and N, and let $f_\bullet \colon L_\bullet \to M_\bullet$ and $g_\bullet \colon M_\bullet \to N_\bullet$ be lifts of f and g. In this section, we derive closed expressions for free resolutions of $\ker g$, $\operatorname{im} f$, $\operatorname{coker} f$ and $\ker g/\operatorname{im} f$. Recall that according to Corollary 2.3.20, kernels of morphisms of free modules are free. In particular, all pullbacks in the following lemmas are free, because they are difference kernels of morphism of free modules.

Lemma 4.1.1 (Kernel resolution). *Let $g\colon M \to N$ be a morphism of finitely generated modules, let $M_\bullet$ and $N_\bullet$ be free resolutions of M and N of length 2, and let $g_\bullet\colon M_\bullet \to N_\bullet$ be a lift of g. Then the sequence*

$$K_\bullet\colon \qquad 0 \to M_2 \xrightarrow{k_2 := \left(\begin{smallmatrix} m_2 \\ -g_2 \end{smallmatrix}\right)} M_1 \oplus N_2 \xrightarrow{k_1 := \left(\begin{smallmatrix} m_1 & 0 \\ g_1 & m_2 \end{smallmatrix}\right)} M_0 \times_{N_0} N_1 \to \ker g \tag{4.1}$$

is a free resolution of $\ker g$, *and*

$$\begin{tikzcd}
K_\bullet\colon \arrow[d, "\kappa_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "k_2"] \arrow[d, equal, "\kappa_2"', "\mathrm{id}_{M_2}"] & M_1 \oplus N_2 \arrow[r, "k_1"] \arrow[d, "\kappa_1"', "=(\mathrm{id}_{M_1},0)"] & M_0 \times_{N_0} N_1 \arrow[r, "k_0"] \arrow[d, "\kappa_0"', "(\mathrm{id}_{M_0},0)"] & \ker g \arrow[d, "\kappa"] \\
M_\bullet\colon & 0 \arrow[r] & M_2 \arrow[r, "m_2"'] & M_1 \arrow[r, "m_1"'] & M_0 \arrow[r] & M
\end{tikzcd} \tag{4.2}$$

is a lift of the canonical inclusion $\kappa\colon \ker g \hookrightarrow M$.

Remark 4.1.2. The matrices in (4.1) have morphisms as entries. This is a convenient way to write down morphisms from and to direct sums of modules. Note that a priori, the matrix $k_1 = \left(\begin{smallmatrix} m_1 & 0 \\ g_1 & m_2 \end{smallmatrix}\right)$ from the statement of the lemma only defines a morphism $M_1 \oplus N_2 \to M_0 \oplus N_1$. It is part of the lemma that this morphism factors through the submodule $M_0 \times_{N_0} N_1$ of $M_0 \oplus N_1$.

Proof of Lemma 4.1.1. During the proof, we will explain the construction of the first row of the following (supposedly commutative) diagram:

$$\begin{tikzcd}
K_\bullet\colon & 0 \arrow[r] & M_2 \arrow[r, "k_2"] \arrow[d, "\kappa_2"] & M_1 \oplus N_2 \arrow[r, "k_1"] \arrow[d, "\kappa_1"] & M_0 \times_{N_0} N_1 \arrow[r, "k_0"] \arrow[d, "\kappa_0"] & \ker g \arrow[r] \arrow[d, hook, "\kappa"] & 0 \\
M_\bullet\colon & 0 \arrow[r] & M_2 \arrow[r, "m_2"'] \arrow[d, "g_2"] & M_1 \arrow[r, "m_1"'] \arrow[d, "g_1"] & M_0 \arrow[r, "m_0"'] \arrow[d, "g_0"] & M \arrow[r] \arrow[d, "g"] & 0 \\
N_\bullet\colon & 0 \arrow[r] & N_2 \arrow[r, "n_2"'] & N_1 \arrow[r, "n_1"'] & N_0 \arrow[r, "n_0"'] & N \arrow[r] & 0.
\end{tikzcd} \tag{4.3}$$

Its middle and lower row are exact, and the squares between them commute. We construct maps $k_\bullet$ and $\kappa_\bullet$ that render the first row exact and the diagram commutative.

Claim. There exists a surjective morphism $M_0 \times_{N_0} N_1 \to \ker g$.

Proof of claim. Let $K_0 := M_0 \times_{N_0} N_1 = \ker(g_0, -n_1)$. The morphism $\kappa_0 := (\mathrm{id}_{M_1}, 0)\colon K_0 \to M$ satisfies

$$g m_0 \kappa_0 = (g m_0, 0)|_{K_0} \overset{(\dagger)}{=} n_0(0, n_1)|_{K_0} = 0,$$

using the definition of K_0 in (†). By the universal property of $\ker g$, there exists a unique morphism $k_0\colon K_0 \to \ker g$ such that $m_0 \kappa_0 = \kappa k_0$. To show that k_0 is surjective, let $x \in \ker g$. Since m_0 is surjective, there exists an $x' \in M_0$ such that $m_0(x') = \kappa(x)$. We obtain

$$n_0 g_0(x') = g m_0(x') = g(x) = 0.$$

Exactness of the lower row implies that there exists a $y \in N_1$ such that $n_1(y) = g_0(x')$. By definition of K_0, this means that $(x', y) \in K_0$. We obtain $\kappa k_0(x', y) = m_0(x') = \kappa(x)$. Since κ is injective, this implies that $k_0(x', y) = x$, hence k_0 is surjective. This proves the claim.

Now, consider $\ker k_0$ and the pullback

$$K_0 \times_{M_0} M_1 = \ker\left(\begin{smallmatrix} g_0 & -n_1 & 0 \\ \mathrm{id}_{M_0} & 0 & -m_1 \end{smallmatrix}\right) \subseteq M_0 \oplus N_1 \oplus M_1. \tag{4.4}$$

By injectivity of κ, we have $\ker k_0 = \ker \kappa k_0 = \ker m_0\kappa_0$. Applying the claim to the morphism $m_0\kappa_0 \colon K_0 \to M$ (instead of g) shows that the morphism $(\mathrm{id}_{K_0}, 0) \colon K_0 \times_{M_0} M_1 \to K_0$ surjects onto $\ker k_0$. For $K_1 \coloneqq M_1 \oplus N_2$, consider the morphism

$$\alpha \coloneqq \begin{pmatrix} m_1 & 0 \\ g_1 & -n_2 \\ \mathrm{id}_{M_1} & 0 \end{pmatrix} \colon K_1 \longrightarrow K_0 \oplus M_1. \tag{4.5}$$

Because n_2 is injective, so is α. We show that $\operatorname{im}\alpha = K_0 \times_{M_0} M_1$. Since

$$\left(\begin{smallmatrix} g_0 & -n_1 & 0 \\ \mathrm{id}_{M_0} & 0 & -m_1 \end{smallmatrix}\right)\begin{pmatrix} m_1 & 0 \\ g_1 & -n_2 \\ \mathrm{id}_{M_1} & 0 \end{pmatrix} = 0. \tag{4.6}$$

we have $\operatorname{im}\alpha \subseteq K_0 \times_{M_0} M_1$. To show that $\operatorname{im}\alpha = K_0 \times_{M_0} M_1$, let $(x, y, x') \in K_0 \times_{M_0} M_1$. According to (4.4), we get that $x = m_1(x')$ and

$$n_1(g_1(x') - y) = g_0(m_1(x')) - n_1(y) = g_0(x) - n_1(y) = 0. \tag{4.7}$$

Exactness of the lower row implies that $g_1(x') - y = n_2(y')$ for some $y' \in N_2$. Therefore,

$$\alpha\left(\begin{smallmatrix} x' \\ y' \end{smallmatrix}\right) = \begin{pmatrix} m_1 & 0 \\ g_1 & -n_2 \\ \mathrm{id}_{M_1} & 0 \end{pmatrix}\left(\begin{smallmatrix} x' \\ y' \end{smallmatrix}\right) = \begin{pmatrix} m(x') \\ g_1(x') - n_2(y') \\ x' \end{pmatrix} = \left(\begin{smallmatrix} x \\ y \\ x' \end{smallmatrix}\right). \tag{4.8}$$

This shows that α maps K_1 isomorphically to $K_0 \times_{M_0} M_1$. We obtain that

$$k_1 = \left(\begin{smallmatrix} m_1 & 0 \\ g_1 & -n_2 \end{smallmatrix}\right) \colon K_1 \overset{\alpha}{\hookrightarrow} K_0 \times_{M_0} M_1 \xrightarrow{(\mathrm{id}_{K_0}, 0)} K_0 \tag{4.9}$$

surjects onto $\ker k_0$. To obtain K_2, we calculate

$$\begin{aligned} \ker k_1 = \ker\left(\begin{smallmatrix} m_1 & 0 \\ g_1 & -n_2 \end{smallmatrix}\right) = \ker(m_1, 0) \cap \ker(g_1, -n_2) &\overset{(i)}{\cong} (M_2 \oplus N_2) \cap \ker(g_1, -n_2) \\ \overset{(ii)}{\cong} (M_2 \oplus N_2) \cap \ker(g_1 m_2, -n_2) = \ker(n_2 g_2, -n_2) &\overset{(iii)}{\cong} \ker(g_2, -\mathrm{id}_{N_2}) \\ &\cong \{(m, g_2(m)) \mid m \in M_2\} \cong M_2, \end{aligned} \tag{4.10}$$

where we use exactness of $M_\bullet$ in (i), injectivity of m_2 in (ii), and injectivity of n_2 in (iii). With $K_2 \coloneqq M_2$, we obtain that the morphism

$$k_2 \coloneqq \left(\begin{smallmatrix} m_2 \\ -g_2 \end{smallmatrix}\right) \colon K_2 \xrightarrow[\cong]{\left(\begin{smallmatrix} \mathrm{id}_{M_2} \\ -g_2 \end{smallmatrix}\right)} \ker(g_2, \mathrm{id}_{N_2}) = \ker(g_1 m_2, n_2) \xrightarrow{\left(\begin{smallmatrix} m_2 & 0 \\ 0 & \mathrm{id}_{N_2} \end{smallmatrix}\right)} K_1 \tag{4.11}$$

is injective and surjects onto $\ker k_1$. This completes the resolution $K_\bullet$ of $\ker g$. One sees easily that $\kappa_1 = (\mathrm{id}_{M_1}, 0)$ and $\kappa_2 = \mathrm{id}_{M_2}$ make the diagram (4.3) commute. □

Lemma 4.1.3 (Image resolution). *Let $f \colon L \to M$ be a morphism of finitely generated modules, let $L_\bullet$ and $M_\bullet$ be free resolutions of length 2 of L and M, and let $f_\bullet \colon L_\bullet \to M_\bullet$ be a lift of f. Then*

$$J_\bullet \colon \qquad 0 \to M_2 \xrightarrow{j_2 \coloneqq \left(\begin{smallmatrix} 0 \\ m_2 \end{smallmatrix}\right)} L_0 \times_{M_0} M_1 \xrightarrow{j_1 \coloneqq (\mathrm{id}_{L_0}, 0)} L_0 \xrightarrow{j_0} \operatorname{im} f \tag{4.12}$$

is a free resolution of $\operatorname{im} f$, *and*

$$\begin{tikzcd} L_\bullet\colon \arrow[d, "\pi_\bullet"'] & 0 \arrow[r] & L_2 \arrow[r, "l_2"] \arrow[d, "\pi_2"', "f_2"] & L_1 \arrow[r, "l_1"] \arrow[d, "\pi_1"', "\left(\begin{smallmatrix} l_1 \\ f_1 \end{smallmatrix}\right)"] & L_0 \arrow[r, "l_0"] \arrow[d, equal, "\pi_0"'] & L \arrow[d, "\pi"] \\ J_\bullet\colon & 0 \arrow[r] & M_2 \arrow[r, "j_2"'] & L_0 \times_{M_0} M_1 \arrow[r, "j_1"'] & L_0 \arrow[r, "j_0"'] & \operatorname{im} f. \end{tikzcd} \tag{4.13}$$

is a lift of the canonical map $\pi \colon L \to \operatorname{im} f$.

Proof. The morphism $j_0 := \pi l_0\colon L_0 \to \operatorname{im} f$ is surjective because it is a composition of two surjective morphisms. Applying Lemma 4.1.1 to j_0, we get that $j_1 := (\mathrm{id}_{L_0}, 0)\colon L_0 \times_{M_0} M_1 \to L_0$ surjects onto $\ker j_0$. Lastly, $\ker j_1 = 0 \times_{M_0} M_1 \cong \ker m_1 \cong M_2$; hence the map $j_2 := \left(\begin{smallmatrix}0\\ m_2\end{smallmatrix}\right)\colon M_2 \to L_0 \times_{M_0} M_1$ maps M_2 isomorphically to $\ker j_1$, so (4.12) is a free resolution of $\operatorname{im} f$. Commutativity of (4.13) can be seen immediately. □

Lemma 4.1.4 (Image inclusion). *Let $L \xrightarrow{f} M \xrightarrow{g} N$ be morphisms of finitely generated modules with $gf = 0$, let $L_\bullet, M_\bullet$ and $N_\bullet$ be free resolutions of length 2, and let $f_\bullet\colon L_\bullet \to M_\bullet$ and $g_\bullet\colon M_\bullet \to N_\bullet$ be lifts of f and g. Then there are maps $\zeta\colon L_0 \to N_1$ and $\eta\colon L \times_{M_0} M_1 \to N_2$ such that the inclusion map $i\colon \operatorname{im} f \hookrightarrow \ker g$ is lifted by the following morphism of free resolutions:*

$$\begin{tikzcd}
J_\bullet\colon \arrow[d, "i_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "j_2"] \arrow[d, equal, "i_2"] & L_0 \times_{M_0} M_1 \arrow[r, "j_1"] \arrow[d, "i_1 = \left(\begin{smallmatrix}0 & \mathrm{id}\\ & \eta\end{smallmatrix}\right)"] & L_0 \arrow[r, "j_0"] \arrow[d, "i_0 = \left(\begin{smallmatrix}f_0\\ \zeta\end{smallmatrix}\right)"] & \operatorname{im} f \arrow[d, "i"] \\
K_\bullet\colon & 0 \arrow[r] & M_2 \arrow[r, "k_2"'] & M_1 \oplus N_2 \arrow[r, "k_1"'] & M_0 \times_{N_0} N_1 \arrow[r, "k_0"'] & \ker g.
\end{tikzcd} \tag{4.14}$$

Proof. Consider the diagram

$$\begin{tikzcd}
L_\bullet\colon \arrow[d, "\pi_\bullet"'] \arrow[ddd, bend left, "f_\bullet"] & 0 \arrow[r] & L_2 \arrow[r, "l_2"] \arrow[d, "f_2"'] \arrow[ddd, bend left, "f_2"] & L_1 \arrow[r, "l_1"] \arrow[d, "\left(\begin{smallmatrix}l_1\\ f_1\end{smallmatrix}\right)"'] \arrow[ddd, bend left, "f_1"] & L_0 \arrow[r, "l_0"] \arrow[d, equal] \arrow[ddd, bend left, "f_0"] & L \arrow[r] \arrow[d, "\pi"] \arrow[ddd, bend left, "f"] & 0 \\
J_\bullet\colon \arrow[d, dashed, "i_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "\left(\begin{smallmatrix}0\\ m_2\end{smallmatrix}\right)"] \arrow[d, dashed, "i_2"'] \arrow[dr, phantom, "(***)"] & \overbrace{L_0 \times_{M_0} M_1}^{J_1} \arrow[r, "(\mathrm{id},0)", "j_1"'] \arrow[d, dashed, "i_1"] \arrow[dr, phantom, "(**)"] & L_0 \arrow[r, "j_0"] \arrow[d, dashed, "i_0"] \arrow[dr, phantom, "(*)"] & \operatorname{im} f \arrow[r] \arrow[d, "i"] & 0 \\
K_\bullet\colon \arrow[d, "\kappa_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "\left(\begin{smallmatrix}m_2\\ -g_2\end{smallmatrix}\right)"'] \arrow[d, equal] & \underbrace{M_1 \oplus N_2}_{K_1} \arrow[r, "\left(\begin{smallmatrix}m_1 & 0\\ g_2 & n_2\end{smallmatrix}\right)"'] \arrow[d, "(\mathrm{id},0)"'] & \underbrace{M_0 \times_{N_0} N_1}_{K_0} \arrow[r, "k_0"] \arrow[d, "(\mathrm{id},0)"'] & \ker g \arrow[r] \arrow[d, "\kappa"'] & 0 \\
M_\bullet\colon \arrow[d, "g_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "m_2"'] \arrow[d, "g_2"'] & M_1 \arrow[r, "m_1"'] \arrow[d, "g_1"] & M_0 \arrow[r, "m_0"'] \arrow[d, "g_0"] & M \arrow[r] \arrow[d, "g"] & 0 \\
N_\bullet\colon & 0 \arrow[r] & N_2 \arrow[r, "n_2"'] & N_1 \arrow[r, "n_1"'] & N_0 \arrow[r, "n_0"'] & N \arrow[r] & 0.
\end{tikzcd} \tag{4.15}$$

According to Lemmas 4.1.1 and 4.1.3, all rows are exact, and all squares except between rows $J_\bullet$ and $K_\bullet$ commute. We construct the dashed maps $i_\bullet$ that make the diagram commute.

By commutativity, $n_0 g_0 f_0 = g f l_0 = 0$. Therefore, the morphism $g_0 f_0\colon L_0 \to N_0$ factors uniquely through $\ker n_0$. By the universal property of the free (and in particular projective) module N_1, there is a morphism $\zeta\colon L_0 \to N_1$ such that $g_0 f_0 = n_1 \zeta$. Then $(g_0, -n_1)\left(\begin{smallmatrix}f_0\\ \zeta\end{smallmatrix}\right) = 0$, so the morphism $i_0 := \left(\begin{smallmatrix}f_0\\ \zeta\end{smallmatrix}\right)\colon L_0 \to M_0 \oplus N_1$ factors through $K_0 = \ker(g_0, -n_1)$. This shows commutativity of $(*)$.

For the construction of i_1, consider the map $(\zeta, -g_1)\colon L_0 \oplus M_1 \to N_1$. Its restriction to $J_1 = \ker(f_0, -m_1)$ satisfies

$$n_1(\zeta, -g_1)|_{J_1} \overset{a)}{=} (g_0 f_0, -n_1 g_1)|_{J_1} \overset{b)}{=} g_0(f_0, -m_1)|_{J_1} \overset{c)}{=} 0, \tag{4.16}$$

using the definition of ζ in $a)$, the commutativity of the lower row in $b)$ and the definition of J_1 in $c)$. Bt the universal property of $\ker n_1$, there is a unique morphism $\eta\colon J_1 \to \ker n_1 = N_2$ such that $n_2 \eta = (\zeta, -g_1)|_{J_1}$. We check that the map $i_1 := \left(\begin{smallmatrix}0 & \mathrm{id}_{M_1}\\ & \eta\end{smallmatrix}\right)\colon J_1 \to K_1$ makes the square $(**)$ commute. We calculate

$$k_1 i_1 = \left(\begin{smallmatrix}m_1 & 0\\ g_1 & n_2\end{smallmatrix}\right)\left(\begin{smallmatrix}0 & \mathrm{id}\\ & \eta\end{smallmatrix}\right)\Big|_{J_1} = \left(\begin{smallmatrix}0 & m_1\\ \zeta & 0\end{smallmatrix}\right)\Big|_{J_1} \overset{(\dagger)}{=} \left(\begin{smallmatrix}f_0 & 0\\ \zeta & 0\end{smallmatrix}\right)\Big|_{J_1} = \left(\begin{smallmatrix}f_0\\ \zeta\end{smallmatrix}\right)(\mathrm{id}_{K_0}, 0)|_{J_1} = i_0 j_1,$$

using the definition of J_1 in $(\dagger)$. Therefore, $(**)$ commutes.

Lastly, we check that $i_2 = \mathrm{id}_{M_2}$ makes the square $(***)$ in commute. By the definition of η, we get

$$n_2 \eta j_2 = (\zeta, -g_1)\left(\begin{smallmatrix} 0 \\ m_2 \end{smallmatrix}\right) = -n_2 g_2.$$

As n_2 is injective, this implies $\eta j_2 = -g_2$. Therefore,

$$k_2 i_2 = \left(\begin{smallmatrix} m_2 \\ -g_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 & \mathrm{id}_{M_1} \\ & \eta \end{smallmatrix}\right)\left(\begin{smallmatrix} 0 \\ m_2 \end{smallmatrix}\right) = i_1 j_2,$$

so $(***)$ commutes. □

Lemma 4.1.5 (Cokernel resolution). *Let $f\colon L \to M$ be a morphism of finitely generated modules, let $L_\bullet$ and $M_\bullet$ be free resolutions of length* 2 *of L and M, and let $f_\bullet\colon L_\bullet \to M_\bullet$ be a lift of f. Then*

$$K'_\bullet\colon \qquad 0 \longrightarrow L_0 \times_{M_0} M_1 \xrightarrow[k'_2]{} L_0 \oplus M_1 \xrightarrow[k'_1]{(-f_0, m_1)} M_0 \xrightarrow{k'_0} \operatorname{coker} f \tag{4.17}$$

is a free resolution of $\operatorname{coker} f$*, and*

$$\begin{tikzcd}
M_\bullet\colon \arrow[d, "\kappa'_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "m_2"] \arrow[d, "\kappa'_2"', "\left(\begin{smallmatrix} 0 \\ m_2 \end{smallmatrix}\right)"] & M_1 \arrow[r, "m_1"] \arrow[d, "\kappa'_1"', "\left(\begin{smallmatrix} 0 \\ \mathrm{id}_{M_1} \end{smallmatrix}\right)"] & M_0 \arrow[r, "m_0"] \arrow[d, "\kappa'_0"', "\mathrm{id}_{M_0}"] & M \arrow[d, "\kappa'"] \\
K'_\bullet\colon & 0 \arrow[r] & L_0 \times_{M_0} M_1 \arrow[r, "k'_2"'] & L_0 \oplus M_1 \arrow[r, "k'_1"'] & M_0 \arrow[r, "k'_0"'] & \operatorname{coker} f
\end{tikzcd} \tag{4.18}$$

is a lift of the canonical morphism $\kappa'\colon M \to$ $\operatorname{coker} f$*.*

Proof. The morphism $k'_0 := \kappa' m_0\colon M_0 \to \operatorname{coker} f$ is surjective because it is a composition of surjective morphisms. Consider the morphism $k'_1 := (-f_0, m_1)\colon L_0 \oplus M_1 \to M_0$. Because

$$k'_0 k'_1 = k' m_0(-f_0, m_1) = (-k' f l_0, m_0 m_1) = 0,$$

the morphism k'_1 factors through $\ker k'_0$. To show that k'_1 surjects onto $\ker k'_0$, let $x \in \ker k'_0$. Then $m_0(x) \in \ker \kappa' = \operatorname{im} f$, so there exists a $y \in L_0$ such that $f l_0(y) = m_0(x)$. Then $m_0(x - f_0(y)) = 0$, so there exist $x' \in M_1$ such that $x = -f_0(y) + m_1(x')$. This shows that k'_1 surjects onto $\ker k' m_0$. Then $L_0 \times_{M_0} M_1 = \ker(-f_0, m_1) = \ker k'_1$ is a free module, so (4.17) is a free resolution of $\operatorname{coker} f$. The diagram (4.18) clearly commutes. □

Combining the above statements, we obtain the following expression for a free resolution of the subquotient $\ker g / \operatorname{im} f$:

Theorem 4.1.6 (Homology resolution). *Let $L \xrightarrow{f} M \xrightarrow{g} N$ be maps of finitely generated modules with $gf = 0$, let $L_\bullet, M_\bullet$ and $N_\bullet$ be free resolutions of L, M and N of length* 2*, let $f_\bullet\colon L_\bullet \to M_\bullet$ and $g_\bullet\colon M_\bullet \to N_\bullet$ be lifts of f and g, and let ζ and η be as in Lemma 4.1.4. Then the sequence*

$$H_\bullet\colon \qquad L_0 \times_{M_0} M_1 \xrightarrow{h_2 := \begin{pmatrix} \mathrm{id}_{L_0} & 0 \\ 0 & \mathrm{id} \\ \multicolumn{2}{c}{\eta} \end{pmatrix}} L_0 \oplus M_1 \oplus N_2 \xrightarrow{h_1 := \left(\begin{smallmatrix} -f_0 & m_1 & 0 \\ -\zeta & g_1 & n_2 \end{smallmatrix}\right)} M_0 \times_{N_0} N_1 \to \frac{\ker g}{\operatorname{im} f} \tag{4.19}$$

is a free resolution of $\ker g / \operatorname{im} f$*, and*

$$\begin{tikzcd}
K_\bullet\colon \arrow[d, "q_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "k_2"] \arrow[d, "q_2"', "\left(\begin{smallmatrix} 0 \\ m_2 \end{smallmatrix}\right)"] & M_1 \oplus N_2 \arrow[r, "k_1"] \arrow[d, "q_1"', "\left(\begin{smallmatrix} 0 & 0 \\ \mathrm{id} & 0 \\ 0 & \mathrm{id} \end{smallmatrix}\right)"] & M_0 \times_{N_0} N_1 \arrow[r, "k_0"] \arrow[d, equal, "q_0"'] & \ker g \arrow[d, "q"'] \\
H_\bullet\colon & 0 \arrow[r] & L_0 \times_{M_0} M_1 \arrow[r, "h_2"'] & L_0 \oplus M_1 \oplus N_2 \arrow[r, "h_1"'] & M_0 \times_{N_0} N_1 \arrow[r, "h_0"'] & \frac{\ker g}{\operatorname{im} f}
\end{tikzcd} \tag{4.20}$$

is a lift of the canonical map $q\colon$ $\ker g \to \ker g / \operatorname{im} f$*.*

Proof. Applying Lemma 4.1.5 to the morphism $i_\bullet\colon J_\bullet \to K_\bullet$ from Lemma 4.1.4 gives that

$$0 \to L_0 \times_{K_0} K_1 \to L_0 \oplus \underbrace{M_1 \oplus N_2}_{K_1} \xrightarrow{h_1} \underbrace{M_0 \times_{N_0} N_1}_{K_0} \to \ker g/\operatorname{im} f \tag{4.21}$$

is a free resolution of $\ker g/\operatorname{im} f$, and that

$$\begin{tikzcd}
K_2 \arrow[r, "k_2"] \arrow[d, "q_2=\left(\begin{smallmatrix}0\\k_2\end{smallmatrix}\right)"] & K_1 \arrow[r, "k_1"] \arrow[d, "q_1=\left(\begin{smallmatrix}0\\\mathrm{id}\end{smallmatrix}\right)"] & K_0 \arrow[r, "k_0"] \arrow[d, equal, "q_0"] & \ker g \arrow[d, "q"'] \\
L_0 \times_{K_0} K_1 \arrow[r] & L_0 \oplus K_1 \arrow[r, "h_1"'] & M_0 \times_{N_0} N_1 \arrow[r, "h_0"'] & \frac{\ker g}{\operatorname{im} f}
\end{tikzcd} \tag{4.22}$$

is a lift of q. It remains to show that $L_0 \times_{K_0} K_1 \cong L_0 \times_{M_0} M_1$. Because $K_0 \subseteq M_0 \oplus N_1$, we get that

$$L_0 \times_{K_0} K_1 = L_0 \times_{M_0 \oplus N_1} K_1 = \ker\left(\begin{smallmatrix} f_0 & -m_1 & 0 \\ \eta & -g_1 & -n_2 \end{smallmatrix}\right) \subseteq L_0 \oplus M_1 \oplus N_2.$$

The morphism

$$h_2 := \begin{pmatrix} \mathrm{id} & 0 \\ 0 & \mathrm{id} \\ \multicolumn{2}{c}{\eta} \end{pmatrix}\colon L_0 \times_{M_0} M_1 \longrightarrow L_0 \oplus M_1 \oplus N_2. \tag{4.23}$$

is obviously injective. We show that $\operatorname{im} h_2 = L_0 \times_{K_0} K_1$. Since

$$\left(\begin{smallmatrix} f_0 & -m_1 & 0 \\ \zeta & -g_1 & -n_1 \end{smallmatrix}\right)\left(\begin{smallmatrix} \mathrm{id}_{L_0} & 0 \\ 0 & \mathrm{id}_{M_1} \end{smallmatrix}\right)\Big|_{H_2} = \left(\begin{smallmatrix} f_0 & -m_1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right)\Big|_{H_2} = 0, \tag{4.24}$$

the morphism h_2 factors through $L_0 \times_{K_0} K_1$, so $\operatorname{im} h_2 \subseteq L_0 \times_{K_0} K_1$. To show equality, let $(x, y, z) \in \ker h_1$. In particular, $(x, y) \in L_0 \times_{M_0} M_1 = \ker(f_0, -m_1)$. By definition of η, we get $n_2\eta\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right) = \zeta(x) - g_1(y) = n_2(z)$. By injectivity of n_2, we get $z = \eta\left(\begin{smallmatrix} x \\ y \end{smallmatrix}\right)$. This shows that h_2 maps $L_0 \times_{M_0} M_1$ isomorphically to $L_0 \times_{K_0} K_1$. □

Remark (Homology as a kernel). Let $L_\bullet, M_\bullet, N_\bullet, f_\bullet$ and $g_\bullet$ be as in Theorem 4.1.6. From Lemmas 4.1.3 and 4.1.5 we get free resolutions $K'_\bullet$ and $J'_\bullet$ of $\operatorname{coker} f$ and $\operatorname{im} g$, respectively. Analogously to Lemma 4.1.4, the morphisms $\kappa'_\bullet$ and $i'_\bullet$ in the commutative diagram

$$\begin{tikzcd}
M_\bullet\colon \arrow[d, "\kappa_\bullet"'] & 0 \arrow[r] & M_2 \arrow[r, "m_2"] \arrow[d, "\left(\begin{smallmatrix}0\\m_2\end{smallmatrix}\right)"] & M_1 \arrow[r, "m_1"] \arrow[d, "\left(\begin{smallmatrix}0\\\mathrm{id}_{M_1}\end{smallmatrix}\right)"'] & M_0 \arrow[r, "m_0"'] \arrow[d, equal] & M \arrow[d, "\kappa'"] \\
K'_\bullet\colon \arrow[d, "i'_\bullet"'] & 0 \arrow[r] & L_0 \times_{M_0} M_1 \arrow[r] \arrow[d, "\eta"] & L_0 \oplus M_1 \arrow[r, "(-f_0{,}m_1)"] \arrow[d, "\left(\begin{smallmatrix}-f_0 & m_1\\-\zeta & g_1\end{smallmatrix}\right)"] & M_0 \arrow[r, "k'_0"] \arrow[d, equal] & \operatorname{coker} f \arrow[d, "i'"'] \\
J'_\bullet\colon & 0 \arrow[r] & N_2 \arrow[r, "\left(\begin{smallmatrix}0\\n_2\end{smallmatrix}\right)"'] & M_0 \times_{N_0} N_1 \arrow[r, "(\mathrm{id}_{M_0}{,}0)"'] & M_0 \arrow[r] & \operatorname{im} g
\end{tikzcd} \tag{4.25}$$

are lifts of the canonical morphisms $\kappa'\colon M \to \operatorname{coker} f$ and $i'\colon \operatorname{coker} f \to \operatorname{im} g$, respectively. Applying Lemma 4.1.1 to $i'_\bullet$ gives the same free resolution of $\ker g/\operatorname{im} f = \operatorname{coker} i \cong \ker i'$ as in (4.19).

Recall from Remark 4.1.2 that the matrix used to define, for example, h_1, a priory defines a morphism $L_0 \oplus M_1 \oplus N_2 \to M_0 \oplus N_1$, and we mean by the notation that it factors uniquely through $M_0 \times_{N_0} N_1$. When it comes to actually computing graded matrices representing the morphisms h_2, h_1, we have to be a bit more precise. In particular, we have to distinguish a modules and their submodules more strictly, such as $M_0 \oplus N_1$ and its submodule $M_0 \times_{N_0} N_1$. Therefore, we formulate the following "pedantic" version of Theorem 4.1.6:

Theorem 4.1.7. *Let $L \xrightarrow{f} M \xrightarrow{g} N$ be maps of finitely generated modules with $gf = 0$, let $L_\bullet, M_\bullet$ and $N_\bullet$ be free resolutions of L, M and N of length 2, let $f_\bullet \colon L_\bullet \to M_\bullet$ and $g_\bullet \colon M_\bullet \to N_\bullet$ be lifts of f and g, and let ζ and η be as in Lemma 4.1.4. Then there exist morphisms h_0 and $h_1 = \left(\begin{smallmatrix}\psi\\ \eta\end{smallmatrix}\right)$ that make the diagram*

$$\begin{tikzcd}
0 \arrow[r, dashed] & L_0 \times_{M_0} M_1 \arrow[r, dashed, "h_2"] \arrow[d, hook, "\kappa'"'] & L_0 \oplus M_1 \oplus N_2 \arrow[r, dashed, "h_1"] \arrow[d, "\begin{pmatrix} \mathrm{id} & & \\ & \mathrm{id} & \\ & & n_2 \end{pmatrix}"] & M_0 \times_{N_0} N_1 \arrow[d, hook, "\kappa"] \\
 & L_0 \oplus M_1 \arrow[r, "\begin{pmatrix} \mathrm{id} & \\ & \mathrm{id} \\ \zeta & -g_1 \end{pmatrix}"'] \arrow[d, "{(f_0,-m_1)}"'] & L_0 \oplus M_1 \oplus N_1 \arrow[r, "\begin{pmatrix} -f_0 & m_1 & 0 \\ -\zeta & g_1 & \mathrm{id} \end{pmatrix}"'] & M_0 \oplus N_1 \arrow[d, "{(g_0,-n_1)}"] \\
 & M_0 & & N_0
\end{tikzcd}$$

of free modules commute, where ϕ and ψ are the canonical morphisms. Moreover, h_1 and h_2 form a free resolution of $\ker g / \operatorname{im} f$.

Let $C^\bullet$ be a cochain complex of not necessarily free A-modules. Assume we have fixed a free resolution

$$0 \to C_2^d \xrightarrow{c_2^d} C_1^d \xrightarrow{c_1^d} C_0^d$$

of C^d for each q, and a lift $\delta_\bullet^d \colon C_\bullet^{d-1} \to C_\bullet^d$ of the coboundary morphism δ^d. Then Theorem 4.1.7 states that there is a commutative diagram

$$\begin{tikzcd}
0 \arrow[r] & \underbrace{\ker(\delta_0^d, -c_1^d)}_{=:H^d(C^\bullet)_2} \arrow[r, dashed, "h_2^d"] \arrow[d, hook, "\kappa^{d-1}"'] & \underbrace{C_0^{d-1} \oplus C_1^d \oplus C_2^{d+1}}_{=:H^d(C^\bullet)_1} \arrow[r, dashed, "h_1^d"] \arrow[d, "\begin{pmatrix} \mathrm{id} & & \\ & \mathrm{id} & \\ & & c_2^{d+1} \end{pmatrix}"'] & \underbrace{\ker(\delta_0^{d+1}, -c_1^{d+1})}_{=:H^d(C^\bullet)_0} \arrow[d, "\kappa^d"] \\
 & C_0^{d-1} \oplus C_1^d \arrow[r, "\begin{pmatrix} \mathrm{id} & \\ & \mathrm{id} \\ \zeta & -\delta_1^{d+2} \end{pmatrix}"'] \arrow[d, "{(\delta_0^d,-c_1^d)}"'] & C_0^{d-1} \oplus C_1^d \oplus C_2^{d+1} \arrow[r, "\begin{pmatrix} -\delta_0^d & c_1^d & 0 \\ -\zeta & \delta_1^{d+1} & \mathrm{id} \end{pmatrix}"'] & C_0^d \oplus C_1^d \arrow[d, "{(\delta_0^{d+1},-c_1^{d+1})}"] \\
 & C_0^d & & C_0^{d+1}
\end{tikzcd} \tag{4.26}$$

whose first line is a free resolution $H^d(C^\bullet)_\bullet$ of $H^d(C^\bullet)$. Note that $H^d(C^\bullet)_2 = H^{d-1}(C^\bullet)_0$.

Remark (Theorem 4.1.7 as a mapping cone). If the coboundary morphisms $\delta_\bullet^d \colon C_\bullet^d \to C_\bullet^{d+1}$ satisfy $\delta_i^{d+1}\delta_i^d = 0$ for all i and q, then then is an easier way to prove that (4.26) is a free resolution of $H^d(C^\bullet)$. Namely, in this case, $0 \to C_2^\bullet \to C_1^\bullet \to C_0^\bullet$ is an exact sequence of cochain complexes. Recall that for any morphism f of cochain complexes, $\operatorname{coker} f$ is quasi-isomorphic to the mapping cone of f, which we denote by $\operatorname{cone} f$ [127, §1.5.8]. In particular, the iterated mapping cone

$$\hat{C}^\bullet := \operatorname{cone}(\operatorname{cone}(C_2^\bullet \to C_1^\bullet) \to C_0^\bullet))$$

is a cochain complex of free modules that satisfies $H^d(\hat{C}^\bullet) \cong H^d(C^\bullet)$ for all q, and

$$0 \to Z^{d-1}(\hat{C}^\bullet) \to \hat{C}^d \to Z^d(\hat{C}^\bullet) \tag{4.27}$$

is a free resolution of $H^d(\hat{C}^\bullet) \cong H^d(C^\bullet)$. Unravelling the definition shows that $\hat{C}^\bullet$ is the cochain complex of free modules $\hat{C}^d = C_0^d \oplus C_1^{d+1} \oplus C_2^{d+2}$ with coboundary morphism

$$\hat{\delta}^{d+1} := \begin{pmatrix} -\delta_0^{d+1} & c_1^{d+1} & 0 \\ 0 & \delta_1^{d+2} & -c_2^{d+2} \\ 0 & 0 & -\delta_2^{d+3} \end{pmatrix} \colon \hat{C}^d \to C'^{d+1}.$$

One may verify that the morphism

$$Z^d(\hat{C}^\bullet) = \ker \hat{d}^{d+1} \xrightarrow{\left(\begin{smallmatrix} \mathrm{id} & 0 & 0 \\ 0 & \mathrm{id} & 0 \end{smallmatrix}\right)} \ker(\delta^{d+1}, -c_1^{d+1}) = C_0^d \times_{C_0^{d+1}} C_1^{d+1}$$

is an isomorphism. Therefore, the free resolution (4.27) coincides with the one from Theorem 4.1.6.

4.2 Simplicial cohomology

In the following, we apply the above construction to the simplicial cochain complex of a finite, one-critically $\mathbf{Z}^2$-filtered simplicial complex $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}^2}$. As before, let $K = \operatorname{colim} K_*$. Note that Theorem 4.1.6 does not directly apply to the chain complexes $C^\bullet(K_*)$ and $C^\bullet(K_*, K)$, because the modules $C^d(K_*)$ and $C^d(K_*, K)$ are not finitely generated. Instead, we replace $C^\bullet(K_*)$ by an appropriate chain complex of finitely generated modules $R_z C^\bullet(K_*)$. Its homology then allows us to obtain $H^\bullet(K_*)$; see below.

Definition 4.2.1. For $z \in \mathbf{Z}^2$, let $R_z, E_z \colon \mathrm{Vec}^{\mathbf{Z}^2} \to \mathrm{Vec}^{\mathbf{Z}^2}$ be the exact functors with

$$\begin{aligned} (R_z M)_w &= \left\{\begin{smallmatrix} M_w & \text{if } -z \le w, \\ 0 & \text{otherwise} \end{smallmatrix}\right. & \qquad (R_z M)_{ww'} &= \left\{\begin{smallmatrix} M_{ww'} & \text{if } -z \le w \le w', \\ 0 & \text{otherwise} \end{smallmatrix}\right. \\ (E_z M)_w &= M_{-z \vee w} & (E_z M)_{ww'} &= M_{-z\vee w, z \vee w'}, \end{aligned}$$

called *restriction* and *extension* of M with respect to z.

As it will turn out later, using $-z$ rather than z in the definition is for convenience.

Example 4.2.2. For $z = 0$, the functor R_z and E_z maps M to

$$R_z M = \left[\begin{tikzcd} & \vdots & \vdots & \vdots & \vdots & \\ \cdots \arrow[r, equal] & 0 \arrow[r] \arrow[u, equal] & M_{0,2} \arrow[r] \arrow[u] & M_{1,2} \arrow[r] \arrow[u] & M_{2,2} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & 0 \arrow[r] \arrow[u, equal] & M_{0,1} \arrow[r] \arrow[u] & M_{1,1} \arrow[r] \arrow[u] & M_{2,1} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & 0 \arrow[r] \arrow[u, equal] & M_{0,0} \arrow[r] \arrow[u] & M_{1,0} \arrow[r] \arrow[u] & M_{2,0} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & 0 \arrow[r, equal] \arrow[u, equal] & 0 \arrow[r, equal] \arrow[u] & 0 \arrow[r, equal] \arrow[u] & 0 \arrow[r, equal] \arrow[u] & \cdots \\ & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \end{tikzcd}\right], \quad E_z M = \left[\begin{tikzcd} & \vdots & \vdots & \vdots & \vdots & \\ \cdots \arrow[r, equal] & M_{0,2} \arrow[r, equal] \arrow[u] & M_{0,2} \arrow[r] \arrow[u] & M_{1,2} \arrow[r] \arrow[u] & M_{2,2} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & M_{0,1} \arrow[r, equal] \arrow[u] & M_{0,1} \arrow[r] \arrow[u] & M_{1,1} \arrow[r] \arrow[u] & M_{2,1} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & M_{0,0} \arrow[r, equal] \arrow[u] & M_{0,0} \arrow[r] \arrow[u] & M_{1,0} \arrow[r] \arrow[u] & M_{2,0} \arrow[r] \arrow[u] & \cdots \\ \cdots \arrow[r, equal] & M_{0,0} \arrow[r, equal] \arrow[u, equal] & M_{0,0} \arrow[r] \arrow[u, equal] & M_{1,0} \arrow[r] \arrow[u, equal] & M_{2,0} \arrow[r] \arrow[u, equal] & \cdots \\ & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \vdots \arrow[u, equal] & \end{tikzcd}\right].$$

The idea behind these and the following definition is that we use R_z to "cut off" a non-finitely generated module and replace it by a finitely generated one. Namely if $M \in \mathrm{vec}^{\mathbf{Z}^2}$ is a pointwise finite dimensional module, then $R_z M$ is finitely generated. In particular, if F is a free module of finite rank, then $R_z F^*$ is finitely generated. Furthermore, if $M_w \cong M_{-z\vee w}$ whenever $-z \not\le w$, then $M \cong E_z R_z M$.

The idea behind this is that if M is non-finitely generated, but there is a $z \in Z^2$ such that everything "interesting" in M happens below z, we may replace M by the finitely generated module $R_z M$, do our calculations, and extend the result to $\mathbf{Z}^2$ using E_z.

For example, recall that the dual $F(w)^*$ of a free module $F(w)$ for some $w \in \mathbf{Z}^2$ has the components $(F(w)^*)_u = \left\{\begin{smallmatrix} k & \text{if } u \le -w \\ 0 & \text{otherwise,} \end{smallmatrix}\right.$. Therefore, it satisfies $(F(v)^*)_w \cong (F(v)^*)_{-z\vee w}$ whenever $w \not\ge -z$. We obtain:

Lemma 4.2.3. *If F is a free module such that $z > w$ for all $w \in \operatorname{rk} F$, then $F^* \cong E_z R_z F^*$.*

Fix a $z = (z_1, z_2) \in \mathbf{Z}^2$ as in the lemma. We use the following notation to write down a free resolution of $R_z F^*$.

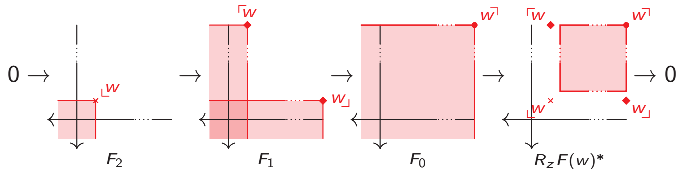


Figure 4.1: The free resolution (4.28) of the restricted module $R_wF(w)^*$, where $w < z$. Note that the diagrams are drawn with the structure morphisms pointing downward. The symbols •, ◆ and × indicate the grades of the generators, relations and 2-syzygies of this free resolution.

Definition 4.2.4. For $w = (w_1, w_2) \in \mathbf{Z}^2$ with $w < z$, define

$$\begin{aligned} {}^{\ulcorner}w &:= (-w_1+1, -z_2), & w^{\urcorner} &:= (-z_1, -z_2), \\ {}_{\llcorner}w &:= (-w_1+1, -w_2+1), & w_{\lrcorner} &:= (-z_1, -w_2+1). \end{aligned}$$

We call coordinates of the form ${}^{\ulcorner}w$, $w_{\lrcorner}$ or $w^{\urcorner}$ *infinite coordinates* and coordinates of the form ${}_{\llcorner}w$ *finite coordinates*. The infinite coordinates are the ones that contain at least one component from z. The definition is chosen such that the sequence

$$0 \to F({}_{\llcorner}w) \xrightarrow{\binom{1}{-1}} F(w_{\lrcorner}) \oplus F({}^{\ulcorner}w) \xrightarrow{(1,1)} F(w^{\urcorner}) \to R_zF(w)^* \tag{4.28}$$

is a minimal free resolution of $R_zF(w)^*$; see Figure 4.1. If F is a finite rank free module, then taking a direct sum of (4.28) gives a minimal free resolution of R_zF^*. Let $C_\bullet$ be a chain complex of finite rank free modules, and let $C^\bullet := (C_\bullet)^*$ be its dual cochain complex. If $z > w$ for all $w \in \operatorname{rk} C_\bullet$, then it follows from exactness of the functors R_z and E_z that

$$E_zH^d(R_zC^\bullet(K_*)) \cong E_zR_zH^d(C^\bullet(K_*)) \cong H^d(C^\bullet(K_*)).$$

Now, $R_zC^\bullet(K_*)$ is a cochain complex of finitely generated modules. Therefore, we can compute a minimal free resolution of $H^d(C^\bullet(K_*))$ by taking the cochain complex $R_zC^\bullet(K_*)$ of finitely generated modules, compute a minimal free resolution of its cohomology, and extend this to a free resolution of $H^d(C^\bullet(K_*))$, using the functor E_z.

Remark. The last step deserves a more detailed comment. On the level of resolutions, obtaining a minimal free resolution of $E_zH^d(R_zC^\bullet(K_*))$ from one of $H^d(R_zC^\bullet(K_*))$ involves some calculation. However, no calculation is needed if the ultimate goal is to obtain a minimal free resolution of $H^d(C^\bullet(K_*))$. We will explain this now. We assume that $H^\bullet(K_*)$ has bounded support. Let $F_\bullet \to H^d(R_zC^\bullet(K_*))$ be a minimal free resolution. Then, according to Corollary 3.2.11, $F'_\bullet := (\nu F_\bullet[2]\langle\epsilon\rangle)^*$ is a minimal free resolution of $H^d(R_zC^\bullet(K_*))^*$. It will contain some relations and 2-syzygies of infinite grades. For $v = (v_1, v_2)$ and $w = (w_1, w_2) \in \mathbf{Z}^2$, we say $v \ll w$ if $v_1 < w_1$ and $v_2 < w_2$. Let $G_\bullet$ and $G'_\bullet$ be the chain complex with

$$G_i := \bigoplus_{\substack{w \in \operatorname{rk} F'_i, \\ w \ll z}} F(w), \qquad G'_i := \bigoplus_{\substack{w \in \operatorname{rk} F'_i, \\ w \not\ll z}} F(w),$$

for $i = 0, 1, 2$, such that $F'_i = G_i \oplus G'_i$ for all i. One may check that $\partial^F_\bullet(G_\bullet) \subseteq G_\bullet$, which follows because $\operatorname{Hom}(F(v), (w)) = 0$ unless $v \geq w$. Therefore, $(G_\bullet, \partial^F_\bullet|_{G_\bullet})$ defines a chain complex. One may check that $G_\bullet$ is a minimal free resolution of $H_\bullet(K_*)$. Informally, $G_\bullet$ is obtained from $F'_\bullet$ by dropping all basis elements of infinite grade.

To compute a minimal free resolution of $H^d(R_zC^\bullet(K_*))$, we choose a free resolution $C^d_\bullet$ of each module $R_zC^\bullet(K_*)$. Then a free resolution of $R_zH^\bullet(C^\bullet) \cong H^\bullet(R_zC^\bullet)$ is given by (4.26), and (if $z > w$ for all $w \in \operatorname{rk} C_\bullet$) a free resolution of $H^\bullet(C^\bullet)$ is obtained by applying E_z to this. One can compute matrices representing this free resolution using Algorithms 6 and 7. In the following, we make this explicit.

4.2.1 Absolute cohomology

Let $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}^2}$ be one-critical and $z > g(\sigma)$ for all $\sigma \in K_*$. We construct graded matrices that represent the free resolutions $C^d_\bullet$ of each module $C^d(K_*)$ and the lifts $\delta^d_\bullet$ of the coboundary morphisms δ^d.

For algorithmic reasons, we enumerate the d-simplices of K_* reverse colexicographically by grade; that is, for each d, we assume that $K^d_* = \{\sigma_{d,1}, \dots, \sigma_{d,n_d}\}$ with

$$g(\sigma_{d,1}) \succeq_{\text{colex}} \cdots \succeq_{\text{colex}} g(\sigma_{d,n_d});$$

see Definition 2.4.2. Then the standard basis of $C_d(K_*) = \bigoplus_{i=1}^{n_d} F(g(\sigma_{di}))$ is colexicographically ordered. We do so because the procedure `Ker`() (Algorithm 6) requires its input matrix to have its columns colexicographically ordered. For each d, let π_d be the permutation of $\{1, \dots, n_d\}$ such that

$$g(\sigma_{d,\pi_d(1)}) \succeq_{\text{lex}} \cdots \succeq_{\text{lex}} g(\sigma_{d,\pi(n_d)}).$$

It is convenient to assume that the reordering π_d is *stable*; that is, $\pi_d(i) < \pi_d(j)$ for all $i < j$ with $g(\sigma_i) = g(\sigma_j)$.

We define the free modules

$$C^d_0 \coloneqq \bigoplus_{i=1}^{n_d} F(g^{\urcorner}(\sigma_{d,i})), \qquad C^d_1 \coloneqq C^d_{\ulcorner 1} \oplus C^d_{1 \lrcorner}, \qquad C^d_2 \coloneqq \bigoplus_{i=1}^{n_d} F{}_{\llcorner}g(\sigma_{d,i})), \tag{4.29}$$

where $C^d_{1\lrcorner} \coloneqq \bigoplus_{i=1}^{n_d} F(g_{\lrcorner}(\sigma_{d,i}))$ and $C^d_{\ulcorner 1} \coloneqq \bigoplus_{i=1}^{n_d} F({}^{\ulcorner}g(\sigma_{d,\pi_d(i)}))$. Note that $C^d(K_*)_2$, $C^d(K_*)_1$ and $C^d(K_*)_0$ have colexicographically ordered bases. Analogously to (4.28), these modules fit into a free resolution

$$0 \to C^d_2 \xrightarrow{c^d_2} C^d_1 \xrightarrow{c^d_1} C^d_0 \to R_z C^d(K_*) \tag{4.30}$$

of $R_z C^d(K_*)$. For each d, let E_d be the (ungraded) $n_d \times n_d$-unit matrix, and Π_d be the permutation matrix with entries $[\Pi_d]_{ij} = \left\{\begin{smallmatrix} 1 \text{ if } i = \pi^d(j) \\ 0 \text{ otherwise} \end{smallmatrix}\right.$. With respect to the bases (4.29), the resolution (4.30) is represented by the graded matrices c^d_2 and c^d_1 with $\mathsf{u}(c^d_2) = \left(\begin{smallmatrix} \Pi_d^{-1} \\ -E_d \end{smallmatrix}\right)$ and $\mathsf{u}(c^d_1) = (\Pi_d, E_d)$ and the appropriate row grades.

To obtain a graded matrix representing δ^d, let D^d be the (ungraded) matrix representing the coboundary morphism of $C^\bullet(K)$ with respect to the standard basis. Then δ^d lifts to the morphism

$$\begin{tikzcd}
C^d_\bullet\colon \arrow[d, "\delta^{d+1}_\bullet"'] & 0 \arrow[r] & C^d_2 \arrow[r, "c^d_2"] \arrow[d, "\delta^{d+1}_2"] & C^d_{\ulcorner 1} \oplus C^d_{1\lrcorner} \arrow[r, "c^d_1"] \arrow[d, "\delta^{d+1}_1"] & C^d_0 \arrow[r] \arrow[d, "\delta^{d+1}_0"] & R_z C^d(K_*) \arrow[d, "\delta^{d+1}"] \\
C^{d+1}_\bullet\colon & 0 \arrow[r] & C^{d+1}_2 \arrow[r, "c^{d+1}_2"'] & C^{d+1}_{\ulcorner 1} \oplus C^{d+1}_{1\lrcorner} \arrow[r, "c^{d+1}_1"'] & C^{d+1}_0 \arrow[r] & R_z C^{d+1}(K_*)
\end{tikzcd}$$

of free resolutions that is represented (with respect to the standard bases (4.29)) by the graded matrices with

$$\mathsf{u}(\delta^{d+1}_0) = D^d, \qquad \mathsf{u}(\delta^{d+1}_1) = \left(\begin{smallmatrix} \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 \\ 0 & D^{d+1} \end{smallmatrix}\right), \qquad \mathsf{u}(\delta^{d+1}_2) = D^d, \tag{4.31}$$

and the appropriate row and column grades.

Proposition 4.2.5 (Absolute cohomology resolution)**.** *For each d, there exists the dashed morphisms that make the diagram*

$$\begin{tikzcd}
 & 0 \arrow[d] & & 0 \arrow[d] \\
0 \arrow[r] & \underbrace{H^d(K_*)_2}_{\ker(\delta_0^d,-c_1^d)} \arrow[r, hook, dashed, "h_2^d"] \arrow[d, hook, "\kappa^{d-1}"] & \underbrace{H^d(K_*)_1}_{C_0^{d-1}\oplus C_1^d\oplus C_2^{d+1}} \arrow[r, dashed, "h_1^d"] \arrow[d, hook, "\begin{pmatrix} \mathrm{id} & 0 & 0 \\ 0 & \mathrm{id} & 0 \\ 0 & 0 & c_2^{d+1} \end{pmatrix}"] & \underbrace{H^d(K_*)_0}_{\ker(\delta_0^{d+1},-c_1^{d+1})} \arrow[d, hook, "\kappa^d"] \\
 & C_0^{d-1} \oplus C_1^d \arrow[r, "\begin{pmatrix} \mathrm{id} & 0 \\ 0 & \mathrm{id} \\ 0 & \delta_1^{d+1} \end{pmatrix}"'] \arrow[d, "{(\delta_0^d,-c_1^d)}"'] & C_0^{d-1} \oplus C_1^d \oplus C_1^{d+1} \arrow[r, "\begin{pmatrix} -\delta_0^d & c_1^d & 0 \\ 0 & \delta_1^{d+1} & \mathrm{id} \end{pmatrix}"'] & C_0^d \oplus C_1^{d+1} \arrow[d, "{(\delta_0^{d+1},-c_1^{d+1})}"] \\
 & C_0^d & & C_0^{d+1}
\end{tikzcd}$$

commute, such that the first row of the diagram then is a free resolution of $H^d(K_*)$.

Proof. This follows from Theorem 4.1.7. Since $\delta_i^{d+1}\delta_i^d = 0$ for all d, we may choose $\zeta^d = 0$. Actually, $\mathrm{Hom}(C_0^{d-1}, C_1^{d+1}) = 0$ because all basis elements of $C^{d-1}(K_*)_0$ are of grade z, while all basis elements of $C^{d+1}(K_*)_1$ are of grades strictly greater than z. Therefore, $\zeta^d = 0$ is the only possible choice for ζ^d. We obtain that $h_2^d = \left(\begin{smallmatrix} \kappa^{d-1} \\ \eta^d \end{smallmatrix}\right)$, where $\eta\colon H^d(K_*)_2 \to C_2^{d+1}$ satisfies $c_2^{d+1}\eta^d = (0, -\delta_1^{d+1})\kappa^{d-1}$. □

Using the above steps, we obtain the following strategy to compute a minimal free resolution of absolute cohomology:

4.2.6. *A pair of graded matrices h_2^d and h_1^d representing a minimal free resolution of $R_zH^d(K_*)$ can be computed in the following steps:*

(*i*) *Compute a graded matrix representing the kernel inclusion κ^d by applying Algorithm 6 to the graded matrix $(\delta_0^{d+1}, -c_1^{d+1})$. Analogously, compute the kernel inclusion κ^{d-1}.*

(*ii*) *Compute h_1^d by solving the linear system $\kappa^d h_1^d = \left(\begin{smallmatrix} -\delta_0^d & c_1^d & 0 \\ 0 & \delta_1^{d+1} & c_2^{d+1} \end{smallmatrix}\right)$, using Algorithm 7. Proposition 4.2.5 ensures that a solution exists.*

(*iii*) *Compute a graded matrix representing h_2^d. Commutativity of the above diagram implies that h_2^d has to be of the form $h_2^d = \left(\begin{smallmatrix} \kappa^{d-1} \\ \eta^{d+1} \end{smallmatrix}\right)$, where η^d satisfies $c_2^d\eta^d = (0, \delta_1^{d+1})\kappa^{d-1}$. Because*

$$c_2^{d+1}\eta^d = \begin{pmatrix} \Pi_{d+1}^{-1} \\ -E_{d+1} \end{pmatrix}\eta^d \quad \textit{and} \quad (0, \delta_1^{d+1})\kappa^{d-1} = \begin{pmatrix} 0 & \Pi_{d+1}^{-1}D^{d+1}\Pi_d & 0 \\ 0 & 0 & D^{d+1} \end{pmatrix}\kappa^{d-1},$$

the graded matrix η^d must have the underlying matrix

$$\mathsf{u}(\eta^d) = -(0, 0, D^{d+1})\,\mathsf{u}(\kappa^{d-1}) = (0, D^{d+1}\Pi_d, 0)\,\mathsf{u}(\kappa^{d-1}),$$

and the appropriate row and column grades to fit into the diagram. It follows from Theorem 4.1.7 that the thus defined graded matrix is valid.

(*iv*) *To obtain a minimal free resolution of $R_zH^d(K_*)$, apply Algorithm 4.*

4.2.2 Relative cohomology

Analogously, we construct a free resolution of $R_zH^{d+1}(K, K_*)$. If $H^\bullet(K) = 0$, then we obtain that $H^d(K_*) \cong H^{d+1}(K, K_*)$ for all d, which motivates the study of $R_zH^{d+1}(K, K_*)$, rather than $R_zH^d(K, K_*)$. We have

$$R_zC^d(K, K_*) = \left(\frac{R_z\Delta C_d(K)}{R_zC_d(K_*)}\right)^* = \ker(\rho\colon R_z\Delta C^d(K) \to R_zC^d(K_*)),$$

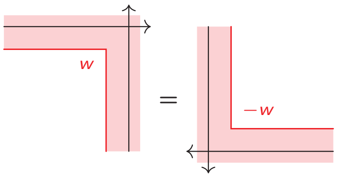


Figure 4.2: The relative cochain complex $C^\bullet(K, K_*)$ is direct sum of the modules $(\Delta k/F(w))^*$, where Δk is the constant module with $(\Delta k)_w = k$ for all w. As before, it is convenient to draw modules with downward pointing coordinate axes.

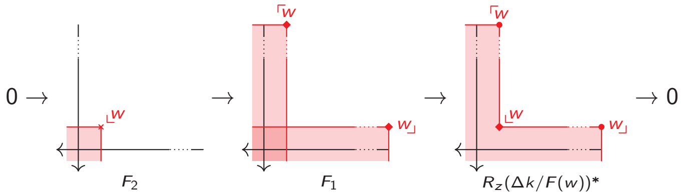


Figure 4.3: Free resolution of the module $R_z(\Delta k/F(w))^*$. The symbols • and ⬩ indicate the grades of the generators and relations.

where ρ is induced by the restriction map $\rho\colon \gamma \mapsto \gamma|_{K_*}$. The module

$$R_z\Delta C^d(K) = \bigoplus_{\sigma\in K} F(\bar{g}(\sigma)) = C^d_0$$

is free, so the restriction ρ lifts to a morphism

$$\begin{tikzcd}
\Delta C^d(K)\colon \arrow[d, "\rho_\bullet"'] & 0 \arrow[r] & 0 \arrow[r] \arrow[d] & 0 \arrow[r] \arrow[d] & R_z\Delta C^d(K) \arrow[d, equal] \\
C^d_\bullet\colon & 0 \arrow[r] & C^d_2 \arrow[r, "c^d_2"'] & C^d_1 \arrow[r, "c^d_1"'] & C^d_0
\end{tikzcd}$$

of free resolutions, where $C^d_\bullet$ is the free resolution from (4.30). Lemma 4.1.1 shows that

$$0 \to C^d_2 \xrightarrow{c^d_2} C^d_1 \to R_zC^d(K, K_*) \tag{4.32}$$

is a free resolution of $R_zC^\bullet(K, K_*)$, and

$$\begin{tikzcd}
0 \arrow[r] & C^d_{\ulcorner 1} \oplus C^d_{1\urcorner} \arrow[r, "c^d_1"] \arrow[d, "\delta^{d+1}_1"] & C^d_0 \arrow[r] \arrow[d, "\delta^{d+1}_0"] & R_zC^d(K, K_*) \\
0 \arrow[r] & C^{d+1}_{\ulcorner 1} \oplus C^{d+1}_{1\urcorner} \arrow[r, "c^{d+1}_1"'] & C^{d+1}_0 \arrow[r] & R_zC^{d+1}(K, K_*)
\end{tikzcd} \tag{4.33}$$

is a lift of the coboundary morphism $\delta^{d+1}\colon R_zC^d(K, K_*) \to R_zC^{d+1}(K, K_*)$. The morphisms δ^{d+1}_1 and δ^{d+1}_2 are the same as in as in (4.31).

Proposition 4.2.7 (Relative cohomology resolution)**.** *For each d, there exists morphisms h^{d+1}_1 and h^{d+1}_2 that make the diagram*

$$\begin{tikzcd}
\underbrace{H^{d+1}(K, K_*)_2}_{\ker c^{d+1}_1\delta^{d+1}_1} \arrow[r, hook, dashed, "h^{d+1}_2"] \arrow[d, hook, "\kappa^d"'] & \underbrace{H^{d+1}(K, K_*)_1}_{C^d_1\oplus C^{d+1}_2} \arrow[r, dashed, "h^{d+1}_1"] \arrow[d, "\begin{pmatrix} \mathrm{id} & 0 \\ 0 & c^{d+2}_2 \end{pmatrix}"] & \underbrace{H^{d+1}(K, K_*)_0}_{\ker c^{d+2}_1\delta^{d+2}_1} \arrow[d, hook, "\kappa^{d+1}"] \\
C^d_1 \arrow[r, "\begin{pmatrix} \mathrm{id} \\ \delta^{d+1}_1 \end{pmatrix}"'] \arrow[d, "c^{d+1}_1\delta^{d+1}_1"'] & C^d_1 \oplus C^{d+1}_1 \arrow[r, "(\delta^{d+1}_1{,} -\mathrm{id})"'] & C^{d+1}_1 \arrow[d, "c^{d+2}_1\delta^{d+2}_1"] \\
C^{d+1}_0 & & C^{d+2}_0
\end{tikzcd} \tag{4.34}$$

commute and form a free resolution of the module $R_zH^{d+1}(K, K_*)$.

Remark. We point to the fact that the morphisms c_i^d and δ_i^d are the same as in the previous section, while the other functions h_i^d and κ^d are not the same as in the previous section.

Proof of Proposition 4.2.7. Applying Theorem 4.1.7 to (4.33) shows that there exist morphisms $\tilde{h}_2$, $\tilde{h}_1$ that make the diagram

$$\begin{tikzcd}
\ker(\delta_1^{d+1}, c_2^{d+1}) \arrow[r, hook, dashed, "\tilde{h}_2^{d+1}"] \arrow[d, hook, "\tilde{\kappa}^d"'] & C_1^d \oplus C_2^{d+1} \arrow[r, dashed, "\tilde{h}_1^{d+1}"] \arrow[dl, equal] \arrow[dr, "\begin{pmatrix} -\delta_1^{d+1} & c_2^{d+1} \\ 0 & \delta_2^{d+2} \end{pmatrix}"'] & \ker(\delta_1^{d+2}, c_2^{d+2}) \arrow[d, hook, "\tilde{\kappa}^{d+1}"] \\
C_1^d \oplus C_2^{d+1} \arrow[d, "(\delta_1^{d+1}{,}c_2^{d+1})"] & & C_1^{d+1} \oplus C_2^{d+2} \arrow[d, "(\delta_1^{d+2}{,}c_2^{d+2})"] \\
C_1^{d+1} & & C_1^{d+2},
\end{tikzcd} \tag{4.35}$$

commute and form a free resolution of $R_zH^{d+1}(K, K_*)$. To show the statement of the proposition, consider the commutative diagram

$$\begin{tikzcd}
\ker(\delta_1^{d+2}, -c_2^{d+2}) \arrow[r, dashed, "\psi'"] \arrow[d, hook, "\tilde{\phi}^{d+1}"] & \ker c_1^{d+2} d_1^{d+2} \arrow[d, hook, "\phi^{d+1}"] \\
C_1^{d+1} \oplus C_2^{d+2} \arrow[r, "\psi=(\mathrm{id},0)"] \arrow[d, "(\delta_1^{d+2}{,}-c_2^{d+2})"'] & C_1^{d+1} \arrow[d, "c_1^{d+2}\delta_1^{d+2}"] \\
C_1^{d+2} \arrow[r, "c_1^{d+2}d_1^{d+2}"'] & C_0^{d+2},
\end{tikzcd}$$

One checks that the lower square commutes, so ψ induces the indicated morphism ψ'. Then ψ' is an isomorphism. To see this, we notice that ψ' is the morphism

$$\begin{aligned}
&\ker(\delta_1^{d+2}, c_2^{d+2}) \\
&= \ker\begin{pmatrix} \Pi_{d+2}^{-1} D^{d+2} \Pi_{d+1} & 0 & -\Pi_{d+2}^{-1} \\ 0 & D^{d+2} & E_{d+2} \end{pmatrix} \\
&= \{(\alpha, \beta, \gamma) \in C_{\ulcorner 1}^{d+1} \oplus C_{1\lrcorner}^{d+1} \oplus C_2^{d+2} \mid -\gamma = D^{d+2}\Pi_{d+1}\alpha = D^{d+2}\beta\} \\
&\overset{\psi'}{\to} \{(a, b) \in C_{\ulcorner 1}^{d+1} \oplus C_{1\lrcorner}^{d+1} \mid D^{d+2}\Pi_{d+1}\alpha = D^{d+2}\beta\} \\
&= \ker(D^{d+2}\Pi_{d+1}, D^{d+2}) \\
&= \ker c_1^{d+2} d_1^{d+2}.
\end{aligned}$$

An inverse to ψ' is given by $\psi' \colon \left(\begin{smallmatrix} \alpha \\ \beta \end{smallmatrix}\right) \mapsto \begin{pmatrix} \alpha \\ \beta \\ D^{d+2}\Pi_{d+1}\alpha \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ D^{d+2}\beta \end{pmatrix}$. The statement follows from applying this to (4.35). □

Using the above steps, we obtain the following strategy to compute a minimal free resolution of relative cohomology:

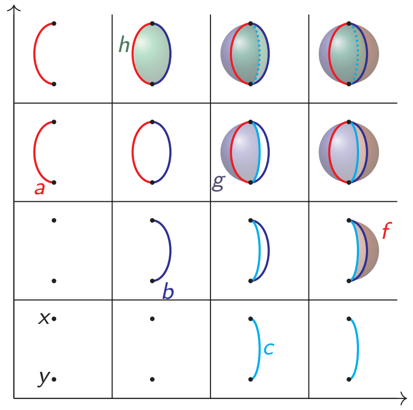


Figure 4.4: One-critical filtration K_* of a 2-sphere by two 0-cells x, y, three 1-cells a, b, c with $\partial a = \partial b = \partial c = x + y$ and three 2-cells f, g, h with $\partial f = b + c$, $\partial g = a + c$ and $\partial h = a + b$.

4.2.8. *A pair of graded matrices* h_2^{d+1}, h_1^{d+1} *representing the free resolution* (4.34) *of* $R_z H^{d+1}(K, K_*)$ *can be computed in the following steps:*

(*i*) *Compute a graded matrix* κ^{d+1} *representing a basis of* $\ker(c_1^{d+2} \delta_1^{d+2})$ *and a graded matrix* κ^d *representing a basis of* $\ker c_1^{d+1} \delta_1^{d+1}$, *using Algorithm 6.*

(*ii*) *Compute* h_1^{d+1} *by solving the linear system* $\kappa^{d+1} h_1^d = (d_1^{d+1}, -c_2^{d+1})$, *using Algorithm 7.*

(*iii*) *Compute* $h_2^{d+1} = \left(\begin{smallmatrix} \kappa^d \\ \eta^{d+1} \end{smallmatrix}\right)$, *where* $c_2^{d+1} \eta^{d+1} = \delta_1^{d+1} \phi^{d+1}$. *Because*

$$c_2^{d+1} \eta^{d+1} = \begin{pmatrix} \Pi_{d+1}^{-1} \\ -E_{d+1} \end{pmatrix} \eta^{d+1} \quad \textit{and} \quad \delta_1^{d+1} \phi^d = \begin{pmatrix} \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 \\ 0 & D^{d+1} \end{pmatrix} \phi^d,$$

the entries of η^{d+1} *satisfy*

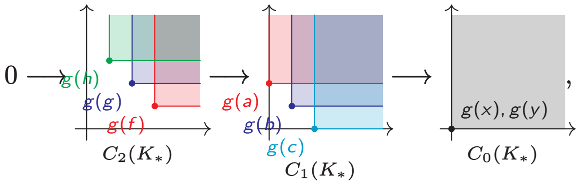
$$\mathsf{u}(\eta^{d+1}) = -(0, D^{d+1})\, \mathsf{u}(\kappa^d) = (D^{d+1} \Pi_d, 0)\, \mathsf{u}(\kappa^d).$$

(*iv*) *The obtained resolution can be converted to a minimal free resolution of* $R_z H^{d+1}(K, K_*)$ *using Algorithm 3.*

4.3 Example

In this section, we compute the relative and absolute two-parameter persistent (non-reduced) cohomology of the one-critically $\mathbf{Z}^2$-filtered cell complex K_* shown in Figure 4.4. We compute (co)homology with coefficients in $\mathbf{F}_2$. The complex K_* has the (absolute) simplicial chain complex

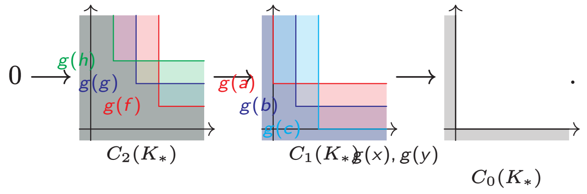
$C_\bullet(K_*)$:



which is a chain complex of free modules. A homogeneous basis is given by the simplices $\sigma \in K_*$. We use the symbol • to denote the grades of the basis elements of $C_\bullet$, which are precisely the grades of the simplices. Analogously, the relative chain complex $C_\bullet(K, K_*) = \Delta C_\bullet(K)/C(K_*)$

is the chain complex

$C_\bullet(K, K_*)$:

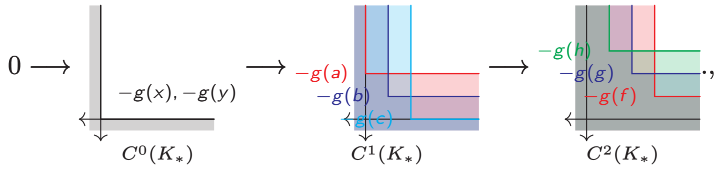


It is a chain complex of non-finitely generated modules.

4.3.1 Relative cohomology

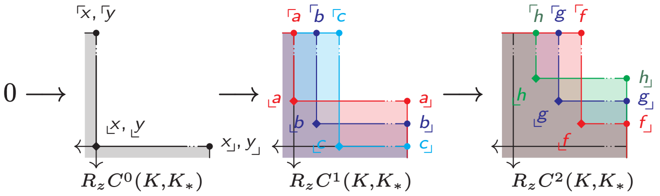
We start with the easier case of computing a free resolution of $C^\bullet(K, K_*)$. The relative cochain complex $C^\bullet(K, K_*)$ has the graded components $C^\bullet(K, K_*)_z = (C_\bullet(K, K_*)_{-z})^*$. Drawing the coordinate axes pointing downwards, we get

$C^\bullet(K, K_*)$:



We choose a fixed $z \in \mathbf{Z}^2$ with $z \geq g(\sigma)$ for all $\sigma \in K_*$, and compute the persistence module $R_z H^\bullet(K, K_*)$. This yields the cochain complex

$R_z C^\bullet(K, K_*)$:



of finitely generated modules. Each simplex $\sigma \in K_*$ corresponds to two generators $\ulcorner\sigma$ and $\sigma_\lrcorner$ of $R_z C^\bullet(K, K_*)$ of the grades $\ulcorner g(\sigma)$ and $g_\lrcorner(\sigma)$, indicated by the symbol •, and a relation $_\llcorner\sigma\colon \ulcorner\sigma \equiv \sigma_\lrcorner$ of grade $_\llcorner g(\sigma)$, indicated by the symbol ⧫. We decorate matrix row and columns by the basis elements of the codomain and domain they correspond to. The matrices

$$D^1 = \begin{array}{c|cc} & x & y \\ \hline c & 1 & 1 \\ b & 1 & 1 \\ a & 1 & 1 \end{array}, \qquad D^2 = \begin{array}{c|ccc} & c & b & a \\ \hline f & 1 & 1 & \\ g & 1 & & 1 \\ h & & 1 & 1 \end{array}$$

represent the coboundary morphisms of the cochain complex $C^\bullet(K)$. Let

$$\Pi_0 = \begin{array}{c|cc} & x & y \\ \hline x & 1 & \\ y & & 1 \end{array}, \qquad \Pi_1 = \begin{array}{c|ccc} & c & b & a \\ \hline a & & & 1 \\ b & & 1 & \\ c & 1 & & \end{array}, \qquad \Pi_2 = \begin{array}{c|ccc} & f & g & h \\ \hline h & & & 1 \\ g & & 1 & \\ f & 1 & & \end{array},$$

be the permutation matrices that represent the re-ordering of the simplices from reverse lexicographic ordering (columns) to reverse colexicographic ordering (rows).

0-Cohomology The module $R_z H^0(K, K_*)$ is generated by the kernel of the map

$$c_1^{d+2}\delta_1^{d+2} = C^1_2 \; \begin{array}{cc} C^0_{\ulcorner 1} & C^0_{1\lrcorner} \\ (\Pi_1^{-1}D^1 & \mid \; D^1) \end{array} = \begin{array}{c|cc|cc} & \ulcorner x & \ulcorner y & x_\lrcorner & y_\lrcorner \\ \hline a^\urcorner & 1 & 1 & 1 & 1 \\ b^\urcorner & 1 & 1 & 1 & 1 \\ c^\urcorner & 1 & 1 & 1 & 1 \end{array} \colon C^0_1 \to C^1_0.$$

The columns of this matrix are in colexicographic order. Applying Algorithm 6 to this matrix, we obtain a homogeneous basis

$$\kappa^0 = \left(\begin{array}{c|ccc} & \gamma_1 & \gamma_2 & \gamma_3 \\ \hline \ulcorner x & & 1 & 1 \\ \ulcorner y & & 1 & \\ \cline{2-4} x\lrcorner & 1 & & 1 \\ y\lrcorner & 1 & & \end{array}\right) \tag{4.36}$$

of $\ker c_1^{d+2}\delta_1^{d+2}$. The grade of each column is the join of the row grades of the non-zero entries in that column; that is

$$\begin{aligned} g(\gamma_1) &= g(x\lrcorner) \vee g(y\lrcorner) = g(x\lrcorner) = g(y\lrcorner), \\ g(\gamma_2) &= g(\ulcorner x) \vee g(\ulcorner y) = g(\ulcorner x) = g(\ulcorner y), \\ g(\gamma_3) &= g(\ulcorner x) \vee g(x\lrcorner) = g(\llcorner x) = g(\llcorner y). \end{aligned} \tag{4.37}$$

To get the relations of $R_zH^0(K, K_*)$, we have to compute a matrix representing the morphism h_1^0 in (4.34). The matrix (which we also denote by h_1^0) is the unique (because κ^0 is injective) solution

$$h_1^0 = \left(\begin{array}{c|cc} & \llcorner x & \llcorner y \\ \hline \gamma_1 & & 1 \\ \gamma_2 & & 1 \\ \gamma_3 & \textcircled{1} & 1 \end{array}\right) \quad \text{of the linear system} \quad \kappa^0 h_1^0 \overset{!}{=} \left(\begin{array}{c|cc} & \llcorner x & \llcorner y \\ \hline \ulcorner x & 1 & \\ \ulcorner y & & 1 \\ \cline{2-3} x\lrcorner & 1 & \\ y\lrcorner & & 1 \end{array}\right) = c_2^0.$$

Then h_1^0 is a free presentation of $R_zH^0(K, K_*)$. Since it is it injective, it also represents a free resolution (of length one) of $R_zH^0(K, K_*)$. To obtain a *minimal* free resolution, one has to eliminate all homological 1-balls from h_1^0. Recall that Algorithm 3 achieves this by identifying a maximal pairwise disjoint set of local pairs of the matrix h_1^0, and eliminates all non-zero entries in the local rows via left-to-right column additions, and deletes the local rows and columns. Applying Algorithm 3 to h_1^0 identifies the local pair corresponding to the circled entry, eliminates all non-zero entries in the same row by column additions, and deletes the corresponding row and column. We obtain the minimal (in the sense of Definition 2.3.21) graded matrix

$$\tilde{h}_1^0 = \left(\begin{array}{c|c} & \overbrace{\llcorner x + \llcorner y}^{r_1} \\ \hline \gamma_1 = x\lrcorner + y\lrcorner & 1 \\ \gamma_2 = \ulcorner x + \ulcorner y & 1 \end{array}\right),$$

which represents a minimal free presentation (and resolution)

$$0 \to \langle r_1 \rangle \xrightarrow{\tilde{h}_1^0} \langle \gamma_1, \gamma_2 \rangle \to R_zH^0(K, K_*) \tag{4.38}$$

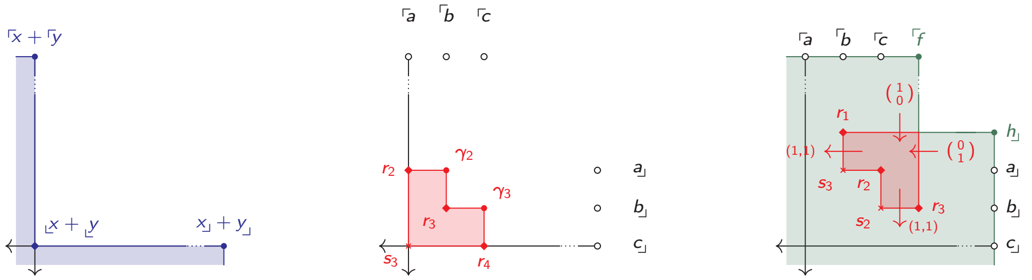
of $R_zH^0(K, K_*)$; see Figure 4.5a.

1-Cohomology Analogously, the module $R_zH^1(K, K_*)$ is generated by the basis

$$\kappa^1 := \left(\begin{array}{c|cccc} & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \hline \ulcorner c & & 1 & 1 & 1 \\ \ulcorner b & & & 1 & 1 \\ \ulcorner a & & & & 1 \\ \cline{2-5} a\lrcorner & 1 & 1 & 1 & \\ b\lrcorner & 1 & 1 & & \\ c\lrcorner & 1 & & & \end{array}\right) \quad \text{of} \quad \ker \underbrace{\left(\begin{array}{c|ccc|ccc} & \ulcorner c & \ulcorner b & \ulcorner a & a\lrcorner & b\lrcorner & c\lrcorner \\ \hline \ulcorner f & 1 & 1 & & & 1 & 1 \\ \ulcorner g & 1 & & 1 & 1 & & 1 \\ \ulcorner h & & 1 & 1 & 1 & 1 & \end{array}\right)}_{(D^1\Pi_2^{-1},\, D^1)}. \tag{4.39}$$

The matrix h_1^1 that represents the relations of $R_zH^1(K, K_*)$ is the unique solution

$$h_1^1 = \left(\begin{array}{c|cc|cc|ccc} & \ulcorner x & \ulcorner y & x\lrcorner & y\lrcorner & \llcorner a & \llcorner b & \llcorner c \\ \hline \gamma_1 & & & \textcircled{1} & 1 & & & 1 \\ \gamma_2 & & & & & & 1 & 1 \\ \gamma_3 & & & & & 1 & 1 & \\ \gamma_4 & \textcircled{1} & 1 & & & 1 & & \end{array}\right) \quad \text{of} \quad \kappa^1 h_1^1 \overset{!}{=} \left(\begin{array}{c|cc|cc|ccc} & \ulcorner x & \ulcorner y & x\lrcorner & y\lrcorner & \llcorner a & \llcorner b & \llcorner c \\ \hline \ulcorner c & 1 & 1 & & & & & 1 \\ \ulcorner b & 1 & 1 & & & & 1 & \\ \ulcorner a & 1 & 1 & & & 1 & & \\ \cline{2-8} a\lrcorner & & & 1 & 1 & 1 & & \\ b\lrcorner & & & 1 & 1 & & 1 & \\ c\lrcorner & & & 1 & 1 & & & 1 \end{array}\right). \tag{4.40}$$



(a) $R_z H^0(K, K_*)$ **(b)** $R_z H^1(K, K_*)$ **(c)** $R_z H^2(K, K_*)$

Figure 4.5: The free resolutions (4.38), (4.42) and (4.43) of $R_z H^\bullet(K, K_*)$ for the filtered simplicial complex K_* from Figure 4.4. The symbols •, ◆ and × denote the grades of the 0-, 1- and 2-syzygies of the modules; the symbols ○ indicate the grades of the relevant generators ${}^\ulcorner\sigma$ and $\sigma_\lrcorner$ of $C^\bullet(K, K_*)$. The module $R_z H^2(K, K_*)$ is indecomposable, because it contains a indecomposable quiver representation.

A matrix h_2^1 that represents a basis of $H^1(K, K_*)_2$ can be obtained by forming the matrix product

$$h_2^1 := \begin{array}{cc} & \begin{array}{cc} C^0_{\ulcorner 1} & C^0_{1\lrcorner} \end{array} \\ \begin{array}{c} C^0_{\ulcorner 1} \\ C^0_{1\lrcorner} \\ C^1_2 \end{array} & \left(\begin{array}{c|c} 1 & \\ \hline & 1 \\ \hline & D^1 \end{array}\right) \end{array} \cdot \kappa^0 = \begin{array}{cc} & \begin{array}{ccc} \overbrace{x_\lrcorner+y_\lrcorner}^{s_1} & \overbrace{{}^\ulcorner x+{}^\ulcorner y}^{s_2} & \overbrace{{}^\ulcorner x+x_\lrcorner}^{s_3} \end{array} \\ \begin{array}{c} {}^\ulcorner x \\ {}^\ulcorner y \\ x_\lrcorner \\ y_\lrcorner \\ {}_\llcorner a \\ {}_\llcorner b \\ {}_\llcorner c \end{array} & \left(\begin{array}{ccc} & \textcircled{1} & 1 \\ & 1 & \\ \hline 1 & & 1 \\ \textcircled{1} & & \\ \hline & & 1 \\ & & 1 \\ & & 1 \end{array}\right) \end{array} \tag{4.41}$$

where κ^0 is the matrix representing the basis of $R_z H^1(K, K_*)_2 \cong R_z H^0(K, K_*)_0$ from (4.36). Here, we use the symbols s_i to distinguish the columns of κ^0 from those of κ^1. Note that the left matrix in (4.41) (specifically, its bottom right block) is not valid in general. Nevertheless, it follows from Theorem 4.1.6 that the resulting matrix h_2^1 is valid. Then h_1^1 and h_2^1 represent a free resolution of $R_z H^1(K, K_*)$.

To obtain a minimal free resolution, we apply Algorithm 3 first to h_2^1. Note that the rows of h_2^1 (that is, the columns of h_1^1) are not lexicographically or colexicographically ordered. Nevertheless, the block shape of the right side in (4.40) ensures that the rows of h_1^1 are in non-descending order, as required by Algorithm 3. Therefore, reordering the rows of h_1^1 and the columns of h_2^1 is not necessary. Algorithm 3 identifies local pairs of h_2^1 corresponding to the circled entries in (4.41). We get triple $(\tilde{h}_2^1, r, c) = \texttt{Minimize}(h_2^1)$, where $\tilde{h}_2^1$ is a minimal graded matrix, and r and c contain the non-local row and column indices of h_2^1; that is, the indices that do not belong to a local pair of h_2^1. We obtain the matrices

$$\tilde{h}_2^1 = \begin{array}{cc} & s_3 \\ \begin{array}{c} {}^\ulcorner x \\ x_\lrcorner \\ {}_\llcorner a \\ {}_\llcorner b \\ {}_\llcorner c \end{array} & \left(\begin{array}{c} 1 \\ \hline 1 \\ \hline 1 \\ 1 \\ 1 \end{array}\right) \end{array}, \qquad \tilde{h}_1^1 := ([h_1^1]_{ij})_{j \in r} = \begin{array}{cc} & \begin{array}{c|c|ccc} {}^\ulcorner x & x_\lrcorner & {}_\llcorner a & {}_\llcorner b & {}_\llcorner c \end{array} \\ \begin{array}{c} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{array} & \left(\begin{array}{c|c|ccc} & \textcircled{1} & & & 1 \\ & & & 1 & 1 \\ & & 1 & 1 & \\ \textcircled{1} & & 1 & & \end{array}\right) \end{array}.$$

We apply Algorithm 3 to $\tilde{h}_1^1$, which identifies the circled local entries. With $(\tilde{h}_1'^1, r', c') = \texttt{Minimize}(\tilde{h}_1^1)$, we get that the matrices

$$\tilde{h}_2'^1 := ([\tilde{h}_2^1]_{ij})_{i \in c'} = \begin{array}{cc} & \overbrace{{}^\ulcorner x+x_\lrcorner}^{s_3} \\ \begin{array}{c} {}_\llcorner a \\ {}_\llcorner b \\ {}_\llcorner c \end{array} & \left(\begin{array}{c} 1 \\ 1 \\ 1 \end{array}\right) \end{array}, \quad \text{and} \quad \tilde{h}_1'^1 = \begin{array}{cc} & \begin{array}{ccc} \overbrace{{}_\llcorner a+{}^\ulcorner x}^{r_2} & \overbrace{{}_\llcorner b}^{r_3} & \overbrace{{}_\llcorner c+x_\lrcorner}^{r_4} \end{array} \\ \begin{array}{c} \gamma_2 = a_\lrcorner+b_\lrcorner+{}^\ulcorner c \\ \gamma_3 = a_\lrcorner+{}^\ulcorner b+{}^\ulcorner c \end{array} & \left(\begin{array}{ccc} & 1 & 1 \\ 1 & 1 & \end{array}\right) \end{array}$$

represent the minimal free resolution

$$0 \to \langle s_3 \rangle \xrightarrow{\tilde{h}_2'^1} \langle r_2, r_3, r_4 \rangle \xrightarrow{\tilde{h}_1'^1} \langle \gamma_2, \gamma_3 \rangle \to R_z H^1(K, K_*) \tag{4.42}$$

of $R_zH^1(K, K_*)$, see Figure 4.5b.

2-Cohomology Because K_* has no 3-simplicies, $R_zH^2(K, K_*)$ is generated by the generators ${}^{\ulcorner}f$, ${}^{\ulcorner}g$, ${}^{\ulcorner}h$, $f_{\lrcorner}$, $g_{\lrcorner}$ and $h_{\lrcorner}$ of C^2_1. The relations of $R_zH^2(K, K_*)$ are represented by the matrix

$$h^2_1 = \begin{array}{c} \\ C^2_{\ulcorner 1} \\ C^2_{1\lrcorner} \end{array} \begin{array}{c|c|c} C^1_{\ulcorner 1} & C^1_{1\lrcorner} & C^2_2 \\ \Pi_2^{-1}D^2\Pi_1 & & \Pi_2^{-1} \\ \hline & D^2 & E_2 \end{array} = \begin{array}{c} \\ {}^{\ulcorner}f \\ {}^{\ulcorner}g \\ {}^{\ulcorner}h \\ h_{\lrcorner} \\ g_{\lrcorner} \\ f_{\lrcorner} \end{array} \begin{array}{ccc|ccc|ccc} {}^{\ulcorner}c & {}^{\ulcorner}b & {}^{\ulcorner}a & a_{\lrcorner} & b_{\lrcorner} & c_{\lrcorner} & {}_{\llcorner}h & {}_{\llcorner}g & {}_{\llcorner}f \\ 1 & 1 & & & & & & & 1 \\ 1 & & 1 & & & & & 1 & \\ & 1 & 1 & & & & 1 & & \\ \hline & & & 1 & 1 & & 1 & & \\ & & & 1 & & 1 & & 1 & \\ & & & & 1 & 1 & & & 1 \end{array}.$$

Analogously to (4.41), we obtain the 2-syzygies (viz. the kernel of h^2_1) by forming the product

$$h^2_2 := \begin{array}{c} \\ C^1_{\ulcorner 1} \\ C^1_{1\lrcorner} \\ C^2_2 \end{array} \begin{pmatrix} \begin{array}{c|c} C^1_{\ulcorner 1} & C^1_{1\lrcorner} \\ E_1 & \\ \hline & E_2 \\ \hline & D^2 \end{array} \end{pmatrix} \cdot \kappa^1 = \begin{array}{c} \\ \\ {}^{\ulcorner}c \\ {}^{\ulcorner}b \\ {}^{\ulcorner}a \\ a_{\lrcorner} \\ b_{\lrcorner} \\ c_{\lrcorner} \\ {}_{\llcorner}h \\ {}_{\llcorner}g \\ {}_{\llcorner}f \end{array} \begin{array}{cccc} s_1 & s_2 & s_3 & s_4 \\ \overbrace{a_{\lrcorner}+b_{\lrcorner}+c_{\lrcorner}} & \overbrace{a_{\lrcorner}+b_{\lrcorner}+{}^{\ulcorner}c} & \overbrace{a_{\lrcorner}+{}^{\ulcorner}b+{}^{\ulcorner}c} & \overbrace{{}^{\ulcorner}a+{}^{\ulcorner}b+{}^{\ulcorner}c} \\ & 1 & 1 & 1 \\ & & 1 & 1 \\ & & & \textcircled{1} \\ \hline 1 & & & \\ 1 & 1 & & \\ \textcircled{1} & 1 & 1 & \\ \hline & & 1 & \\ & 1 & 1 & \\ & 1 & & \end{array}$$

with the matrix κ^1_0 from (4.39). Applying Algorithm 3 to h^2_2 identifies the circled local entries. Analogously to the above, we obtain

$$\tilde{h}^2_2 = \begin{array}{c} \\ {}^{\ulcorner}c \\ {}^{\ulcorner}b \\ a_{\lrcorner} \\ b_{\lrcorner} \\ {}_{\llcorner}f \\ {}_{\llcorner}g \\ {}_{\llcorner}h \end{array} \begin{array}{cc} s_2 & s_3 \\ 1 & 1 \\ & 1 \\ \hline 1 & 1 \\ 1 & \\ \hline & 1 \\ 1 & 1 \\ 1 & \end{array}, \qquad \tilde{h}^2_1 = \begin{array}{c} \\ {}^{\ulcorner}f \\ {}^{\ulcorner}g \\ {}^{\ulcorner}h \\ h_{\lrcorner} \\ g_{\lrcorner} \\ f_{\lrcorner} \end{array} \begin{array}{cc|cc|ccc} {}^{\ulcorner}c & {}^{\ulcorner}b & a_{\lrcorner} & b_{\lrcorner} & {}_{\llcorner}h & {}_{\llcorner}g & {}_{\llcorner}f \\ 1 & 1 & & & & & 1 \\ \textcircled{1} & & & & & 1 & \\ & \textcircled{1} & & & 1 & & \\ \hline & & 1 & 1 & 1 & & \\ & & \textcircled{1} & & & 1 & \\ & & & \textcircled{1} & & & 1 \end{array}.$$

Applying Algorithm 3 to $\tilde{h}^2_1$ identifies the circled local entries. We get that the matrices

$$\tilde{h}'^2_2 = \begin{array}{c} \\ \\ {}_{\llcorner}h \\ {}_{\llcorner}g \\ {}_{\llcorner}f \end{array} \begin{array}{cc} s_2 & s_3 \\ \overbrace{a_{\lrcorner}+b_{\lrcorner}+{}^{\ulcorner}c} & \overbrace{a_{\lrcorner}+{}^{\ulcorner}b+{}^{\ulcorner}c} \\ & 1 \\ 1 & 1 \\ 1 & \end{array}, \qquad \tilde{h}'^2_1 = \begin{array}{c} \\ \\ {}^{\ulcorner}f \\ h_{\lrcorner} \end{array} \begin{array}{ccc} r_1 & r_2 & r_3 \\ \overbrace{{}_{\llcorner}f+b_{\lrcorner}} & \overbrace{{}_{\llcorner}g+{}^{\ulcorner}a+c_{\lrcorner}} & \overbrace{{}_{\llcorner}h+{}^{\ulcorner}b} \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$

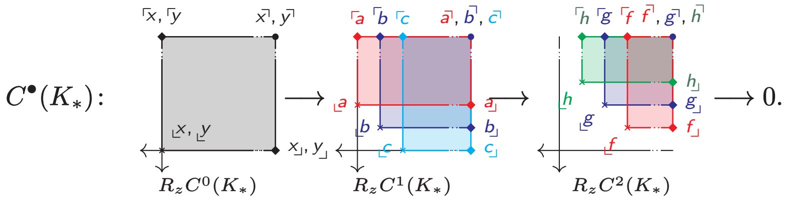
represent the minimal free resolution

$$0 \to \langle s_2, s_3 \rangle \xrightarrow{\tilde{h}'^2_2} \langle r_1, r_2, r_3 \rangle \xrightarrow{\tilde{h}'^2_1} \langle {}^{\ulcorner}f, h_{\lrcorner} \rangle \to R_zH^2(K, K_*) \tag{4.43}$$

of $R_zH^2(K, K_*)$, see Figure 4.5c.

4.3.2 Absolute cohomology

Analogously, we compute a minimal free resolutions of $R_zH^\bullet(K_*)$, using the restricted absolute cochain complex


$C^\bullet(K_*)$:
${}^{\ulcorner}x$, ${}^{\ulcorner}y$
$x^{\urcorner}$, $y^{\urcorner}$
${}_{\llcorner}x$, ${}_{\llcorner}y$
$x_{\lrcorner}$, $y_{\lrcorner}$
$R_zC^0(K_*)$
${}^{\ulcorner}a$ ${}^{\ulcorner}b$ ${}^{\ulcorner}c$
$a^{\urcorner}$, $b^{\urcorner}$, $c^{\urcorner}$
${}_{\llcorner}a$
$a_{\lrcorner}$
${}_{\llcorner}b$
$b_{\lrcorner}$
${}_{\llcorner}c$
$c_{\lrcorner}$
$R_zC^1(K_*)$
${}^{\ulcorner}h$ ${}^{\ulcorner}g$ ${}^{\ulcorner}f$
$f^{\urcorner}$, $g^{\urcorner}$, $h^{\urcorner}$
$h_{\lrcorner}$
${}_{\llcorner}h$
$g_{\lrcorner}$
${}_{\llcorner}g$
$f_{\lrcorner}$
${}_{\llcorner}f$
$R_zC^2(K_*)$
$\longrightarrow 0.$

Each d-simplex $\sigma \in K_*$ constributes one generator $\sigma\urcorner$ marked by •, two relations $\ulcorner\sigma, \sigma\lrcorner\colon \sigma\urcorner \equiv 0$ marked by ◆, and the 2-syzygy $\llcorner\sigma\colon \ulcorner\sigma \equiv \sigma\lrcorner$ marked by ×.

0-Homology From Proposition 4.2.5, we obtain that a generating system of $R_zH^0(K_*)$ is represented by the basis

$$\kappa^0 = \begin{array}{c|ccccc} & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 \\ \hline x\urcorner & 1 & 1 & 1 & 1 & 1 \\ y\urcorner & 1 & & & & \\ \hline \ulcorner c & & & 1 & 1 & 1 \\ \ulcorner b & & & & 1 & 1 \\ \ulcorner a & & & & & 1 \\ \hline a\lrcorner & & 1 & 1 & 1 & \\ b\lrcorner & & 1 & 1 & & \\ c\lrcorner & & 1 & & & \end{array} \quad \text{of} \quad \ker(\delta_0^1, c_1^1) = \ker \begin{array}{c|cc|ccc|ccc} & x\urcorner & y\urcorner & \ulcorner c & \ulcorner b & \ulcorner a & a\lrcorner & b\lrcorner & c\lrcorner \\ \hline a\urcorner & 1 & 1 & & & 1 & 1 & & \\ b\urcorner & 1 & 1 & & 1 & & & 1 & \\ c\urcorner & 1 & 1 & 1 & & & & & 1 \end{array}. \tag{4.44}$$

The relations of $R_zH^0(K_*)$ are represented by the unique solution

$$h_1^0 = \begin{array}{c|cc|cc|ccc} & \ulcorner x & \ulcorner y & x\lrcorner & y\lrcorner & \llcorner a & \llcorner b & \llcorner c \\ \hline \gamma_1 & & 1 & & 1 & & & \\ \gamma_2 & & & \textcircled{1} & 1 & & & 1 \\ \gamma_3 & & & & & & 1 & 1 \\ \gamma_4 & & & & & 1 & 1 & \\ \gamma_5 & \textcircled{1} & 1 & & & 1 & & \end{array} \quad \text{of} \quad \kappa^0 h_1^0 \overset{!}{=} \begin{array}{c|cc|cc|ccc} & \ulcorner x & \ulcorner y & x\lrcorner & y\lrcorner & \llcorner a & \llcorner b & \llcorner c \\ \hline x\urcorner & 1 & & 1 & & & & \\ y\urcorner & & 1 & & 1 & & & \\ \hline \ulcorner c & 1 & 1 & & & & & 1 \\ \ulcorner b & 1 & 1 & & & & 1 & \\ \ulcorner a & 1 & 1 & & & 1 & & \\ \hline a\lrcorner & & & 1 & 1 & 1 & & \\ b\lrcorner & & & 1 & 1 & & 1 & \\ c\lrcorner & & & 1 & 1 & & & 1 \end{array}. \tag{4.45}$$

The relations of $R_zH^0(K_*)$ (viz. the kernel of h_1^1) are represented by the matrix product

$$h_1^0 = \begin{array}{c|c|c} & C^0_{\ulcorner 1} & C^1_{1\lrcorner} \\ \hline C^0_{\ulcorner 1} & E_0 & \\ \hline C^0_{1\lrcorner} & & E_0 \\ \hline C^1_2 & 0 & D^1 \end{array} \cdot \underbrace{\begin{array}{c|cc} & \llcorner x & \llcorner y \\ \hline \ulcorner x & 1 & \\ \ulcorner y & & 1 \\ \hline x\lrcorner & 1 & \\ y\lrcorner & & 1 \end{array}}_{c_2^0} = \begin{array}{c|cc} & \llcorner x & \llcorner y \\ \hline \ulcorner x & 1 & \\ \ulcorner y & & 1 \\ \hline x\lrcorner & 1 & \\ y\lrcorner & & 1 \\ \hline \llcorner a & 1 & 1 \\ \llcorner b & 1 & 1 \\ \llcorner c & 1 & 1 \end{array}.$$

The matrix c_2^0 has no local entries. Applying Algorithm 3 identifies the local pairs corresponding to the circled entries in (4.45) We obtain that the matrices

$$\tilde{h}_2^0 = \begin{array}{c|cc} & \llcorner x & \llcorner y \\ \hline r_1 & & 1 \\ \hline r_2 & & 1 \\ \hline \llcorner a & 1 & 1 \\ \llcorner b & 1 & 1 \\ \llcorner c & 1 & 1 \end{array}, \qquad \tilde{h}_1^0 = \begin{array}{c|ccccc} & \overbrace{\ulcorner x + \ulcorner y}^{r_1} & \overbrace{x\lrcorner + y\lrcorner}^{r_2} & \llcorner a & \llcorner b & \llcorner c \\ \hline \gamma_1 = x\urcorner + y\urcorner & 1 & 1 & & & \\ \gamma_3 = x\urcorner + a\lrcorner + b\lrcorner + \ulcorner c & & & & 1 & 1 \\ \gamma_4 = x\urcorner + a\lrcorner + \ulcorner b + \ulcorner c & & & 1 & 1 & \end{array}$$

represent the minimal free resolution

$$0 \to \langle \llcorner x, \llcorner y \rangle \xrightarrow{\tilde{h}_2^0} \langle \ulcorner y, y\lrcorner, \llcorner a, \llcorner b, \llcorner c \rangle \xrightarrow{\tilde{h}_1^0} \langle \gamma_1, \gamma_3, \gamma_4 \rangle \to R_zH^0(K_*)$$

of $R_zH^0(K_*)$. We may perform invertible column operations on $\tilde{h}_2^0$ without changing the result. Thus, replacing $\tilde{h}_2^0$ by

$$\tilde{h}_2'^0 := \begin{array}{c|cc} & \llcorner x & \llcorner x + \llcorner y \\ \hline r_1 & & 1 \\ r_2 & & 1 \\ \hline \llcorner a & 1 & \\ \llcorner b & 1 & \\ \llcorner c & 1 & \end{array},$$

we see that

$$aR_zH^0(K_*) \cong \frac{\langle \gamma_1 \rangle}{\langle r_1, r_2 \rangle} \oplus \frac{\langle \gamma_3, \gamma_4 \rangle}{\langle \llcorner a, \llcorner b, \llcorner c \rangle}. \tag{4.46}$$

see Figure 4.6a.

1-Cohomology Analogously, we obtain the generators or $R_zH^1(K_*)$ by computing the basis

$$\kappa^1 = \left(\begin{array}{c|cccccc}
 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 \\
a^{\urcorner} & 1 & 1 & & & 1 & \\
b^{\urcorner} & 1 & & 1 & 1 & 1 & 1 \\
c^{\urcorner} & 1 & & & & & \\ \hline
{}^{\ulcorner}f & & & & 1 & 1 & 1 \\
{}^{\ulcorner}g & & & & & 1 & \\
{}^{\ulcorner}h & & & & & & 1 \\ \hline
h_{\lrcorner} & & 1 & 1 & 1 & & \\
g_{\lrcorner} & & 1 & & & & \\
f_{\lrcorner} & & & 1 & & &
\end{array}\right) \quad \text{of} \quad \ker \left(\begin{array}{c|ccc|ccc|ccc}
 & a^{\urcorner} & b^{\urcorner} & c^{\urcorner} & {}^{\ulcorner}f & {}^{\ulcorner}g & {}^{\ulcorner}h & h_{\lrcorner} & g_{\lrcorner} & f_{\lrcorner} \\
h^{\urcorner} & 1 & 1 & & & & 1 & 1 & & \\
g^{\urcorner} & 1 & & 1 & & 1 & & & 1 & \\
f^{\urcorner} & & 1 & 1 & 1 & & & & & 1
\end{array}\right).$$

The relations of $R_zH^1(K_*)$ are represented by the unique solution

$$h_1^1 = \left(\begin{array}{c|cc|ccc|ccc|ccc}
 & x^{\urcorner} & y^{\urcorner} & {}^{\ulcorner}c & {}^{\ulcorner}b & {}^{\ulcorner}a & a_{\lrcorner} & b_{\lrcorner} & c_{\lrcorner} & {}_{\llcorner}h & {}_{\llcorner}g & {}_{\llcorner}f \\
\gamma_1 & 1 & 1 & 1 & & & & & & & & \\
\gamma_2 & & & & & & 1 & & 1 & & 1 & \\
\gamma_3 & & & & & & & 1 & 1 & & & 1 \\
\gamma_4 & & & & & & & & & 1 & 1 & 1 \\
\gamma_5 & & & 1 & & 1 & & & & & 1 & \\
\gamma_6 & & & & 1 & 1 & & & & 1 & &
\end{array}\right) \quad \text{of} \quad \kappa^1 h_1^1 \overset{!}{=} \left(\begin{array}{c|cc|ccc|ccc|ccc}
 & x^{\urcorner} & y^{\urcorner} & {}^{\ulcorner}c & {}^{\ulcorner}b & {}^{\ulcorner}a & a_{\lrcorner} & b_{\lrcorner} & c_{\lrcorner} & {}_{\llcorner}h & {}_{\llcorner}g & {}_{\llcorner}f \\
a^{\urcorner} & 1 & 1 & & & 1 & 1 & & & & & \\
b^{\urcorner} & 1 & 1 & & 1 & & & 1 & & & & \\
c^{\urcorner} & 1 & 1 & 1 & & & & & 1 & & & \\ \hline
{}^{\ulcorner}f & & & 1 & 1 & & & & & & & 1 \\
{}^{\ulcorner}g & & & 1 & & 1 & & & & & 1 & \\
{}^{\ulcorner}h & & & & 1 & 1 & & & & 1 & & \\ \hline
h_{\lrcorner} & & & & & & 1 & 1 & & 1 & & \\
g_{\lrcorner} & & & & & & 1 & & 1 & & 1 & \\
f_{\lrcorner} & & & & & & & 1 & 1 & & & 1
\end{array}\right),$$

and the 2-syzygies of $R_zH^1(K_*)$ are represented by the matrix product

$$h_2^1 := \left(\begin{array}{c|c|c|c}
 & C_0^0 & C^1_{\ulcorner 1} & C^1_{1\lrcorner} \\
C_0^0 & E_0 & & \\ \hline
C^1_{\ulcorner 1} & & E_1 & \\ \hline
C^1_{1\lrcorner} & & & E_1 \\ \hline
C_2^2 & & & D^2
\end{array}\right) \kappa^1 = \left(\begin{array}{c|ccccc}
 & s_1 & s_2 & s_3 & s_4 & s_5 \\
x^{\urcorner} & 1 & 1 & 1 & 1 & 1 \\
y^{\urcorner} & \textcircled{1} & & & & \\ \hline
{}^{\ulcorner}c & & & 1 & 1 & 1 \\
{}^{\ulcorner}b & & & & 1 & 1 \\
{}^{\ulcorner}a & & & & & \textcircled{1} \\ \hline
a_{\lrcorner} & & 1 & 1 & 1 & \\
b_{\lrcorner} & & 1 & 1 & & \\
c_{\lrcorner} & & \textcircled{1} & & & \\ \hline
{}_{\llcorner}h & & & 1 & & \\
{}_{\llcorner}g & & & 1 & 1 & \\
{}_{\llcorner}f & & & & 1 &
\end{array}\right),$$

where κ^0 is the kernel basis from (4.45). Applying Algorithm 3 yields

$$\tilde{h}_2^1 = \left(\begin{array}{c|cc}
 & s_3 & s_4 \\
x^{\urcorner} & 1 & 1 \\ \hline
{}^{\ulcorner}c & 1 & 1 \\
{}^{\ulcorner}b & & 1 \\ \hline
a_{\lrcorner} & 1 & 1 \\
b_{\lrcorner} & 1 & \\ \hline
{}_{\llcorner}h & 1 & \\
{}_{\llcorner}g & 1 & 1 \\
{}_{\llcorner}f & & 1
\end{array}\right), \qquad \tilde{h}_1^1 = \left(\begin{array}{c|c|cc|cc|ccc}
 & x^{\urcorner} & {}^{\ulcorner}c & {}^{\ulcorner}b & a_{\lrcorner} & b_{\lrcorner} & {}_{\llcorner}h & {}_{\llcorner}g & {}_{\llcorner}f \\
\gamma_1 & \textcircled{1} & 1 & & & & & & \\
\gamma_2 & & & & \textcircled{1} & & & 1 & \\
\gamma_3 & & & & & \textcircled{1} & & & 1 \\
\gamma_4 & & & & & & 1 & 1 & 1 \\
\gamma_5 & & \textcircled{1} & & & & & 1 & \\
\gamma_6 & & & \textcircled{1} & & & 1 & &
\end{array}\right)$$

and finally

$$\tilde{h}'^1_2 \begin{array}{c|cc}
 & s_3 & s_4 \\
r_1 & 1 & \\
r_2 & 1 & 1 \\
r_3 & & 1
\end{array}, \qquad \tilde{h}'^1_1 = \begin{array}{c|ccc}
 & \overbrace{{}_{\llcorner}h+{}^{\ulcorner}b}^{r_1} & \overbrace{{}_{\llcorner}g+a_{\lrcorner}+{}^{\ulcorner}c}^{r_2} & \overbrace{{}_{\llcorner}f+b_{\lrcorner}}^{r_3} \\
\gamma_4 = b^{\urcorner}+{}^{\ulcorner}f+h_{\lrcorner} & 1 & 1 & 1
\end{array},$$

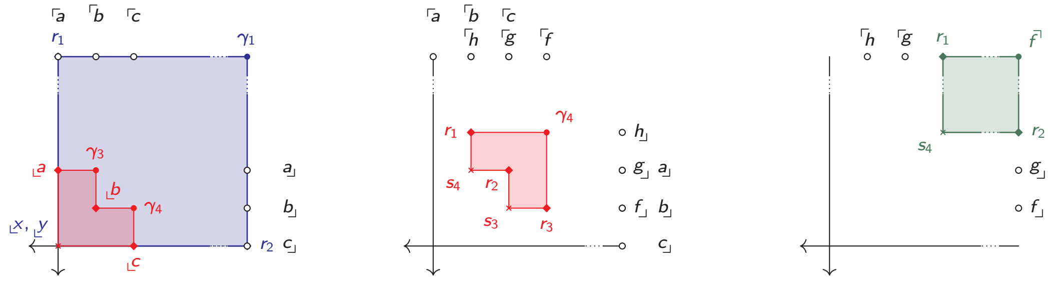
which represent the minimal free resolution

$$0 \to \langle s_3, s_4 \rangle \xrightarrow{\tilde{h}'^1_2} \langle r_1, r_2, r_3 \rangle \xrightarrow{\tilde{h}'^1_1} \langle \gamma_4 \rangle \to R_zH^1(K_*) \tag{4.47}$$

of $R_zH^1(K_*)$; see Figure 4.6b.

2-Cohomology Lastly, there are no 3-simplices, which implies $C_\bullet^3 = 0$. According to Proposition 4.2.5, $R_zH^2(K_*)$ is generated by $C_0^2 = \langle f^{\urcorner}, g^{\urcorner}, h^{\urcorner} \rangle$. Without calculation, we obtain that

$$h_2^2 = \kappa^1 = \left(\begin{array}{c|cccccc}
 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\
a^{\urcorner} & 1 & 1 & & & 1 & \\
b^{\urcorner} & 1 & & 1 & 1 & 1 & \\ \hline
c^{\urcorner} & \textcircled{1} & & & & & \\
{}^{\ulcorner}f & & & & 1 & 1 & 1 \\
{}^{\ulcorner}g & & & & & \textcircled{1} & \\
{}^{\ulcorner}h & & & & & & \textcircled{1} \\ \hline
h_{\lrcorner} & & 1 & 1 & 1 & & \\
g_{\lrcorner} & & \textcircled{1} & & & & \\
f_{\lrcorner} & & & \textcircled{1} & & &
\end{array}\right), \qquad h_1^2 = \left(\begin{array}{c|ccc|ccc|ccc}
 & a^{\urcorner} & b^{\urcorner} & c^{\urcorner} & {}^{\ulcorner}f & {}^{\ulcorner}g & {}^{\ulcorner}h & h_{\lrcorner} & g_{\lrcorner} & f_{\lrcorner} \\
f^{\urcorner} & & 1 & 1 & 1 & & & & & 1 \\
g^{\urcorner} & 1 & & 1 & & 1 & & & 1 & \\
h^{\urcorner} & 1 & 1 & & & & 1 & 1 & &
\end{array}\right).$$



(a) $R_zH^0(K_*)$ **(b)** $R_zH^1(K_*)$ **(c)** $R_zH^2(K_*)$

Figure 4.6: Truncated absolute cohomology $R_zH^\bullet(K_*)$ of the filtered simplicial complex K_* from Figure 4.4. The module $R_zH^0(K_*)$ is decomposable into a direct sum of its red and its blue submodule.

Applying Algorithm 3 to h_2^2 yields

$$\tilde{h}_2^2 = \begin{array}{c|c} & s_4 \\ a^\urcorner & 0 \\ b^\urcorner & 1 \\ \hline {}^\ulcorner f & 1 \\ \hline h_\lrcorner & 1 \end{array}, \qquad \tilde{h}_1^2 = \begin{array}{c|cccc} & a^\urcorner & b^\urcorner & {}^\ulcorner f & h_\lrcorner \\ f^\urcorner & & 1 & 1 & \\ g^\urcorner & 1 & & & \\ h^\urcorner & 1 & 1 & & 1 \end{array}.$$

When we apply Algorithm 3 to $\tilde{h}_1^2$, we are in a situation that we have not met before: namely, one of the local pairs of $\tilde{h}_1^2$ can only be identified after a column addition caused by the removal of the non-zero entries in the local rows. Specifically, Algorithm 3 transforms $\tilde{h}_1^2$ as follows:

$$\tilde{h}_1^2 = \begin{array}{c|cc|cc} & a^\urcorner & b^\urcorner & {}^\ulcorner f & h_\lrcorner \\ f^\urcorner & & 1 & 1 & \\ g^\urcorner & 1 & & & \\ h^\urcorner & \textcircled{1} & 1 & & 1 \end{array} \rightsquigarrow \begin{array}{c|cc|cc} & a^\urcorner & a^\urcorner{+}b^\urcorner & {}^\ulcorner f & h_\lrcorner \\ f^\urcorner & & 1 & 1 & \\ g^\urcorner & 1 & \textcircled{1} & & \\ h^\urcorner & \textcircled{1} & & & 1 \end{array} \rightsquigarrow \begin{array}{c|cc|cc} & a^\urcorner & a^\urcorner{+}b^\urcorner & {}^\ulcorner f & h_\lrcorner{+}b^\urcorner \\ f^\urcorner & & 1 & 1 & 1 \\ g^\urcorner & 1 & \textcircled{1} & & \\ h^\urcorner & \textcircled{1} & & & \end{array},$$

where the first arrow happens during the first phase and the second arrow during the second phase of Algorithm 3, with the circled local pairs. Removal of the rows and columns with local pairs gives

$$\tilde{h}_2'^2 = \begin{array}{c|c} & s_4 \\ r_1 & 1 \\ r_2 & 1 \end{array}, \qquad \tilde{h}_1'^2 = \begin{array}{c|cc} & \overbrace{f^\urcorner}^{r_1} & \overbrace{h_\lrcorner{+}b^\urcorner}^{r_2} \\ f^\urcorner & 1 & 1 \end{array}.$$

Then $\tilde{h}_2'^2$ and $\tilde{h}_1'^2$ represent the minimal free resolution

$$0 \to \langle s_4 \rangle \xrightarrow{\tilde{h}_2'^2} \langle r_1, r_2 \rangle \xrightarrow{\tilde{h}_1'^2} \langle f^\urcorner \rangle \to R_zH^2(K_*) \tag{4.48}$$

of $R_zH^2(K_*)$; see Figure 4.6c.

4.3.2.1 The long exact sequence

Lastly, we give an explicit description of the the long exact sequence

$$\begin{tikzcd} 0 \arrow[r] & R_zH^0(K,K_*) \arrow[r, "p^0"] & R_z\Delta H^0(K) \arrow[r, "i^0"] & R_zH^0(K_*) \arrow[dll, rounded corners, "\delta"] \\ & R_zH^1(K,K_*) \arrow[r, "p^0"] & R_z\Delta H^1(K) \arrow[r, "i^1"] & R_zH^1(K_*) \arrow[dll, rounded corners, "\delta"] \\ & R_zH^2(K,K_*) \arrow[r, "p^1"] & R_z\Delta H^2(K) \arrow[r, "i^2"] & R_zH^2(K_*) \arrow[r] & 0, \end{tikzcd} \tag{4.49}$$

for the above filtration K_*. Here, p^* and i^* are the morphisms induced by the morphisms $i\colon K_* \hookrightarrow K$ and $p\colon K \to K/K_*$. Recall from Propositions 4.2.5 and 4.2.7 that in the free resolutions $R_zH^d(K,K_*)_\bullet$ and $R_zH^d(K_*)_\bullet$, we had

$$\begin{aligned} R_zH^d(K,K_*)_0 &= \ker(\qquad\qquad\qquad C^d(K_*)_1 \longrightarrow C^{d+1}(K_*)_0), \\ R_zH^d(K_*)_0 &= \ker(C^d(K_*)_0 \oplus C^{d+1}(K_*)_1 \longrightarrow C^{d+1}(K_*)_0). \end{aligned}$$

Consider the module $R_z\Delta H^d(K)$, which has the components $(R_z\Delta H^d(K))_w = H^d(K)$ if $w \geq -z$, and $(R_z\Delta H^d(K))_w = 0$ otherwise. This has a free resolution

$$0 \to \ker(C_0^{d-1} \to C_0^d) \to C_0^{d-1} \to \ker(C_0^d \to C_0^{d+1}) \to R_z\Delta H^d(K). \tag{4.50}$$

Note that $R_z\Delta H^d(K) = H^d(C_0^\bullet)$. The long exact sequence (4.49) is induced by the commutative diagram

$$\begin{tikzcd}
 & & \cdots \arrow[r] & R_zH^d(K,K_*) \arrow[rr, "i^d"] & & R_z\Delta H^d(K) \arrow[rr, "p^d"] & & R_zH^d(K_*) \arrow[rr, "\delta"] & & R_zH^{d+1}(K,K_*) \arrow[r] & \cdots \\
 & \cdots \arrow[r] & \ker\delta_0^d c_1^d \arrow[ur, two heads] \arrow[rr] \arrow[dd, hook] & & \ker\delta_0^d \arrow[ur, two heads] \arrow[rr] \arrow[dd, hook] & & \ker(\delta_0^d, -c_1^{d+1}) \arrow[ur, two heads] \arrow[rr] \arrow[dd, hook] & & \ker\delta_0^{d+1} \arrow[ur, two heads] \arrow[r] \arrow[dd, hook] & \cdots & \\
\cdots \arrow[r] & C_1^{d-1} \oplus C_2^d \arrow[ur, dashed, two heads] \arrow[rr, "{(c_1^{d-1},0)}"] \arrow[dr, "{(\delta_1^{d-1},-c_2^d)}"'] & & C_0^{d-1} \arrow[ur, dashed, two heads] \arrow[rr, "{\begin{pmatrix}1\\0\\0\end{pmatrix}}"] \arrow[dr, "\delta_0^{d-1}"'] & & C_0^{d-1} \oplus C_1^d \oplus C_2^{d+1} \arrow[ur, dashed, two heads] \arrow[rr, "{\begin{pmatrix}0&1&0\\0&0&1\end{pmatrix}}"] \arrow[dr, "{\begin{pmatrix}\delta_0^{d-1} & c_1^d & 0\\ 0 & \delta_1^d & c_2^{d+1}\end{pmatrix}}"'] & & C_1^d \oplus C_2^{d+1} \arrow[ur, dashed, two heads] \arrow[rr] \arrow[dr, "{(\delta_1^d,-c_2^{d+1})}"'] & & \cdots & \\
 & \cdots \arrow[r] & C_1^d \arrow[rr, "c_1^d"'] \arrow[d, "\delta_0^d c_1^d"'] & & C_0^d \arrow[rr, "{\begin{pmatrix}1\\0\end{pmatrix}}"'] \arrow[d, "\delta_0^d"'] & & C_0^d \oplus C_1^{d+1} \arrow[rr, "{(0,-1)}"'] \arrow[d, "{(\delta_0^d,-c_1^{d+1})}"'] & & C_1^{d+1} \arrow[r] \arrow[d, "\delta_0^{d+1}c_1^{d+1}"'] & \cdots & \\
 & \cdots \arrow[r, equal] & C_0^{d+1} \arrow[rr, equal] & & C_0^{d+1} \arrow[rr, equal] & & C_0^{d+1} \arrow[rr, "\delta_0^d"'] & & C_0^{d+2} \arrow[r, equal] & \cdots . &
\end{tikzcd}$$

In the diagram, the dashed arrows are the free presentations of $R_zH^\bullet(K,K_*)$, $R_zH^\bullet(K_*)$ and $R_z\Delta H^\bullet(K_*)$ from Propositions 4.2.5 and 4.2.7 and (4.50). The connecting homomorphism $\delta\colon H^d(K_*) \to H^{d+1}(K,K_*)$ is given by projecting a generator $(a,b) \in H^d(K_*)_0 \subseteq C_0^d \oplus C_1^{d+1}$ of $H^d(K_*)$ to its component $b \in C_1^{d+1}$. In practice, this amounts to dropping all summands of the form $\sigma^{\urcorner}$. Recall the generators of $H^\bullet(K_*)$ from (4.46) and (4.47) and $H^\bullet(K,K_*)$ from (4.42) and (4.43). We obtain that

$$\begin{aligned} \delta\colon H^0(K_*) &\longrightarrow H^1(K,K_*), \\ \gamma_1 = x^{\urcorner} + y^{\urcorner} \qquad\qquad &\longmapsto 0, \\ \gamma_3 = x^{\urcorner} + a_{\lrcorner} + b_{\lrcorner} + {}^{\ulcorner}b &\longmapsto a_{\lrcorner} + b_{\lrcorner} + {}^{\ulcorner}b = \gamma_2, \\ \gamma_4 = x^{\urcorner} + a_{\lrcorner} + {}^{\ulcorner}b + {}^{\ulcorner}b &\longmapsto a_{\lrcorner} + {}^{\ulcorner}b + {}^{\ulcorner}b = \gamma_3, \end{aligned} \qquad \begin{aligned} \delta\colon H^1(K_*) &\longrightarrow H^2(K,K_*), \\ \gamma_4 = b^{\urcorner} + {}^{\ulcorner}f + h_{\lrcorner} &\longmapsto {}^{\ulcorner}f + h_{\lrcorner}. \end{aligned}$$

Therefore, the long exact sequence (4.49) is the one shown in Figure 4.7.

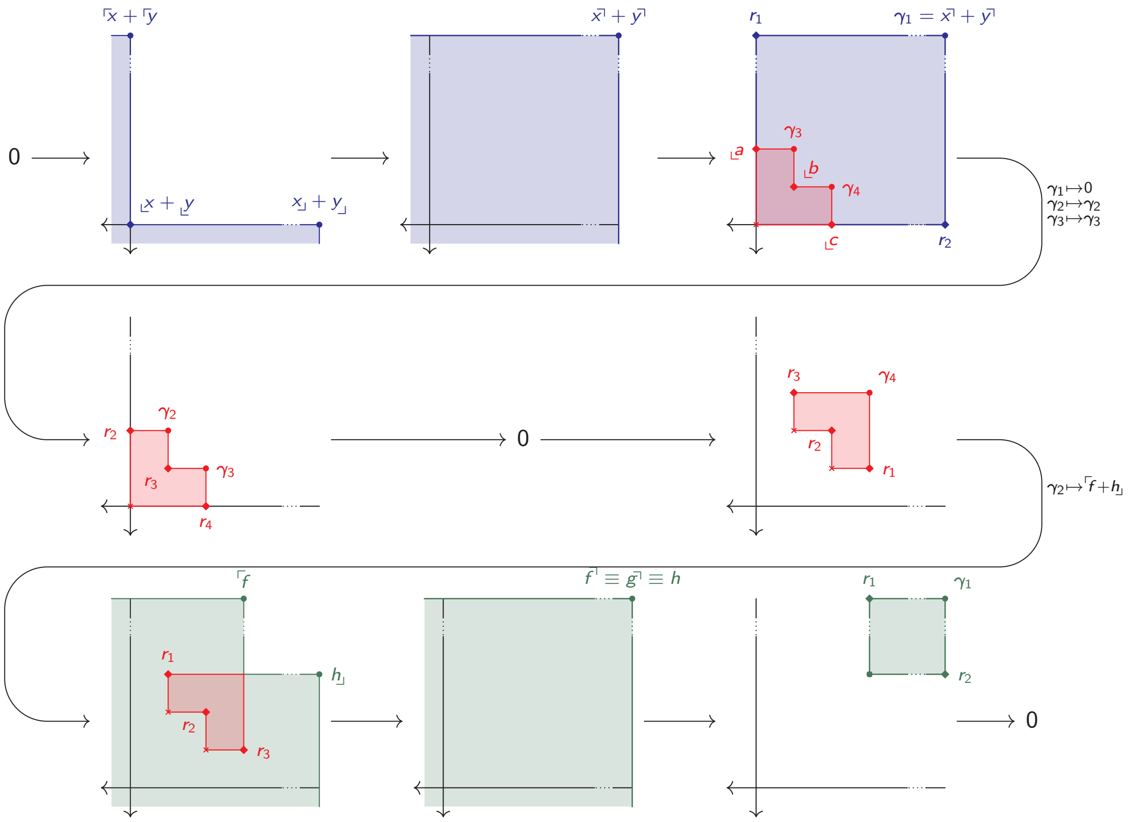
Remark. This example also exhibits a particularity that cannot happen in one-parameter persistence. Namely, recall that unsplicing the long exact sequence (4.49) gives short exact sequences

$$\begin{gathered} 0 \to \operatorname{coker} p^d \to H^d(K_*) \to \ker p^{d+1} \to 0, \\ 0 \to \operatorname{coker} i^d \to H^{d+1}(K,K_*) \to \ker i^{d+1} \to 0 \end{gathered}$$

for all d. In one-parameter persistence, both short exact sequences split for all d (see Proposition 2.2.10). For the above example, Figure 4.7 shows that the short exact sequence

$$0 \to \operatorname{coker} i^1 \to H^2(K,K_*) \to \ker i^2 \to 0 \tag{4.51}$$

need not split for $d = 2$; see Figure 4.8. This implies that $H^\bullet(K_*)$ does not determine $H^\bullet(K,K_*)$ uniquely. An explicit example for this is given in Section 4.6.



0
$\ulcorner x + \ulcorner y$
$x^\urcorner + y^\urcorner$
r_1
$\gamma_1 = x^\urcorner + y^\urcorner$
$\llcorner x + \llcorner y$
$x_\lrcorner + y_\lrcorner$
$\llcorner a$
γ_3
$\llcorner b$
γ_4
$\llcorner c$
r_2
$\gamma_1 \mapsto 0$
$\gamma_2 \mapsto \gamma_2$
$\gamma_3 \mapsto \gamma_3$
γ_2
r_2
r_3
γ_3
r_4
0
r_3
γ_4
r_2
r_1
$\gamma_2 \mapsto \ulcorner f + h_\lrcorner$
$\ulcorner f$
$f^\urcorner \equiv g^\urcorner \equiv h$
r_1
γ_1
r_1
$h_\lrcorner$
r_2
r_3
r_2
0

Figure 4.7: Long exact sequence (4.49) in (truncated) cohomology of the pair (K, K_*) for the complex K_* from Figure 4.4. See Figures 4.5 and 4.6 for details on the modules. Note that the module at the bottom left is indecomposable. This gives rise to the non-split short exact sequence in Figure 4.8.

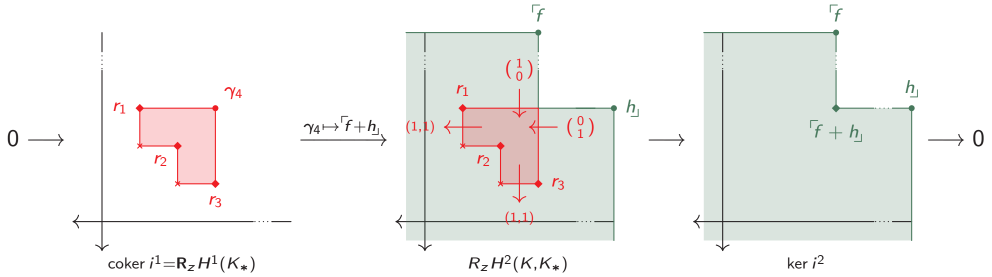


Figure 4.8: The non-split short exact sequence (4.51) obtained from Figure 4.7. The relations of the middle module $R_zH^2(K, K_*)$ are r_1, r_2, r_3: $\ulcorner f + f_\lrcorner \equiv 0$.

4.4 Relative cohomology revisited

In this section, we will have a closer look at the strategy from Section 4.2.2 to compute a minimal free resolution of $H^{d+1}(K, K_*)$. As before, K_* is a two-parameter filtered complex for which $K = \operatorname{colim}_{\mathbf{Z}^2} K_*$ is acyclic. Let D^{d+1} denote the (ungraded) coboundary matrix of K_* throughout this section. In principle, the free resolution of $H^{d+1}(K, K_*)$ from Proposition 4.2.7 can be computed using the procedures `Ker()` and `Factorize()` (Algorithms 6 and 7). This has the following shortcoming:

Recall that our goal is to compute a minimal free resolution of $H_d(K_*)$. As K is acyclic, we have $H^{d+1}(K, K_*) \cong H^d(K_*)$ for all d, so we may compute a minimal free resolution of $H^{d+1}(K, K_*)$, get a minimal free resolution of $H^d(K_*)$ at the same time, and use Corollary 3.2.11 to obtain a minimal free resolution of $H_d(K_*)$. However, computing $H^{d+1}(K, K_*)$ as described in 4.2.8 requires computing $\ker c_1^{d+2}\delta_1^{d+2}$. Given that we are mainly interested in Vietoris–Rips complexes, considering the coboundary matrix δ^{d+2} is not feasible because of the sheer size of a matrix representing this morphism.

In one parameter persistence, this problem is solved by observing that it is not necessary in this case to compute $Z^{d+1}(K, K_*)$ from D^{d+2} because $Z^{d+1}(K, K_*)$ can already be computed from D^{d+1}, which may be much smaller for Vietoris–Rips complexes. In two parameter persistence, we already encountered a similar situation in Theorem C: there, we showed that under suitable conditions on K_*, the free resolution

$$0 \to Z^{d+1}(N^\bullet(K_*)) \longrightarrow N^{d+1}(K_*) \longrightarrow Z^{d+2}(N^\bullet(K_*)) \longrightarrow H^{d+2}(N^\bullet(K_*))$$

of $H^{d+2}(N^\bullet(K_*))$ can be computed from the coboundary matrix D^{d+1} alone, and not, as one may assume, by reducing the coboundary matrix D^{d+3} of $N^\bullet$.

We present a similar result for $H^{d+1}(K, K_*)$. Recall the functors $\operatorname{colim}, \lim \colon \mathrm{Vec}^{\mathbf{Z}^2} \to \mathrm{Vec}$ from Section 2.1.2.

Lemma 4.4.1. *If $H^\bullet(K) = 0$, then $\operatorname{colim} \operatorname{im}(d_1^{d+1}, c_2^{d+1}) = \operatorname{colim} \ker(c_1^{d+2}\delta_1^{d+2})$ for all d.*

Proof. Consider the morphisms h_2^{d+1} and h_1^{d+1} that represent the free resolution $H^{d+1}(K, K_*)_\bullet$ of $H^{d+1}(K, K_*)$ from (4.34). Because $\mathbf{Z}^2$ is a direct system, colim is exact. Therefore, the morphisms $\operatorname{colim} h_2^{d+1}$ and $\operatorname{colim} h_1^{d+1}$ form a free resolution of the vector space $\operatorname{colim} H^{d+1}(K, K_*)$. By assumption, $\operatorname{colim} H^{d+1}(K, K_*) = \lim H^{d+1}(K_*) = H^{d+1}(K) = 0$. According to Theorem 2.3.18, the free resolution $\operatorname{colim} H^{d+1}(K, K_*)_\bullet$ is trivial; in particular, it is exact. Therefore, $\operatorname{colim} h_1^{d+1}$ is surjective. From Proposition 4.2.7, we obtain a commutative diagram

$$\begin{tikzcd}
\operatorname{colim} H^{d+1}(K, K_*)_1 \arrow[r, two heads, "\operatorname{colim} h_1^{d+1}"] \arrow[dr, "\operatorname{colim}(d_1^{d+1}{,}c_2^{d+1})"'] & \operatorname{colim} H^{d+1}(K, K_*)_0 \arrow[d, hook, "\operatorname{colim} k^{d+1}"] \\
 & \operatorname{colim} C_1^{d+1}.
\end{tikzcd}$$

Recall that (in any abelian category) a morphism f is surjective if and only if $\operatorname{im} gf = \operatorname{im} f$ for every morphism g that can be composed with f. Since

$$\operatorname{colim}(d_1^{d+1}, c_2^{d+1}) = (\operatorname{colim} k^{d+1})(\operatorname{colim} h_1^{d+1}),$$

the claim follows. □

Theorem D. *If $H_\bullet(K) = 0$, then the free resolution (4.34) of $H^{d+1}(K, K_*)$ can be computed solely from the coboundary matrix D^{d+1} of K_*.*

Proof. According to Proposition 4.2.7, the matrix D^{d+1} is not needed to compute h_2^{d+1}. It remains to show the same for h_1^{d+1}. Consider the submodules

$$\begin{aligned} H^{d+1}(K, K_*)_0 &\coloneqq \ker(c_1^{d+2}\delta_1^{d+2}) \\ &\cup\!\mathsf{I} \\ N &\coloneqq \operatorname{im}(d_1^{d+1}, c_2^{d+1}) \end{aligned}$$

of C_1^{d+1}. According to Lemma 4.4.1, we have $\operatorname{colim} N = \operatorname{colim} H^{d+1}(K, K_*)_0$. Theorem F states that $[\operatorname{colim} N]_{C_1^{d+1}} = H^{d+1}(K, K_*)_0$. A generating system of the vector subspace $\operatorname{colim} N$ of $\operatorname{colim} C_1^{d+1}$ is given by the underlying matrix $\mathtt{u}(\tilde{D}^{d+1})$ of $\tilde{D}^{d+1} \coloneqq c_1^{d+1}\delta^{d+1}$. Theorem 3.4.7 states that the matrix $k^{d+1} \coloneqq \mathtt{Bireduce}(\tilde{D}^{d+1})$ represents a basis of $H^{d+1}(K, K_*)_0$. Recall that by Proposition 4.2.7, it remains to compute the solution h_1^{d+1} of the linear system $\tilde{D}^{d+1} = k^{d+1}h_1^{d+1}$, which now does not involve the matrix D^{d+2} anymore. Thus, h_1^{d+1} can be computed without considering D^{d+2}. □

Almost clearing An immediate optimization step that we have not mentioned so far is analogous to what we explained in Section 2.4.1. Namely, calling $\mathtt{Bireduce}(\tilde{D}^{d+1})$ will reduce some columns of $\tilde{D}^{d+1}$ to zero that could have been predicted. Since $\tilde{D}^{d+1}h_2^{d+1} = k^{d+1}h_1^{d+1}h_2^{d+1} = 0$, the sequence

$$0 \to H^{d+1}(K, K_*)_2 \xhookrightarrow{h_2^{d+1}} H^{d+1}(K, K_*)_1 \xrightarrow{\tilde{D}^{d+1}} C_1^{d+1}$$

is a chain complex; namely a free resolution of $\operatorname{coker} \tilde{D}^{d+1}$. Since k^{d+1} is injective and since $H^{d+1}(K, K_*)_\bullet$ is exact, this complex can be seen as a free resolution of some quotient module M of C_1^{d+1}. Using `KerAndMgsWithKer()` (Algorithm 8), we compute a free resolution $F_\bullet$ of $\operatorname{coker} \tilde{D}^{d+1}$ with $F_0 = C_1^{d+1}$, such that $F_\bullet$ does not contain any homological 2-balls. This resolution $F_\bullet$ fits into the commutative diagram

$$\begin{tikzcd}
0 \arrow[r] & H^{d+1}(K, K_*)_2 \arrow[r, "h_2^{d+1}"] \arrow[d] & H^{d+1}(K, K_*)_1 \arrow[r, "h_1^{d+1}"] \arrow[d] \arrow[ddrr, bend left, "\tilde{D}^{d+1}"] & H^{d+1}(K, K_*)_0 \arrow[r] \arrow[d, equal] \arrow[ddd, hook, "k^{d+1}"'] & H^{d+1}(K, K_*) \arrow[r] \arrow[d, equal] & 0 \\
0 \arrow[r] & F_2 \arrow[r, "f_2"'] & F_1 \arrow[r, "\tilde{f}_1"'] \arrow[ddr, bend right, "f_1"'] & H^{d+1}(K, K_*)_0 \arrow[r, hook] \arrow[dr, "k^{d+1}"] & H^{d+1}(K, K_*) \arrow[r] & 0, \\
 & & & & C_1^{d+1} \arrow[dl, equal] & \\
 & & & C_1^{d+1} & &
\end{tikzcd} \tag{4.52}$$

where $\operatorname{coker} f_1 \cong \operatorname{coker} \tilde{D}^{d+1}$. Note that f_1 is a semi-minimal presentation of $\operatorname{coker} \tilde{D}^{d+1}$, in the sense of [99]. Thus, we can obtain a matrix representing k^{d+1} by calling $\mathtt{Bireduce}(f_1)$ instead of $\mathtt{Bireduce}(\tilde{D}^{d+1})$. This may yield a different basis of $H^{d+1}(K, K_*)_0$ and thus a different matrix k^{d+1}, which nevertheless represents the same morphism k^{d+1}. The morphism $\tilde{f}_1$ then is the unique solution of the linear system $k^{d+1}\tilde{f}_1 = f_1$. Now f_2 and $\tilde{f}_1$ form a free resolution of $H^{d+1}(K, K_*)$ not containing any 2-balls.

Avoiding the factorization of f_1 It seems unnecessary that we first compute k^{d+1} by applying the column reduction algorithm Algorithm 10 to f_1, and in the second step compute the matrix $\tilde{f}_1$ by applying Algorithm 7 to f_1, which is again a column reduction scheme. This can be addressed by keeping track of the column operations during Algorithm 10. Let F and G be free modules, $B\colon F \to G$ be a valid graded matrix, and consider the free submodule $[\operatorname{colim} \operatorname{im} B]_G$ of G.

Algorithm 12: A variant of Algorithm 10 that keeps track of the column operations.

Input: A valid graded $m \times n$-matrix $B\colon F \to G'$

Output: An injective valid graded $m \times n'$-matrix $B'\colon F\colon \to F$ representing a basis of $F' = [\operatorname{colim} \operatorname{im} B]_G$, and a valid graded matrix $W\colon F \to F'$ such that $B = B'W$

```
function BireduceF($B$):
        $W \leftarrow E \in k^{n \times n}$
        $p \leftarrow 0 \in \mathbf{N}^n$
        for $j = 1, \dots, n$ do
            while $i \leftarrow$ c-piv$_{\vec{z}}(B_j) \neq 0$ do
                if $p_i = 0$ then $p_i = j$; break
(a)             if l-piv$_{\vec{z}}(B_j) <$ l-piv$_{\vec{z}}(B_{p_i})$ then swap $p_i$ and $j$
(b)             $B_j \leftarrow B_j - B_{ij}/B_{ip_i} B_{p_i}$
                $W_{p_i,*} \leftarrow W_{p_i,*} + B_{ij}/B_{ip_i} W_{j*}$
(c)     $B \leftarrow (B_j)_{B_j \neq 0}$
        $W \leftarrow (W_{j*})_{B_j \neq 0}$
        $p \leftarrow 0 \in \mathbf{N}^n$
        for $j' = 1, \dots, n$ do
            $j \leftarrow j'$
            while $i \leftarrow$ l-piv$_{\vec{z}}(B_j) \neq 0$ do
                if $p_i = 0$ then $p_i = j$; break
(d)             if c-piv$_{\vec{z}}(B_j) <$ c-piv$_{\vec{z}}(B_{p_i})$ then swap $p_i$ and $j$
(e)             $B_j \leftarrow B_j - B_{ij}/B_{ip_i} B_{p_i}$
                $W_{p_i,*} \leftarrow W_{p_i,*} + B_{ij}/B_{ip_i} W_{j*}$
        $B' \leftarrow [B]_{\mathrm{rg} B}$
        $\mathrm{rg}^W \leftarrow \mathrm{cg}^{B'}$
        $\mathrm{cg}^W \leftarrow \mathrm{cg}^B$
        return $B', W$
```

Proposition 4.4.2. *Algorithm 12 computes two valid graded matrix B' and W, such that B' represents a basis of $[\operatorname{colim} \operatorname{im} B]_G$ and the following diagram commutes:*

$$\begin{tikzcd}
 & {[\operatorname{colim} \operatorname{im} B]_G} \arrow[rd, "B'"] \arrow[dd, dashed] & \\
F \arrow[ru, hook, "W"] \arrow[rr, "B"'] \arrow[dd] & & G \arrow[dd] \\
 & \operatorname{colim} \operatorname{im} B \arrow[rd] & \\
\operatorname{colim} F \arrow[ru, equal] \arrow[rr] & & \operatorname{colim} G.
\end{tikzcd}$$

Proof. Let $F' = [\operatorname{colim} \operatorname{im} B]_G$. Algorithm 12 differs from Algorithm 10 only in maintaining the additional matrix W. Since operations on W do not affect the reduction of B, Theorem 3.4.7 gives that B' is a basis of $[\operatorname{colim} \operatorname{im} B]_G$. It follows from elementary linear algebra that $B = B'W$ as ungraded matrices, and it remains to prove that W is valid. We show this inductively.

Let $B^{(0)} = B, \dots, B^{(t)} = B'$ and $W^{(0)} = E_{n \times n}, \dots, W^{(t)} = W$ be the intermediate steps of the matrices B and W during the algorithm. We regard all matrices $B^{(s)}$ and $W^{(s)}$ as graded matrices with

$$\mathrm{rg}^{B^{(s)}} := \mathrm{rg}^B, \qquad \mathrm{rg}_j^{W^{(s)}} := \mathrm{cg}_j^{B^{(s)}} := \bigvee\nolimits_{B_{ij}^{(s)} \neq 0} \mathrm{rg}_i^B, \qquad \mathrm{cg}^{W^{(s)}} := \mathrm{cg}^B.$$

This renders all matrices $B^{(s)}$ valid and matches the definition of the row and column grades of B' and W at the end of the algorithm. The graded matrix $B^{(0)} = W$ is valid by assumption. The unit matrix $W^{(0)}$ is valid because $\mathrm{cg}^{W^{(0)}} = \mathrm{rg}^{W^{(0)}} = \mathrm{cg}^B$. By induction, we assume that $B^{(s-1)}$ and $W^{(s-1)}$ are valid for some s. Consider a pair of corresponding row and column

Algorithm 13: Relative cohomology algorithm

```
Input:   Coboundary matrices $D^1, D^2, \dots$ representing $C^\bullet(K, K_*)$ with $H^\bullet(K) = 0$
Output:  Matrices representing a minimal free resolution of $H^{d+1}(K, K_*)$ for $d > 0$.
$k \leftarrow$ Ker($(D^1\Pi_0, D^1)\colon C_1^0 \to C_0^1$)
for $d = 0, 1, \dots, d_{max}$ do
    $h_2$       $\leftarrow \left(\begin{smallmatrix} E \\ (0, D^{d+1}) \end{smallmatrix}\right) K\colon H^{d+1}(K, K_*)_1 \to C_1^d \oplus C_2^{d+1}$
    $\tilde{D}$       $\leftarrow \left(\begin{smallmatrix} \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 & \Pi_{d+1}^{-1} \\ 0 & D^{d+1} & E_{d+1} \end{smallmatrix}\right)\colon C_1^d \oplus C_2^{d+1} \to C_1^{d+1}$
    $f_2, r, c$ $\leftarrow$ Minimize($h_2$)
(a) $q$         $\leftarrow$ Reduce($h_2$)                                  ▷ optional
(b) $r$         $\leftarrow (r_j)_{j \notin \{\mathrm{piv}\, q_k\}}$
    $f_1$       $\leftarrow (\tilde{D}_j)_{j \in r}$                              ▷ colim im $f_1$ = colim $\tilde{D}_r$
    if $d < d_{max}$ then $k, \tilde{f}_1 \leftarrow$ BireduceF($f_1$)        ▷ $f_1 = k\tilde{f}_1$
    else $\tilde{f}_1 \leftarrow$ Bireduce($f_1$)
    $\tilde{f}_1, r, c \leftarrow$ Minimize($\tilde{f}_1$)
    $\tilde{f}_2 \leftarrow ([f_2]_{i*})_{i \in c}$
    yield $(\tilde{f}_2, \tilde{f}_1)$                                  ▷ resolution of $H^{d+1}(K, K_*)$
```

operations

$$B_j^{(s)} \leftarrow B_j^{(s-1)} - \lambda B_{p_i}^{(s-1)}, \qquad\qquad W_{p_i,*}^{(s)} \leftarrow W_{p_i,*}^{(s-1)} + \lambda W_{j,*}^{(s-1)} \tag{4.53}$$

for some $\lambda \in k$, either in line (b) or line (e). Because $B^{(s-1)}$ is a $\mathbf{Z}^2$-graded matrix, two column grades $\mathrm{cg}_j^{B^{(s)}}$ and $\mathrm{cg}_{p_i}^{B^{(s)}}$ are comparable whenever two columns $B_{p_i}^{(s-1)}$ and $B_j^{(s)}$ have the same lex or colex pivot. By swapping p_i and j if necessary, lines (a) and (d) ensure that whenever the algorithm performs an operation as in (4.53), then $\mathrm{cg}_{p_i}^{B^{(s-1)}} \leq \mathrm{cg}_j^{B^{s-1}}$ and thus $\mathrm{cg}_j^{B^{(s)}} \leq \mathrm{cg}_j^{B^{(s-1)}}$. Using the induction hypothesis that $W^{(s-1)}$ is valid in $(*)$, we obtain

$$\mathrm{rg}_j^{W^{(s)}} \overset{\mathrm{def}}{=} \mathrm{cg}_j^{B^{(s)}} \leq \mathrm{cg}_j^{B^{(s-1)}} \overset{\mathrm{def}}{=} \mathrm{rg}_j^{W^{(s-1)}} \overset{(*)}{\leq} \mathrm{cg}_k^{W^{(s-1)}} = \mathrm{cg}_k^{W^{(s)}}$$

for all k, so $W^{(s)}$ is valid. By induction, W is valid. □

We use Algorithm 12 to compute a free resolution of $H^\bullet(K, K_*)$:

Proposition 4.4.3. *Algorithm 13 computes graded matrices representing a minimal free resolution of $H^{d+1}(K, K_*)$ for each $0 \leq d \leq d_{max}$.*

Proof. The two gray lines are optional, and we claim that they do not affect the correctness. We first prove correctness without these lines. The algorithm first computes matrices representing the free resolution $F_\bullet$ of coker $\tilde{D}^{d+1}$ from (4.52). Recall that $F_\bullet$ contains no homological 2-balls. The algorithm computes k and $\tilde{f}_1$ such that $f_1 = k\tilde{f}_1$ using `BireduceF()`. Correctness of this step is given by Proposition 4.4.2. Then f_2 and $\tilde{f}_1$ represent a free resolution of $H^{d+1}(K, K_*)$ that does not contain any homological 2-balls. To obtain a minimal free resolution, it remains to split off all homological 1-balls, which is done by `Minimize()`. Note that the matrix k is only needed in the next iteration of the main for-loop. Thus, if $d = d_{\max}$, it suffices to compute $\tilde{f}_1$ but not k, using `Bireduce()`.

Now, consider the algorithm with the two gray lines. Line (a) computes an ungraded, reduced matrix q with $\mathrm{im}\, q = \mathrm{im}\,\mathrm{colim}\, f_1$. This implies that $\mathrm{u}(()f_1)q = 0$. Consider the submatrix $f_1' := ([f_1]_{r_j})_j$ of f_1, where j ranges over the indices $j \geq 1$ that do not occur as the pivot of a column of q. Although $\mathrm{im}\, f_1' \subseteq \mathrm{im}\, f_1$ may be a proper inclusion, f_1' still satisfies $\mathrm{colim}\,\mathrm{im}\, f_1' = \mathrm{colim}\,\mathrm{im}\, f_1$ and $\mathrm{rg}^{f_1'} = \mathrm{rg}^{f_1}$. This is enough to ensure that `BireduceF`(f_1) and `Bireduce`(f_2') correctly compute matrices k and $\tilde{f}_1$ that fit into (4.52). □

Algorithm 14: A variant of Algorithm 12 with clearing.

Input: A valid graded $l \times m$-matrix $B\colon F \to G'$, a reduced valid graded $m \times n$-matrix A such that $BA = 0$

Output: An injective valid graded $m \times n'$-matrix $B'\colon F\colon \to F$ representing a basis of $F' = [\operatorname{colim} \operatorname{im} B]_G$, and a valid graded matrix $W\colon F \to F'$ such that $B = B'W$

```
function BireduceFC($B, A$):
    $V \leftarrow W \leftarrow 1 \in k^{n \times n}$                     ▷ (inverse) reduction matrix
    $r \leftarrow 0 \in \mathbf{N}^n$                              ▷ assignment pivot row ↦
    foreach $k = 1, \dots, n$ do                         column for A
        if $A_k \neq 0$ then
            $j \leftarrow \operatorname{piv} A_k$
            $r_j \leftarrow k$
(a)         $B_j \leftarrow 0$
    $B', W \leftarrow$ BireduceF($B$)
(b) foreach nonzero $j \in \{\operatorname{piv} A_k\}$ in ascending order do    ▷ reconstruct W for cleared
(c)     $W_j \leftarrow 0$                                     columns of B
        $W_j \leftarrow -\frac{1}{A_{r_j r_j}} W A_{r_j}$
    $\mathrm{rg}^W \leftarrow \mathrm{cg}^{B'}$
    $\mathrm{cg}^W \leftarrow \mathrm{cg}^B$
    return $B', W$
```

Clearing Algorithm 13 still has the shortcoming that when calling $\texttt{BireduceF}(\tilde{D})$, it spends some time on reducing columns of $\tilde{D}$ to zero that can be predicted. In a last step, we show an algorithm that avoids this. As before, let $B\colon F \to G$ be a valid graded matrix. Our goal is to compute a basis $B'\colon F' \to G$ of $F' := [\operatorname{colim} \operatorname{im} B]_G$, and a valid graded matrix $W\colon F \to F'$ such that $B = B'W$.

Assume that we know $BA = 0$ for some reduced matrix A. Let $j_1 < \cdots < j_t$ be the pivot indices of the non-zero columns of A, and let B'' be the matrix obtained from B by setting $B_{j_1}, \dots, B_{j_t}$ to zero. Then B and B'' generate two submodules $\operatorname{im} B'' \subseteq \operatorname{im} B \subseteq G$, such that $\operatorname{colim} \operatorname{im} B'' = \operatorname{colim} \operatorname{im} B \subseteq \operatorname{colim} G$. Let $B', U = \texttt{BireduceF}(B'')$. Then B' represents a basis of $[\operatorname{colim} \operatorname{im} B]_G$, and U satisfies $B'' = B'U$. We obtain the commutative diagram

$$\begin{tikzcd}
& F' \arrow[dr, "B'"] & \\
F \arrow[ur, hook, "W"] \arrow[rr, "B"] & & G, \\
& F'' \arrow[ul, "J"] \arrow[ur, "B''"'] \arrow[uu, hook, "U", crossing over] &
\end{tikzcd}$$

where $J_j = \left\{\begin{smallmatrix} 0 & \text{if } j = j_s \text{ for some } s, \\ e_j & \text{otherwise} \end{smallmatrix}\right.$. We extend Algorithm 12 such that it computes the matrix W from U and A.

Proposition 4.4.4. *Let F, G be free modules, $B\colon F \to G$ be a valid graded matrix and A be a matrix such that $\mathsf{u}(B)A = 0$. Then Algorithm 14 computes a valid graded matrix $B'\colon F' \to G$ representing a basis of $F' := [\operatorname{colim} \operatorname{im} B]_G$, and a valid graded matrix $W\colon F \to F'$ such that $B = B'W$.*

The following will be proven after the proposition:

Lemma 4.4.5. *Let B, B' be valid graded matrices such that $\mathrm{cg}_j^{B'} = \bigvee_{B'_{ij} \neq 0} \mathrm{rg}_j^{B'}$ for all j. If W is a graded matrix with row and column grades $\mathrm{cg}^W = \mathrm{cg}^B$ and $\mathrm{cg}^W = \mathrm{rg}^{B'}$ such that $\mathsf{u}(B) = \mathsf{u}(B')\mathsf{u}(W)$, then W is valid.*

Proof of Proposition 4.4.4. Since $BA = 0$, setting $B_j \leftarrow 0$ in line (a) if j occurs as the pivot index of some column of A does not change the image of B; so $\operatorname{im} B'' = \operatorname{im} B$. It follows from Proposition 4.4.2 that B' represents a basis of $[\operatorname{colim} \operatorname{im} B]_G$.

Let W denote the state of W after line (a), and W' be the state of W when the algorithm terminates. It remains to show that W' is valid and satisfies $B = B'W'$. Let $q = \{j_1 < \cdots < j_t\} := \{\operatorname{piv} A_k \mid A_k \neq 0\}$. If $j \in q$, then `BireduceF`(B) never performs a column addition from B_j to another column, and thus also no row addition from W_{j*} to any other row of W. This implies that $B_j = B'_j W'_j$ for all $j \notin q$, and that W'_j is valid in this case.

It remains to show the same for $j \in q$. Let $W^{(0)} = W, \dots, W^{(t)} = W'$ be the intermediate values the matrix W takes during the loop in line (b). Note that $W^{(0)}_j = 0$ for all $j \in q$ because of line (c). We show by induction on $s = 1, \dots, t$ that for all $j \notin \{j_s, \dots .j_t\}$, we have $B_j = B'_j W^{(s)}_j$. For $s = 1$, there is no $j < j_1$ in q, some the statement holds in this case. Assume by induction that for some $s \geq 1$, we have $B_j = BW^{(s-1)}_j$ if $j \notin \{j_s, \dots, j_t\}$. Note that $W^{(s-1)}_j = 0$ for all $j \in \{j_s, \dots, j_t\}$. Let $k = r_{j_s}$ be the column index of A with $j_s = \operatorname{piv} A_k$. We obtain that

$$\begin{aligned} 0 = BA_k = B_{j_s} A_{j_s k} + \sum_{j<j_s} B_j A_{jk} &\overset{(i)}{=} B_{j_s} A_{j_s k} + \sum_{j<j_s} B' W^{(s-1)}_j A_{jk} \\ &\overset{(ii)}{=} B_{j_s} A_{j_s k} + \sum_{j \leq j_s} B' W^{(s-1)}_j A_{jk} \overset{(iii)}{=} B_{j_s} A_{j_s k} + B' W^{(s-1)} A_k, \end{aligned}$$

where (i) holds by the induction hypothesis, (ii) holds because $W^{(s-1)}_{j_s} = 0$, and (iii) holds because j_s is the pivot of A_k. Therefore, we get that

$$B_{j_s} = -\frac{1}{A_{j_s k}} B' W^{(s-1)} A_k,$$

so setting

$$W^{(s)}_j = \begin{cases} W^{(s-1)}_j & \text{if } j \neq j_s, \\ -\frac{1}{A_{j_s,k}} W^{(s-1)} A_k & \text{otherwise} \end{cases}$$

satisfies $B_j = B' W^{(s)}_j$ for all $j \notin \{j_{s+1}, \dots, j_t\}$. Inductively, we obtain that $B = B'W'$. It follows from Lemma 4.4.5 that W' is valid. □

Proof of Lemma 4.4.5. We have

$$\operatorname{cg}^W_j = \operatorname{cg}^B_j \overset{(*)}{\geq} \bigvee_{B_{ij} \neq 0} \operatorname{rg}^B_i \overset{(\dagger)}{\leq} \bigvee_{W_{lj} \neq 0} \bigvee_{B'_{il} \neq 0} \operatorname{rg}^B_i \overset{(\ddagger)}{\leq} \bigvee_{W_{lj} \neq 0} \operatorname{cg}^{B'}_l = \bigvee_{W_{lj} \neq 0} \operatorname{rg}^W_l,$$

where $(*)$ holds because B is valid, $(\dagger)$ holds because $\mathsf{u}(B) = \mathsf{u}(B')\mathsf{u}(W)$ and $(\ddagger)$ holds because B' is valid. Now, $(\ddagger)$ is an equality by assumption, $(\dagger)$ is an equality because B' is bireduced. Then we obtain that $\operatorname{cg}^W_j \geq \bigvee_{W_{lj} \neq 0} \operatorname{rg}^{W'}_l$, so W is valid. □

Remark. We provide another, more conceptual proof of Lemma 4.4.5. Let F, F' and M be free modules such that $B \colon F' \to M$ and $B' \colon F' \to M$, and let $\vec{z} = \operatorname{rg}^B = \operatorname{rg}^{B'}$, $N := \operatorname{im} B$ and $N' := \operatorname{im} B'$. For any module L and any morphism f of modules, let $\bar{L} := \operatorname{colim} F$ and $\bar{f} := \operatorname{colim} f$. From $\mathsf{u}(B) = \mathsf{u}(B')\mathsf{u}(W)$, it follows that $\bar{N} \subseteq \bar{N}'$. From the definition of $[-]_M$, it follows that there is an injective map $n \colon N = [\bar{N}]_M \subseteq [\bar{N}']_M = N'$. By Lemma 3.4.6, the matrix

B' represents a basis of $[V']_M$, so $\operatorname{im} B \subseteq [V]_M \subseteq [V']_M = \operatorname{im} B'$. We obtain a commutative diagram

$$\begin{tikzcd}
 & F \arrow[rr, two heads, "p"] \arrow[dl] \arrow[dd, dashed, "f"'] \arrow[drrrr, bend left=50, "B"] & & N \arrow[dl] \arrow[dd, hook, "n"] \arrow[drr, hook, "i"] & & \\
\bar{F} \arrow[rr, two heads, "\bar{p}"] \arrow[dd, "\mathsf{u}(W)"'] \arrow[dd, dashed, shift left, "\bar{f}"] \arrow[drrrr, bend left=40, "\mathsf{u}(B)"] & & \bar{N} \arrow[dd, hook, "\bar{n}"'] \arrow[drr, hook, "\bar{\imath}"] & & & M \arrow[dl, "\eta_M"] \\
 & F' \arrow[rr, "\cong", "p'"'] \arrow[dl, "\cong"] \arrow[urrrr, bend right=50, "B'"'] & & N' \arrow[dl] \arrow[urr, hook, "i'"] & \bar{M}. & \\
\bar{F}' \arrow[rr, "\cong", "\bar{p}'"'] \arrow[urrrr, bend right=50, "\mathsf{u}(B)"'] & & \bar{N}' \arrow[urr, hook, "\bar{\imath}'"'] & & &
\end{tikzcd}$$

Because there is an isomorphism $p' \colon F' \to N'$, there exists a unique morphism $f \colon F \to F'$ such that $np = p'f$. Because $\bar{p}'$ is an isomorphism, the colimit $\bar{f}$ is the unique morphism such that $\bar{n}\bar{p} = \bar{p}'\bar{f}$. From the assumption, we obtain that

$$\bar{\imath}'\bar{n}\bar{p} = \bar{\imath}\bar{p} = \mathsf{u}(B) = \mathsf{u}(B')\mathsf{u}(W) = \bar{\imath}'\bar{p}'\mathsf{u}(W).$$

Injectivity of $\bar{\imath}'$ implies that $\bar{n}\bar{p} = \bar{p}'\mathsf{u}(W)$. Uniqueness of $\bar{f}$ implies that $\mathsf{u}(W) = \bar{f}$ and thus $W = f$. In particular, W is the representative matrix of a morphism of free modules and thus valid.

We can use Algorithm 14 to establish another algorithm that computes a minimal free resolution of $H^{d+1}(K, K_*)$:

Proposition 4.4.6. *Let K_* and $D^\bullet$ be as above, and let $n_d := |K^d|$. Then for each $d \geq 0$, Algorithm 15 computes a minimal free resolution of $H^{d+1}(K, K_*)$.*

Algorithm 15: Relative cohomology algorithm II

Input: Coboundary matrices $D^1, D^2, \ldots$ representing $C^\bullet(K, K_*)$ with $H^\bullet(K) = 0$
Output: Matrices representing a minimal free resolution of $H^{d+1}(K, K_*)$ for $d > 0$.

```
    $k \leftarrow$ Ker$((D^1\Pi_0, D^1)\colon C_1^0 \to C_0^1)$
    for $d = 0, 1, \ldots$ do
        $h_2 \leftarrow \begin{pmatrix} k \\ (0, D^{d+1})k \end{pmatrix} : H^{d+1}(K, K_*)_1 \to C_1^d \oplus C_2^{d+1}$
        $\tilde{D} \leftarrow \begin{pmatrix} \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 & \Pi_{d+1}^{-1} \\ 0 & D^{d+1} & E_{d+1} \end{pmatrix} : C_1^d \oplus C_2^{d+1} \to C_1^{d+1}$
        Let $R = V\mathsf{u}(h_2)$ be reduced        ▷ standard reduction (Algorithm 1) of ungraded matrices
(a)     if $d = 0$ then $(k, h_1) \leftarrow$ BireduceF$(\tilde{D})$
(b)     else if $d < d_{max}$ then $(k, h_1) \leftarrow$ BireduceFC$(\tilde{D}, R)$
        else
(c)         for $j \in \{\operatorname{piv} A_k \mid A_k \neq 0\}$ do $\tilde{D}_j \leftarrow 0$
            $h_1 \leftarrow$ Bireduce$(\tilde{D})$
        yield MinimizeCpx$(h_1, h_2)$        ▷ resolution of $H^{d+1}(K, K_*)$
```

Proof. We have to show that the matrices h_2, h_1 computed in the dth iteration of the main for-loop in Algorithm 15 represent the morphisms h_2^{d+1}, h_1^{d+1} from the free resolution (4.52) of $H^{d+1}(K, K_*)$. For $d = 0$, this follows directly from Proposition 4.4.2. Because $\tilde{D}h_2$ is a free resolution of $\operatorname{coker} \tilde{D}$, we have $\tilde{D}h_2 = 0$. Then also $\mathsf{u}(\tilde{D})R = 0$ as ungraded matrices, because R is obtained from h_2 by (ungraded) standard matrix reduction. Setting the columns $\tilde{D}_j$ to zero for $j = \operatorname{piv} R_k$ thus does not change the image of $\tilde{D}$. Then Lemma 4.4.5 and Proposition 4.4.4 imply that h_1 and h_2 form a minimal free resolution of $H^{d+1}(K, K_*)$. □

4.5 Computational shortcomings

Recall that, to compute a free resolution of $H^d(K_*)$ as described in 4.2.6, we have to compute, i.a., a matrix κ^d representing the inclusion of $\ker(\delta_0^{d+1}, -c_1^{d+1})$ and solve the factorization problem

$$\kappa^d h_1^d = \begin{pmatrix} -\delta_0^d & c_1^d & 0 \\ 0 & \delta_1^{d+1} & c_2^{d+1} \end{pmatrix} \tag{4.54}$$

for h_1^d. Let $n_d := |K_*^d|$, and let D^d represent the coboundary morphism δ^d of the cochain complex $C^d(K)$ of vector spaces. By definition, $(\delta_0^{d+1}, -c_1^{d+1})$ is represented by a graded matrix with the underlying block matrix

$$\left(\begin{array}{c|cc} D^{d+1} & E_{d+1} & \Pi_{d+1} \end{array}\right) : C_0^d \oplus C_1^{d+1} \to C_0^{d+1} \tag{4.55}$$

of size $n_{d+1} \times (n_d + 2n_{d+1})$, and its kernel has rank $n_d + n_{d+1}$. Similarly, the right hand side of (4.54) is represented by the block matrix

$$\left(\begin{array}{c|cc|c} -D^d & E_d & \Pi_d & 0 \\ \hline 0 & \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 & \Pi_{d+1}^{-1} \\ 0 & 0 & D^{d+1} & -E_{d+1} \end{array}\right) : C_0^{d-1} \oplus C_1^d \oplus C_2^{d+1} \to C_0^d \oplus C_1^{d+1}$$

of shape $(n_d + 2n_{d+1}) \times (n_{d-1} + 2n_d + n_{d+1})$. In experiments with a prototype implementation not reported here, it has been observed that the mere size of these two matrices causes high running times. Also, it has been observed that computing the kernel of (4.55) using `Ker`() causes considerable fill-up of matrix columns (see also [17]), which further increases the runtime.

When working with relative cohomology, the situation is slightly better. Namely, as described in 4.2.8, to compute $H^{d+1}(K, K_*)$, one has to compute the kernel of the morphism $(c_1^{d+2} \delta_1^{d+2})$, which is represented by the matrix

$$(D^{d+2}, D^{d+2}\Pi_{d+1}) \colon C_2^{d+1} \to C_0^{d+2}$$

of size $n_{d+2} \times 2n_{d+1}$. As described above, this can be avoided using one of Algorithms 13 and 15. When using Algorithm 13, the expensive step is to run **`BireduceF`**() (Algorithm 12) on the block matrix

$$\left(\begin{array}{cc|c} \Pi_{d+1}^{-1} D^{d+1} \Pi_d & 0 & \Pi_{d+1}^{-1} \\ 0 & D^{d+1} & E_{d+1} \end{array}\right) : C_1^d \oplus C_2^{d+1} \to C_1^{d+1} \tag{4.56}$$

of size $2n_{d+1} \times (2n_d + n_{d+1})$. Applying **`BireduceF`**() on this matrix has the downside that it needs to reduce many columns to zero, which is why we introduced Algorithm 15 that builds upon **`BireduceFC`**() (Algorithm 14). The problem with this approach is the following. Let $\tilde{D}$ denote the matrix from (4.56) as in Algorithm 15. Then by the design of Algorithm 15, it is necessary that **`BireduceFC`**($\tilde{D}$) outputs both the bireduced reduced matrix $R = \tilde{D}V$ and the reduction matrix V. However, as a clearing strategy, **`BireduceFC`**($\tilde{D}$) set certain columns of R to zero directly. As described above, if a column R_j has been set to zero by clearing, one has to do some computations to obtain the corresponding column V_j, which is done in Algorithm 14, line (b). This process cannot be parallelized in a simple way. Experiments have shown that the loop in Algorithm 14, line (b) consumes all runtime benefit arising from clearing.

For a comparison with Chapter 3, we note that the situation is different in Algorithm 11. Here, the situation is that when $\texttt{Bireduce}(D^{d+1})$ is used to compute a bireduced matrix $R = D^{d+1}V$ for some V, then Algorithm 11 only needs the matrix R for further computations, but not the reduction matrix V. Consequently, it is not necessary to compute the column V_j of V if $R_j = 0$ has been obtained through clearing in this case.

4.6 Absolute and relative cohomology do not determine each other

To finish this section, we present an example that shows that in two-parameter persistence, $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ need not determine each other, unless K is acyclic. This is analogous to Section 3.6, where we provided an example that shows the analogous statement for $H^\bullet(K_*)$ and $H^\bullet(N^\bullet(K_*))$. This is different from one-parameter persistence: recall that in this case, $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ determine each other uniquely up to isomorphism even if K is not acyclic; see Corollary 2.2.11. Specifically, we show:

Theorem B(b). *There exist one-critically two-parameter filtered simplicial complexes K_*, L_* and M_*, such that*

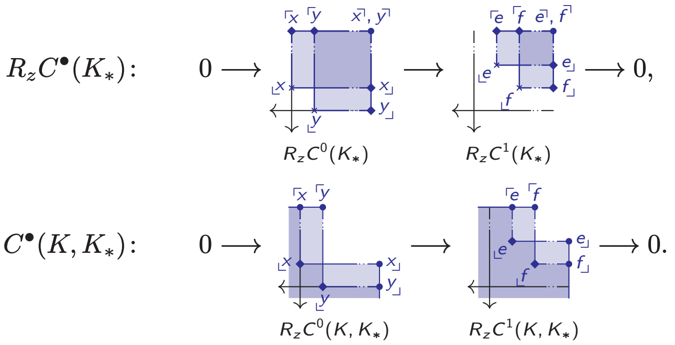
(i) $H^\bullet(K, K_*) \cong H^\bullet(L, L_*)$, *but* $H^\bullet(K_*) \ncong H^\bullet(L_*)$, *and*

(ii) $H^\bullet(K_*) \cong H^\bullet(M_*)$, *but* $H^\bullet(K, K_*) \ncong H^\bullet(M, M_*)$.

The colimits K, L and M are not acyclic.

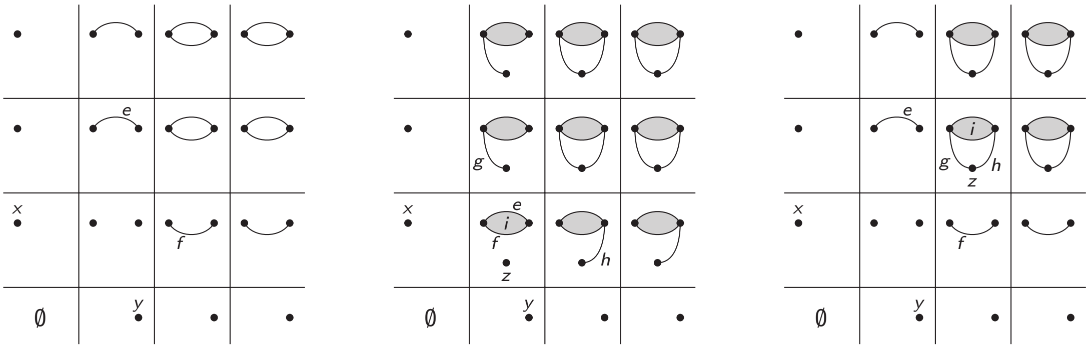
Proof. We use the same complexes as in Section 3.6, but leave out the last 2-cell; see Figure 4.9. Again, we compute non-reduced cohomology with coefficients in $\mathbf{F}_2$. An example for reduced cohomology can be constructed analogously.

Let $z > g(\sigma)$ for all $\sigma \in K_*$, L_* and M_*. For K_*, we get



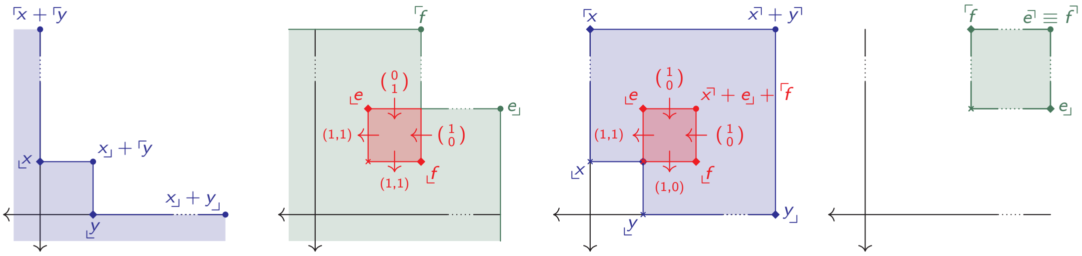
We compute generators and relations of $H^\bullet(K_*)$ and $H^\bullet(K, K_*)$ as in Section 4.3 and obtain the minimal free presentations

$$\begin{aligned} R_zH^0(K, K_*) &= \frac{\langle \ulcorner x + \ulcorner y, x_\lrcorner + \ulcorner y, x_\lrcorner + y_\lrcorner \rangle}{\langle \ulcorner x + \ulcorner y \overset{\llcorner x}{\equiv} x_\lrcorner + \ulcorner y \overset{\llcorner y}{\equiv} x_\lrcorner + y_\lrcorner \rangle}, \quad & R_zH^1(K, K_*) &= \frac{\langle e_\lrcorner, \ulcorner f \rangle}{\langle e_\lrcorner \overset{x_\lrcorner}{\equiv} \ulcorner f, e_\lrcorner \overset{\ulcorner y}{\equiv} \ulcorner f \rangle}, \\ R_zH^0(K_*) &= \frac{\langle x^\urcorner + y^\urcorner, x^\urcorner + e_\lrcorner + \ulcorner f \rangle}{\langle \ulcorner x + \ulcorner y, x_\lrcorner + y_\lrcorner \rangle}, & R_zH^1(K_*) &= \frac{\langle e^\urcorner \equiv f^\urcorner \rangle}{\langle f_\lrcorner, f_\lrcorner \rangle}, \end{aligned} \tag{4.57}$$



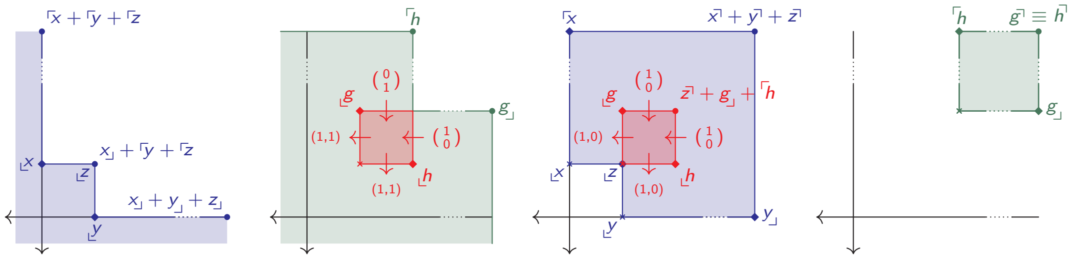
(a) K_* **(b)** L_* **(c)** M_*

Figure 4.9: The filtered simplicial complexes from the proof of Theorem B(b).



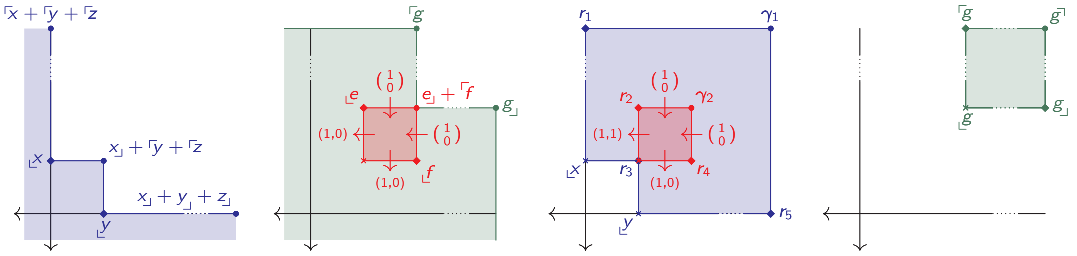
(a) $H^0(K, K_*)$ **(b)** $H^1(K, K_*)$ **(c)** $H^0(K_*)$ **(d)** $H^1(K_*)$

Figure 4.10: The relative and absolute cohomology modules (4.57) of K_* from Figure 4.9a.



(a) $H^0(L, L_*)$ **(b)** $H^1(L, L_*)$ **(c)** $H^0(L_*)$ **(d)** $H^1(L_*)$

Figure 4.11: The relative and absolute cohomology modules (4.58) of the filtered complex L_* from Figure 4.9b.



(a) $H^0(M, M_*)$ **(b)** $H^1(M, M_*)$ **(c)** $H^0(M_*)$ **(d)** $H^1(M_*)$

Figure 4.12: The relative and absolute cohomology modules (4.59) of the filtered complex M_* from Figure 4.9c.

see Figure 4.10. All four modules are indecomposable, and all higher homology modules are zero. Similarly, we obtain the minimal presentations

$$
\begin{aligned}
H^0(L, L_*) &= \frac{\langle \ulcorner x + \ulcorner y + \ulcorner z,\, x\lrcorner + \ulcorner y + \ulcorner z,\, x\lrcorner + y\lrcorner + z\lrcorner \rangle}{\langle \ulcorner x + \ulcorner y + \ulcorner z \overset{\llcorner x}{\equiv} x\lrcorner + \ulcorner y + \ulcorner z \overset{\llcorner y, \llcorner z}{\equiv} x\lrcorner + y\lrcorner + z\lrcorner \rangle}, &
H^1(L, L_*) &= \frac{\langle e\lrcorner, \ulcorner f \rangle}{\langle e\lrcorner \overset{\delta z\lrcorner}{\equiv} \ulcorner f,\, e\lrcorner \overset{\delta \ulcorner z}{\equiv} \ulcorner f \rangle}, \\
H^0(L_*) &= \frac{\langle x\urcorner + y\urcorner + z\urcorner \rangle}{\langle \ulcorner x + \ulcorner y + \ulcorner z,\, x\lrcorner + y\lrcorner + z\lrcorner \rangle} \oplus \frac{\langle z\urcorner + e\lrcorner + \ulcorner f \rangle}{\langle \ulcorner z + \ulcorner e,\, z\lrcorner + f\lrcorner \rangle}, &
H^1(L_*) &= \frac{\langle e\urcorner \equiv f\urcorner \rangle}{\langle e\lrcorner, \ulcorner f \rangle},
\end{aligned}
\tag{4.58}
$$

and

$$
\begin{aligned}
H^0(M, M_*) &= H^0(L, L_*), &
H^1(M, M_*) &= \frac{\langle e\lrcorner + \ulcorner f \rangle}{\langle \ulcorner y + \llcorner e,\, x\lrcorner + \ulcorner f \rangle} \oplus \frac{\langle \ulcorner g, g\lrcorner \rangle}{\langle \llcorner g \rangle}, \\
H^0(M_*) &= \frac{\langle \gamma_1, \gamma_2 \rangle}{\langle \gamma_1 \overset{r_1, r_3, r_5}{\equiv} 0,\, \gamma_1 \overset{r_2}{\equiv} \gamma_2,\, \gamma_2 \overset{r_2}{\equiv} 0 \rangle}, &
H^1(M_*) &= \frac{\langle g\urcorner \rangle}{\langle \ulcorner g, g\lrcorner \rangle},
\end{aligned}
\tag{4.59}
$$

with

$$
\begin{aligned}
\gamma_1 &= x\urcorner + y\urcorner + z\urcorner, & \gamma_2 &= x\urcorner + e\lrcorner + \ulcorner f + \ulcorner g, \\
r_1 &= \ulcorner x + \ulcorner y + \ulcorner z \colon g_1 \equiv 0, & r_2 &= \ulcorner y + \ulcorner z + \llcorner e \colon g_1 \equiv g_2, \\
r_3 &= x\lrcorner + \ulcorner y + \ulcorner z \colon g_1 \equiv 0, & r_4 &= x\lrcorner + \llcorner f + \llcorner g \colon g_2 \equiv 0, \\
r_5 &= x\lrcorner + y\lrcorner + z\lrcorner \colon g_1 \equiv 0;
\end{aligned}
$$

see Figures 4.11 and 4.12.

These are minimal presentations. All other cohomology modules are zero. Comparing Figures 4.10 to 4.12 shows that $H^d(K, K_*) \cong H^d(L, L_*)$ and $H^d(L_*) \cong H^d(M_*)$ for all d. Nevertheless, $H^0(K_*) \ncong H^0(L_*)$ and $H^1(K, K_*) \ncong H^1(M, M_*)$, because one is indecomposable while the other is not. This finishes the construction. □

To finish this section, we show a two-parameter filtered cell complex Q_* such that the short exact sequences

$$
\begin{aligned}
0 \to \operatorname{coker} p^d \to R_z H^d(Q_*) \to \ker p^{d+1} \to 0, \\
0 \to \operatorname{coker} i^d \to R_z H^{d+1}(Q, Q_*) \to \ker i^{d+1} \to 0
\end{aligned}
\tag{4.60}
$$

induced by the long exact sequence

$$
\begin{tikzcd}
0 \arrow[r] & R_z H^0(Q, Q_*) \arrow[r, "p^0"] & R_z H^0(Q) \arrow[r, "i^0"] & R_z H^0(Q_*) \arrow[dll, rounded corners, "\delta"] \\
& R_z H^1(Q, Q_*) \arrow[r, "p^0"] & R_z H^1(Q) \arrow[r, "i^1"] & R_z H^1(Q_*) \arrow[dll, rounded corners, "\delta"] \\
& R_z H^2(Q, Q_*) \arrow[r, "p^1"] & R_z H^2(Q) \arrow[r, "i^2"] & R_z H^2(Q_*) \arrow[r] & \cdots
\end{tikzcd}
\tag{4.61}
$$

never split for any d. Namely, let Q_* be the cell complex with cellular chain complex

$$
C_\bullet(Q_*)\colon\ \cdots \to \langle \rho_3, \sigma_3, \tau_3 \rangle \xrightarrow{\left(\begin{smallmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{smallmatrix}\right)} \langle \rho_2, \sigma_2, \tau_2 \rangle \xrightarrow{\left(\begin{smallmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{smallmatrix}\right)} \langle \rho_1, \sigma_1, \tau_1 \rangle \xrightarrow{\left(\begin{smallmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{smallmatrix}\right)} \langle \rho_0, \tau_0 \rangle \to 0,
$$

one-critically $\mathbf{Z}^2$-filtered with

$$
g(\rho_d) = (2d, 2d+2), \qquad g(\sigma_d) = (2d+1, 2d+1), \qquad g(\tau_d) = (2d+2, 2d)
$$

for all d. We obtain the minimal free presentation matrices

$$R_zH^0(Q,Q_*) = \operatorname{coker}\left[\begin{array}{rcc} & \llcorner\rho_0 & \llcorner\tau_0 \\ \ulcorner\rho_0+\ulcorner\tau_0 & 1 & \\ \rho_{0\lrcorner}+\ulcorner\tau_0 & 1 & 1 \\ \rho_{0\lrcorner}+\tau_{0\lrcorner} & & 1 \end{array}\right], \tag{4.62}$$

$$R_zH^d(Q,Q_*) = \operatorname{coker}\left[\begin{array}{rccc} & \llcorner\rho_d+\ulcorner\tau_{d-1} & \llcorner\sigma_d & \rho_{d-1\lrcorner}+\llcorner\tau_d \\ \ulcorner\sigma_d & 1 & 1 & \\ \rho_{d\lrcorner}+\ulcorner\tau_d & 1 & & 1 \\ \sigma_{d\lrcorner} & & 1 & 1 \end{array}\right], \tag{4.63}$$

$$R_zH^d(Q_*) = \operatorname{coker}\left[\begin{array}{rccccc} & \ulcorner\rho_d+\ulcorner\tau_d & \llcorner\rho_{d+1}+\ulcorner\tau_d & \llcorner\sigma_{d+1} & \llcorner\tau_{d+1}+\rho_{d\lrcorner} & \rho_{d\lrcorner}+\tau_{d\lrcorner} \\ \rho_{d\urcorner}+\rho_{d+1\lrcorner}+\ulcorner\sigma_{d+1}+\ulcorner\tau_{d+1} & & 1 & 1 & & \\ \rho_{d\urcorner}+\tau_{d\urcorner} & 1 & 1 & & 1 & 1 \\ \rho_{d\urcorner}+\rho_{d+1\lrcorner}+\sigma_{d+1\lrcorner}+\ulcorner\tau_{d+1} & & & 1 & 1 & \end{array}\right] \tag{4.64}$$

for all $d > 0$ and $d \geq 0$, respectively. The rows and columns of these matrices are decorated by the representatives in $C^{\bullet}{}_{\bullet}$. These are graded matrices, where each row and column has as grade the join of the grades of the summands in the representative attached to it. The long exact sequence (4.61) is illustrated in Figure 4.13. Each of the modules $H^d(Q,Q_*)$, $H^d(Q)$ and $H^d(Q_*)$ is indecomposable. In particular, the short exact sequences (4.60) cannot split for any d.

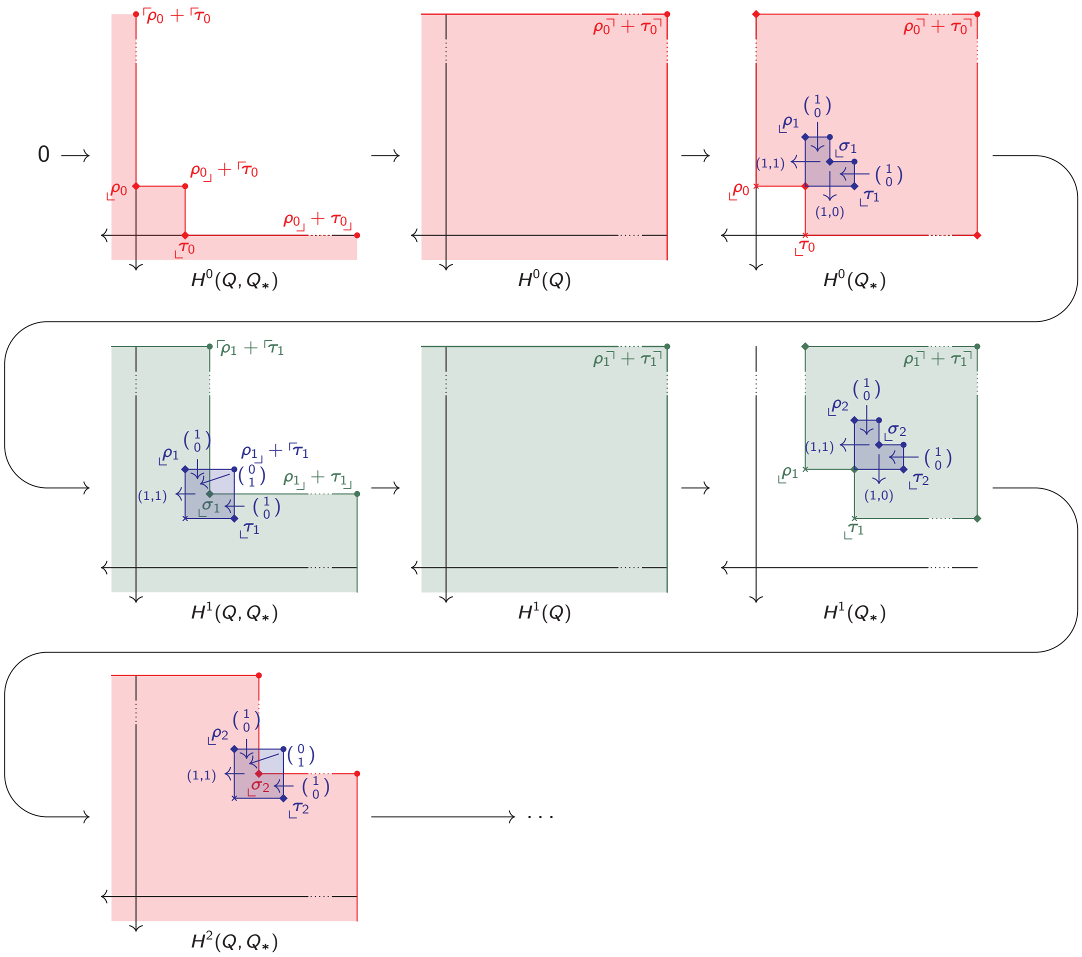

0
$H^0(Q, Q_*)$
$H^0(Q)$
$H^0(Q_*)$
$H^1(Q, Q_*)$
$H^1(Q)$
$H^1(Q_*)$
$H^2(Q, Q_*)$
$\ulcorner\rho_0 + \ulcorner\tau_0$
$\llcorner\rho_0$
$\rho_{0\lrcorner} + \ulcorner\tau_0$
$\rho_{0\lrcorner} + \tau_{0\lrcorner}$
$\llcorner\tau_0$
$\rho_0\urcorner + \tau_0\urcorner$
$\llcorner\rho_1$
$\llcorner\sigma_1$
$\llcorner\tau_1$
$\llcorner\tau_2$
$\llcorner\rho_2$
$\llcorner\sigma_2$
$\ulcorner\rho_1 + \ulcorner\tau_1$
$\rho_{1\lrcorner} + \ulcorner\tau_1$
$\rho_{1\lrcorner} + \tau_{1\lrcorner}$
$\rho_1\urcorner + \tau_1\urcorner$
(1,1)
(1,0)
$\binom{1}{0}$
$\binom{0}{1}$
. . .

Figure 4.13: The long exact cohomology sequence (4.61) of the complex Q_*. The diagrams depict the minimal free presentations (4.62) to (4.64) of the respective cohomology modules. The symbols •, ◆ and × denote the grades of the generators, relations and 2-syzygies of the module. The labels stand for the simplex that determines the grade of the generator, relation or syzygy. Where two colors overlap, the components of the respective module are two-dimensional. The small matrices indicate structure maps into and out of these regions. Each of the modules is indecomposable because it contains an indecomposable quiver representation.

Chapter 5

Implementation and experiments

To show that computing two-parameter persistent cohomology is feasible, we provide a C++ implementation of our approach for the computation of two-parameter persistent homology and cohomology from Chapter 3. In this section, we provide some details about and experimental results obtained with this implementation.

5.1 Implementation

Our software `2pac` (*2-**pa**rameter **c**ohomology*), which is publicly available at [94], implements the two-parameter cohomology algorithm (Algorithm 11) from Chapter 3 and the homology algorithm (Algorithm 9) from Section 2.4.

We do not include the absolute and relative cohomology algorithms from Chapter 4, because earlier experiments with a prototype implementation did not show any performance benefits over the cohomology algorithm.

The code computes all resolutions with coefficients in $\mathbf{F}_2$. The software accepts as input either a sequence of matrices, or a file containing a filtration and a distance matrix of a point cloud. Both kinds of input are read as a binary format described in the file `README.md`. For the interested reader, we provide an overview over the relevant files and classes and explain technical details in the implementation of algorithms presented in this thesis.

5.1.1 Retrieval, building, using

The software can be retrieved from the git repository at [94]. Building the software requires a reasonably recent version of Boost (including `Boost::program_options`) and OpenMP to be installed. To build the software, follow the instructions in `README.md` in the repository. The software has been successfully built on Mac OS using clang++ 15.0.7, and on Ubuntu Linux using g++ 11.2.0.

To run the software on one of the samples, run the command `./2pac -f samples/[file↩ name]`. This computes minimal free resolutions of $H^{d+2}(N^\bullet(K_*))$ and $H_d(K_*)$, where K_* is the density-Rips complex on the filtered point cloud in `[filename]`. Run `./2pac -help` for an overview of the command line arguments. See `README.md` for a description of the binary file format.

Jupyter notebook `EXAMPLE.ipynb` may be convenient to produce the input data, call `2pac`, and visualize the output. This notebook can be used to generate point clouds sampled from different spaces, choose a density function and bandwidth parameter interactively, write the filtered point cloud to disk in the required binary format, invoke `2pac` and read its output, and visualize the computed minimal free resolutions. The notebook can also be used to read a file in scc2020 format [97] and convert it to the binary format.

5.1.2 Organization of the source code

For the reader who is interested in implementation details, we provide an overview of the source code. The code is grouped in several pairs of corresponding source and header files. A documentation of the classes, functions and files can be generated by running `doxygen` in the folder. The documentation can then be found in the sub-folder `documentation/index.html`. We give a short overview of the files and classes of our code. See the documentation in the header files for details.x

`2pac.cpp` Entry point. The function `main` reads an input file (which describes a filtered point cloud), sets up its function-Rips complex K_*, applies preprocessing filters (such as chunk preprocessing) according to command line arguments, and computes minimal free resolutions of $H^{d+2}(N^\bullet(K_*))$ or $H_d(K_*)$. For each dimension, the computation finishes by printing the Betti numbers. During the computation, the program prints the runtimes for each of the steps of the algorithm.

`computation.hpp` Contains classes `Cohomology`, `Homology` and `RelativeCohomology` that implement the computation of minimal free resolutions of (co)homology as described in Algorithms 9, 11 and 13, respectively. To use these, construct an object of the respective type, and call its `operator()` on the (co)boundary matrices $D^1, D^2, \dots$ consecutively, which happens in `2pac.cpp`.

`matrices.hpp` Implements a class `GradedMatrix`, which represents a $\mathbf{Z}^2$-graded matrix. The underlying ungraded matrix is implemented by the class `SparseMatrix`, which represents a matrix as a list of columns. Each column is implemented as an object of type `ColumnType`, which can be one of `HeapColumn` and `ArrayColumn`. See Section 5.1.3 for details. The column type to be used is specified at compile time. The default is heaps; to use vectors, compile with `-DARRAY_MATRICES`.

`bireduce.hpp` implements Algorithm 10 in the function `extend_from_colimit`. The file defines a struct `Params`, which can be passed to `extend_from_colimit` to set some details of the algorithm. This is intended for testing; the defaults should be fine. The function `sparsify` implements Algorithm 22, which is used to reduce the density the matrix computed by Algorithm 10; see Section 5.1.5.

`complexes.hpp` provides the abstract class `Complex`, which represents either a chain or a cochain complex. A (co)chain complex has a method to yield the next (co)boundary matrix. The file provides the following classes deriving from `Complex`:

> **`Pointcloud`** reads a filtered point cloud from a binary file from disk, and assembles the coboundary matrices of the associated function-Rips complex; for generating the matrices, the class uses the combinatorial number system;
>
> **`MatricesFromFile`** reads a sequence of $\mathbf{Z}^2$-graded matrices from disk, and checks that they represent a cochain complex indeed.
>
> We also provide filters, which take as input a `std::unique_ptr` to a `Complex` and represent a new complex, obtained by applying some operation to the original one. For that reason, instances of `Complex` should always be created through `std::make_unique`. For example, `complexes.hpp` provides
>
> **`TransposeComplex`** converts a cochain complex to a chain complex and vice versa by taking the graded transpose of all its (co)boundary matrices.
>
> **`FlipGrades`** exchanges the first and second coordinate of all $\mathbf{Z}^2$-grades of the rows and columns of the (co)boundary matrices.
>
> **`AugmentComplex`** takes the augmented complex of a given (co)chain complex, used to compute reduced (co)homology.

Other filters, provided in separate files are the following:

`Cone.hpp` defines the classes `Cone` and `HBasisCone` deriving from `Complex`. The former makes the complex a cone for large filtration values by forming the simplicial cone as described in Section 3.7.1. The latter implements the strategy from Section 3.7.2; see Section 5.1.4 for implementation details.

`chunk.hpp` implements a class `Chunk` deriving from `Complex`, which implements the chunk preprocessing Algorithm 4 or Algorithm 5, depending on whether it is applied to a chain or cochain complex. Both variants call `minimize` on the (co)boundary matrices subsequently. It is possible to specify a largest dimension to which Algorithm 3 is supposed to be applied.

`factor.hpp`, `minimize.hpp` and **`lw.hpp`** implements Algorithms 3, 7 and 8, respectively.

`indirect.hpp` contains classes `IndirectColumn` and `IndirectMatrix`, which implement a matrix representation through maintaining a heap of pointers to another, underlying matrix. See Section 5.1.3.3 for details.

The other files contain utilities, helper and wrapper functions and classes, which should be self-explanatory.

5.1.3 Sparse matrices

Most algorithms presented in this thesis are matrix column reduction schemes. It is common in implementations of persistent homology to store matrices in a column sparse way; that is, a matrix is represented by a contiguous list (called a *vector*) of its columns in a way that allows accessing the nth column in constant time. In our implementation, we use a `std::vector<T>` of the C++ standard library, where the type `T` implements a matrix column. It has to provide efficient access to the pivot of a column, and an efficient implementation of the addition of two columns. Some operations, such as minimization (Algorithm 3), also require the efficient deletion of the pivot of a column and retrieval of the new pivot. In the context of one parameter persistent homology, various matrix formats have been tried in [12]. In our implementation, we provide support for the *vector* and the *heap* format, which we explain now in detail.

5.1.3.1 Matrix columns as vectors

In the *vector format*, a matrix column M_j is represented by a vector v of pairs (i, M_{ij}), one for each M_{ij} of M_j. The entries are ordered descendingly by the row index i. Each row index occurs at most once in v. The pivot of M_j is the first element in v. As implementation for v, we define the class `SkipVector`, which is a wrapper around `std::vector` that allows for the efficient removal of the pivot.

For coefficients in $\mathbf{F}_2$, it suffices to store the indices i of the non-zero entries. In this case, the addition of two columns v and v' is the symmetric difference of two ordered lists and can be performed in time $O(|v| + |v'|)$. We use the STL-implementation `std::set_symmetric_↩ difference` for this. For other coefficient fields, addition of two columns can be performed by an analogous algorithm.

5.1.3.2 Matrix columns as binary heaps

Let T be a set equipped with a total order. A *priority queue* Q of objects in T is a data structure containing elements of T that allows to efficiently retrieve the maximal element of Q ($\texttt{Top}(Q)$), remove it ($\texttt{PopMax}(Q)$), and insert a new element t into Q ($\texttt{Push}(Q, t)$). We use the STL class `std::priority_queue`. This class implements a priority queue as a *binary heap*, which stores its elements in a contiguous vector on which it maintains a specific indexing scheme. In particular, a descendingly ordered vector is also a binary heap.

Algorithm 16: Consolidation of binary heaps

Input: A binary heap Q.
Output: A consolidated, totally ordered binary heap Q'

```
if $Q = \emptyset$ then return $Q$
$Q' \leftarrow \emptyset$
$(i, \lambda) \leftarrow$ PopMax($Q$)
while $Q \neq \emptyset$ do
    $(i', \lambda') \leftarrow$ PopMax($Q$)
    if $i = i'$ then $\lambda \leftarrow \lambda + \lambda'$
    else
        if $\lambda \neq 0$ then Push($Q'$, $(i, \lambda)$)
        $(i, \lambda) \leftarrow (i', \lambda')$
if $\lambda \neq 0$ then Push($Q'$, $(i, \lambda)$)
return $Q'$
```

Algorithm 17: Pivot extraction from a binary heap

Input: A binary heap Q of pairs $(i, \lambda) \in \mathbf{N} \times k$.
Output: The pivot of the matrix column represented by Q.

```
if $Q = \emptyset$ then return 0
$(i, \lambda) \leftarrow$ PopMax($Q$)
while $Q \neq \emptyset$ do
    $(i', \lambda') \leftarrow$ PopMin($Q$)
    if $i = i'$ then $\lambda \leftarrow \lambda + \lambda'$
    else if $\lambda = 0$ then $(i, \lambda) \leftarrow (i', \lambda')$
    else
        Push($Q$, $(i', \lambda')$)
        Push($Q$, $(i, \lambda)$)
        return $i$
if $\lambda \neq 0$ then return $i$ else return 0
```

In *heap format*, a matrix column M_j is represented by a binary heap Q, whose entries are pairs (i, λ) for $i \in \mathbf{N}$ and $\lambda \in k$. The elements are ordered by i. The heap Q may contain multiple entries (i, λ) for the same row index i. The heap Q represents the matrix column M_j with entries $M_{ij} = \sum_{(i,\lambda)\in Q} \lambda$ for all i.

To retrieve the pivot index of M_j, one has to extract all entries from Q that have the same maximal value of i using PopMax(Q), and sum up their values λ. If they add up to a non-zero value M_{ij}, then this is the pivot entry of M; see Algorithm 17. To add to columns, one merges the representing heaps and reestablishes the heap condition.

We call the heap Q *consolidated* if it contains at most one entry (i, λ) for each i. A binary heap can be consolidated using Algorithm 16. Algorithm 16 produces the output Q' with entries in total order. In particular, this routine converts the heap representation of a matrix column into the vector representation.

5.1.3.3 Indirect representations

We explain another way to represent matrices, built around the vector representation. Let M be an $m \times n$-matrix with entries in $\mathbf{F}_2$ in vector format, where each column M_j is represented by the vector v_j. To efficiently implement column operations on the matrix M, we propose the following way to represent the intermediate steps of a column reduction algorithm.

Algorithm 18: Consolidation of the pivot of an indirect column

Input: A binary heap Q of entries $(v_{ki}, i, k))$
Output: A binary heap Q with a unique entry for which v_{ki} is maximal.

```
while $|Q| \geq 2$ do
    $(v_{ki}, i, k) \leftarrow$ PopMax($Q$)
    $(v_{k'i'}, i', k') \leftarrow$ PopMax($Q$)
    if $v_{ki} = v_{k'i'}$ then                                ▷ entries cancel
        if $i < |v_k|$ then Push($Q$, $(v_{k,i+1}, i+1, k)$)      ▷ $v_k$ still has entries
        if $i' < |v_{k'}|$ then Push($Q$, $(v_{k',i'+1}, i'+1, k')$)   ▷ $v'_{k'}$ still has entries
    else                                                     ▷ pivot established
        Push($Q$, $(v_{k'i'}, i', k')$)                         ▷ push back the last elements
        Push($Q$, $(v_{ki}, i, k)$)
        break
return $Q$
```

Algorithm 19: Converting an implicit matrix column to an explicit one

Input: A binary heap Q of entries (v_{ki}, i, k).
Output: A descendingly ordered vector v'.

```
$v' \leftarrow \emptyset$
while $Q \neq \emptyset$ do
    $(v_{ki}, i, k) \leftarrow$ PopMax($Q$)
    append $v_{ki}$ to $v'$.
```

An *indirect matrix* I with underlying matrix M consists of a binary heap Q_j for each $j \leq n$. Each Q_j consists of entries (v_{ki}, i, k) with $k \leq n$ and $i \leq |v_k|$. In Q_j, the entries are ordered lexicographically.

Then I represents a matrix product MV for some square matrix V, where the information about a column V_j is contained in the heap Q_j. Specifically, the heap Q_j represents the linear combination

$$I_j = MV_j = \sum_{(v_{ki},i,k) \in Q_j} M_k \tag{5.1}$$

of columns of M. Since we consider coefficients in $\mathbf{F}_2$, the linear combination (5.1) corresponds to the symmetric difference of the vectors v_k for $(v_{ki}, i, k) \in Q_j$.

At any time, component i of an entry of Q is chosen such that

$$I_j = \sum_{\substack{(v_{ki},i,k) \in Q_j \\ l \geq i}} e_{v_{kl}}.$$

To explain this, assume that Q_j contains two entries (v_{ki}, i, k) and $(v_{k'i'}, i', k')$, such that $v_{ki} = v_{k'i'}$. Then these two entries represent two entries $M_{v_{ki}k}$ and $M_{v_{k'i'}k'}$ of M that lie in the same row $v_{ki} = v_{k'i'}$. When forming forming the symmetric difference (5.1), they add up to zero. Thus, we may remove these entries from Q_j and replace them by $(v_{ki+1}, i+1, k)$ and $(v_{k'i'+1}, i'+1, k')$, given that v_k and $v_{k'}$ have further entries.

We maintain the property that if $(v_{ki}, i, k) \in Q_j$, then all entries $v_{ki'}$ of v_k with $i' < i$ have been canceled with entries of other vectors $v_{k'}$ occurring in Q_j. We also maintain the property that if $Q_j \neq \emptyset$, then there is a unique entry (v_{ki}, i, k) for which v_{ki} becomes maximal. Then this entry represents the pivot of (5.1). We implement the following operations on the columns I_j:

Initialization We initialize I such that I_j represents the column M_j. To do so, we let $Q_j = \{(v_{j1}, 1, j)\}$ for each j; that is, the unique entry in Q_j points to the entry v_{j1} that represents the pivot of M_j.

Pivot retrieval The pivot index $\operatorname{piv} I_j$ of a non-zero column I_j is the unique element $(v_{ji}, i, j) \in Q_j$ for which v_{ji} is maximal.

Pivot consolidation When Q_j has been manipulated, Algorithm 18 reestablishes the property that Q_j has a unique entry (v_{ki}, i, k) for which v_{ki} becomes maximal. The algorithm keeps taking elements (v_{ki}, i, k) from Q_j (in descending order of the row indices v_{ki}) as long as there are two entries with the same row index v_{ki}. If entries (v_{ki}, i, k) and $(v_{k'i'}, i', k')$ satisfy $v_{ki} = v_{k'i'}$, they correspond to two entries $M_{v_{ki},k}$ and $M_{v_{k'i'},k'}$ that add to zero in (5.1). If $i < |v_k|$, then the vector v_k representing M_k has further entries $v_{ki'} < v_{ki}$ for $i' > i$ that have not been processed yet. Therefore, we add an entry $(v_{k,i+1}, i+1, k)$ pointing to the next entry of v_k back to Q_j. Analogously for i'.

Pivot deletion To set the pivot entry of I_j to zero, we delete the maximal entry of Q_j and consolidate the pivot (Algorithm 18).

Column addition A column operation $I_j \leftarrow I_j + M_{j'}$ is implemented by adding $(v_{j'1}, 1, j')$ to Q_j. A column operation $I_j \leftarrow I_j + I_{j'}$ is implemented by setting $Q_j \leftarrow Q_j \cup Q_{j'}$. In either case, it is necessary to call Algorithm 18 afterwards to consolidate the pivot.

If there are many column operations of the form $I_j \leftarrow I_j + I_{j'}$, setting $Q_j \leftarrow Q_j \cup Q_{j'}$ may cause Q_j to grow prohibitively large. As a remedy, we propose the following approach inspired by the *accelerated* matrix representations proposed in [12]. During a column reduction scheme, as long as we perform column operations $M_j \leftarrow M_j + M_k$ to the same column M_j, we implement these operations by working on I_j instead. After the last column operation on M_j has happened, we convert the manipulated matrix column I_j to an explicit vector v'_j of the non-zero entries in I_j using Algorithm 19, and set $v_j \leftarrow v'_j$. In other words, only the currently manipulated column of M is represented indirectly, while all others are stored explicitly.

5.1.4 Coning off K_*

Recall that Algorithm 11 requires that the input chain complex $C_\bullet$ is eventually acyclic. If K_* does not have this property, we replace $C_\bullet(K_*)$ by a suitable filtered complex $C_\bullet$ as described in Section 3.7. Implementing the simplicial cone construction from Section 3.7.1 is straight forward, see the class `Cone` in file `Cone.hpp`. In the following, we describe the implementation of the strategy from Section 3.7.2 that can be found in the class `HBasisCone` in the same file. This strategy first computes a persistence bases of the one-parameter persistent homologies $H_\bullet(\bigcup_{y\in\mathbf{Z}} K_{(*,y)})$ and $H_\bullet(\bigcup_{x\in\mathbf{Z}} K_{(x,*)})$, and then uses these persistence bases to construct the desired chain eventually acyclic chain complex $\hat{C}_\bullet$.

Computing a persistence basis Before we explain how we build $\hat{K}_*$, we describe an efficient algorithm to compute persistence basis [8, 53].

Let $K_* \in \mathrm{Simp}^{\subseteq\mathbf{Z}}$. For each d, let D_d represent ∂_d with respect to the standard basis. Let V_d be upper triangular such that $R_d = D_d V_d$ is reduced. Assume that $\operatorname{barc} H_d(C_\bullet) = \{(b_i, d_i) \mid i \in I\}$ for some index set I. Recall from Example 2.2.7 that

$$\{[R_{d+1}]_j \neq 0\} \cup \{[V_d]_i \mid [R_d]_i = 0 \text{ and } \nexists j : i = \operatorname{piv}[R_{d+1}]_j\}$$

is a persistence basis of $H_d(K_*)$. Recall from Section 2.2.1 that for Vietoris–Rips complexes, computing the matrices R_d and V_d cannot be efficiently combined with clearing, because clearing cannot be applied to the first (viz. highest dimensional) boundary matrix $D_{d_{\max}}$. If the barcode was known, however, clearing can be realized efficiently also with boundary matrices.

The barcode, in turn, can be efficiently computed using cohomology. The computation of a persistence basis of $H_d(K_*)$ thus proceeds in two steps: first, one applies the clearing algorithm (Algorithm 2) to the coboundary matrices $D^d \coloneqq (D_d)^\perp$. From the resulting matrices R^d and V^d one obtains $\operatorname{barc} H^d(K, K_*)$ and thus $\operatorname{barc} H_d(K_*)$. Second, one uses this barcode to apply the clearing algorithm to the boundary matrices D_d. This results in the matrices R_d and V_d, from which one obtains the desired persistence basis of the homology modules $H_d(K_*)$. In the following, we explain this algorithm in detail.

Recall the following terminology. If $[R_d]_i = 0$, we call i a *birth index* of $H_d(K_*)$. We call i a *non-essential* birth index if $i = \operatorname{piv}[R_{d+1}]_j$ for some j. In this case, we call j a *death index* of $H_d(K_*)$ and (i, j) a *persistence pair* of $H_d(K_*)$. If there is no such j, we call i an *essential* birth index of $H_d(K_*)$. Analogously, if $j = \operatorname{piv}[R^{d+1}]_i$, then (i, j) is called a persistence pair, j a non-essential birth index and i a death index $H^{d+1}(K, K_*)$. If $[R^{d+1}]_i = 0$ and i is not a non-essential birth index, then i is an essential birth index of $H^d(K, K_*)$ (sic).

There is a correspondence

$$\begin{aligned} \left\{ \begin{array}{c} \text{persistence pairs } (j,i) \text{ of} \\ H^{d+1}(K,K_*) \end{array} \right\} &\overset{1:1}{\rightleftarrows} \left\{ \begin{array}{c} \text{persistence pairs } (i',j') \text{ of} \\ H_d(K_*) \end{array} \right\} \\ \left\{ \begin{array}{c} \text{essential birth indices } i \text{ of} \\ H^{d}(K,K_*) \end{array} \right\} &\overset{1:1}{\rightleftarrows} \left\{ \begin{array}{c} \text{essential birth indices } i' \text{ of} \\ H_d(K_*) \end{array} \right\} \end{aligned} \tag{5.2}$$

where $i' = n_d + 1 - i$ and $j' = n_{d+1} + 1 - j$. This follows from Corollary 2.2.11. However, this is only a correspondence of matrix row and column indices and thus of the intervals in the respective barcodes. There is no correspondence of representative (co)chains.

Example 5.1.1. Let K_* be the simplicial complex that has the (reduced) (co)boundary matrices

$$D_0 = (), \quad D_1 = \begin{pmatrix} 1&1&0&0&0 \\ 1&0&1&0&1 \\ 0&0&0&1&1 \\ 0&1&1&1&0 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 1\\1\\0\\1\\1 \end{pmatrix}, \quad D^0 = (), \quad D^1 = \begin{pmatrix} 0&1&1&0 \\ 1&1&0&0 \\ 1&0&1&0 \\ 1&0&0&1 \\ 0&0&1&1 \end{pmatrix}, \quad D^2 = (1\;1\;0\;1\;1)$$

with coefficients in $\mathbf{F}_2$ and $n_0 = 1$, $n_1 = 5$ and $n_2 = 1$. Left-to-right reduction yields

$$R_0 = (), \quad R_1 = \begin{pmatrix} 1&1&0&1&0 \\ 1&0&0&0&0 \\ 0&0&0&1&0 \\ 0&1&0&0&0 \end{pmatrix}, \quad R_2 = \begin{pmatrix} 1\\1\\0\\1\\1 \end{pmatrix}, \quad R^0 = (), \quad R^1 = \begin{pmatrix} 0&1&1&0 \\ 1&1&0&0 \\ 1&0&1&0 \\ 1&0&0&0 \\ 0&0&1&0 \end{pmatrix}, \quad R^2 = (1\;0\;0\;0\;0).$$

We obtain the persistence pairs that we list in the following schematic:

$$\begin{array}{cc|c|c|cc}
 & H_\bullet(K_*) & & & H^\bullet(K,K_*) & \\
R_0 & (1,\infty) & & & H^0(K,K_*)\ \{\ (4,\infty) & R^0 \\
 & (2,1) & & & (5,3) & \\
R_1 & (3,4) & H_0(K_*) & \xleftrightarrow{1:1} & H^1(K,K_*)\ \ (2,2) & R^1 \\
 & (4,2) & & & (4,1) & \\
 & (3,\infty) & & & (3,\infty) & \\
R_2 & (5,1) & H_1(K_*) & \xleftrightarrow{1:1} & H^1(K,K_*)\ \{\ (1,1) & R^2
\end{array}$$

Here, the essential birth indices i are written as pairs (i, ∞). The indices of $H_\bullet(K_*)$ on the left can be obtained from the reduced matrices R_0, R_1 on R_2. Persistence pairs of $H_d(K_*)$ are obtained from R_{d+1}. For the essential birth indices of $H_d(K_*)$, one needs both R_d and R_{d+1}. Red, green and blue numbers are column indices of R_0, R_1 and R_2 on the left and correspond to the respective row indices of R^0, R^1 and R^2 on the right.

To compute a persistence basis of $H_d(C_\bullet)$, we need to compute the reduced columns $[R_{d+1}]_j$, where j ranges over all death indices of $H_d(C_\bullet)$, and the reduction matrix columns $[V_d]_i$ with $[R_d]_i = 0$, where i ranges over the essential birth indices of $H_d(C_\bullet)$. This is the essential idea of the following algorithm.

Proposition 5.1.2 (Efficient persistent homology computation). *Let the* $\mathbf{Z}$*-graded matrices* D_1*,* $D_2, \ldots$ *represent a chain complex* $C(K_*)$ *of free modules for some* $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}}$*. Then Algorithm 20 computes a persistence basis of the reduced homology* $H_d(K_*)$ *for each* d*.*

Algorithm 20: One-parameter persistence basis algorithm. Efficient computation of a persistence basis of $H_\bullet(K_*)$

Input: $\mathbf{Z}$-graded matrices $(D_d \in k^{n_{d-1} \times n_d})_{d \geq 0}$ with row and column grades in ascending order that represent a chain complex $C_\bullet$ of free modules.

Output: Persistence bases for each $H_d(C_\bullet)$.

```
    $R_0 \leftarrow () \in k^{0 \times n_0}$; $V_0 \leftarrow E \in k^{n_0 \times n_0}$
    $b \leftarrow \{1, \dots, n_0\}$                                  ▷ birth indices of $H_{d+1}(K_*)$
    $p \leftarrow () \in \mathbf{N}^0$                                ▷ pivot row to column
(a) for $d = 0, 1 \dots$ do                                           assignment of $R_0$
      let $R^{d+1} = D^\perp_{d+1} V^{d+1}$ be reduced for $V^{d+1}$ invertible upper triangular   ▷ use Algorithm 2
      $t \leftarrow \{n_{d+1} + 1 - \text{piv}[R^{d+1}]_i \mid [R^{d+1}]_i \neq 0\}$   ▷ death indices of $H_d(K_*)$
      $b_\text{e} \leftarrow \{i \in b \mid [R^{d+1}]_{n_d+1-i} = 0\}$     ▷ ess. birth indices of $H_d(K_*)$
      $b \leftarrow \{j \leq n_{d+1} \mid j \notin t\}$                 ▷ birth indices of $H_{d+1}(K_*)$
(b)   for $i \in b_\text{e}$ with do                                  ▷ compute representatives $[V_d]_i$
        while $[R_d]_i \neq 0$ do                                      of essential classes in $H_d(K_*)$
          $h \leftarrow \text{piv}[R_d]_i$
          $[R_d]_i \leftarrow [R_d]_i - [R_d]_{hi}/[R_d]_{hp_h}[R_d]_{p_h}$     ▷ $i$ is a birth index, so $p_h \neq 0$
          $[V_d]_i \leftarrow [V_d]_i - [R_d]_{hi}/[R_d]_{hp_h}[V_d]_{p_h}$
      $R_{d+1} \leftarrow D_{d+1}$
      $V_{d+1} \leftarrow E \in k^{n_{d+1} \times n_{d+1}}$
      $p \leftarrow 0 \in \mathbf{N}^{n_d}$
(c)   for $j \in t$ in asc. order do                                  ▷ compute representatives
        while $p_i \neq 0$ for $j = \text{piv}\, L_k$ do               $[R_{d+1}]_j = D[V_{d+1}]_j$ of
          $[R_{d+1}]_j \leftarrow [R_{d+1}]_j - [R_{d+1}]_{ij}/[R_{d+1}]_{ip_i}[R_{d+1}]_{p_i}$   persistence pairs of $H_d(K_*)$
          $[V_{d+1}]_j \leftarrow [V_{d+1}]_j - [R_{d+1}]_{ij}/[R_{d+1}]_{ip_i}[V_{d+1}]_{p_i}$
        $p_i \leftarrow j$
(d)   yield $\{([R_{d+1}]_j, [V_{d+1}]_j) \mid j \in t\} \cup \{[V_d]_i \mid i \in b_\text{e}\}$   ▷ persistence basis of $H_d$
```

Proof. Assume inductively that at the beginning of the loop in line (a), the set b contains the birth indices of $H_d(K_*)$. Assume further that $R_d = D_d V_d$ such that the submatrix $([R_d]_j)_{j \notin b}$ is reduced, and assume that $p_i = j$ for all non-zero columns $[R_d]_j$ with $i = \text{piv}[R_d]_j$. We show that the algorithm correctly computes a persistence basis of $H_d(K_*)$, and establishes the above conditions for the next iteration.

Let $R^{d+1} = D^\perp_{d+1} V^{d+1}$ be reduced for V^{d+1} invertible upper triangular. If $[R^{d+1}]_i \neq 0$ with $j = \text{piv}[R^{d+1}]_i$ then (j, i) is a persistence pair of $H^{d+1}(C^\bullet)$, so $(n_d + 1 - i, n_{d+1} + 1 - j)$ is persistence pair of $H_d(K_*)$. In particular, $n_d + 1 - i$ is a non-essential birth index and $n_{d+1} + 1 - j$ is a death index of $H_d(K_*)$. Therefore,

$$t = \{n_{d+1} + 1 - \text{piv}[R^{d+1}]_i \mid [R^{d+1}]_i \neq 0\}$$

is the set of death indices of $H_d(K_*)$. On the other hand, if $[R^{d+1}]_i = 0$, then i is a birth index of $H^d(C^\bullet)$. Then i is either an essential birth index of $H^d(C^\bullet)$ (equivalently, $n_d + 1 - i$ is a birth index of $H_d(K_*)$), or a non-essential birth index of $H^d(C^\bullet)$ (equivalently, $n_d + 1 - i$ is a death index of $H_{d-1}(K_*)$). Since by assumption, b contains precisely the birth indices of $H_d(K_*)$, the set

$$b_\text{e} = \{n_d + 1 - i \in b \mid [R^{d+1}]_i = 0\}$$

contains precisely the essential birth indices of $H_d(K_*)$. Because an index $j \leq n_{d+1}$ is either a death index of $H_d(K_*)$ or a birth index of $H_{d+1}(K_*)$, setting

$$b = \{j \leq n_{d+1} \mid j \notin t\}$$

establishes that b contains precisely the birth indices of $H^{d+1}(K_*)$. To compute a persistence basis of $H_d(K_*)$, we need to compute the reduced matrix column $[R_{d+1}]_j$ if $j \in t$ and the reduction matrix columns $[V_d]_i$ if $i \in b_{\mathrm{e}}$.

The latter happens in line (b), which is the usual reduction scheme. In order to compute the columns $[R_d]_i$ and $[V_d]_i$, it suffices to know the columns $[R_d]_k$ and $[V_d]_k$, where k ranges over the death indices of $H_{d-1}(K_*)$, simply because a reduced matrix R_d would satisfy $[R_d]_k = 0$ if k not a death index. A column index k of R_d is either a death index of $H_{d-1}(K_*)$ or a birth index of $H_d(K_*)$. By assumption, $([R_d]_j)_{j \notin b}$ is reduced, and p contains the corresponding row-to-pivot assignment. Therefore, the loop in line (b) correctly computes $[V_d]_i$ for all $i \in b_{\mathrm{e}}$. Note that for each $i \in b_{\mathrm{e}}$, the while loop inside line (b) terminates with $[R_d]_i = 0$. Analogously, line (c) computes the reduced matrix columns $[R_{d+1}]_j$ and $[V_{d+1}]$ where $j \in t$ ranges over the death indices of $H_d(K_*)$, and thus establishes the induction hypothesis that $([R_{d+1}]_j)_{j \notin b} = D_{d+1}([V_{d+1}]_j)_{j \notin b}$ be reduced and p contains the corresponding pivot assignment.

Thus, when the main loop reaches line (d), the hypotheses for the next iteration are satisfied, $\{[R_{d+1}]_j, [V_{d+1}]_j \mid j \in t\}$ is a system of representatives of the finite bars in $\operatorname{barc} H_d(K_*)$, and $\{[V_d]_i \mid i \in b_{\mathrm{e}}\}$ is a system of representatives of the infinite bars in $\operatorname{barc} H_d(K_*)$. □

To obtain matrices R^{d+1} and V^{d+1} such that $R^{d+1} = D^{d+1}V^{d+1}$ is reduced, one may use the Standard Algorithm with clearing (Algorithm 2). As remarked earlier, most birth columns of the boundary matrices of a Vietoris–Rips complex are non-essential and account for the largest fraction of the runtime of Algorithm 1. Because Algorithm 20 skips reduction of these columns, it computes a persistence basis much faster than Algorithm 1 in practice.

Remark. The construction generalizes immediately to the situation of computing a persistence basis of $H_\bullet(C_\bullet)$, where $C_\bullet$ is a chain complex of finite rank free $\mathbf{Z}$-persistence modules. In this case, one replaces $H^\bullet(K, K_*)$ by $H^\bullet(\nu C_\bullet)$, where ν is the Nakayama functor from Definition 3.2.2. This makes sense, because if $K_* \in \mathrm{Simp}^{\subseteq \mathbf{Z}}$, then $C^\bullet(K, K_*) = \nu C_\bullet(K_*)$.

Constructing the eventually acyclic complex $\hat{C}_\bullet$ Let $C_\bullet = \bigoplus_{l=1}^N F(z_l)$ be a chain complex of finite rank free $\mathbf{Z}^2$-modules as above, and let $z_0 > z_l$ for all l. We use Algorithm 20 to build a chain complex $\hat{C}_\bullet$ with $(\hat{C}_\bullet)_z = C_z$ if $z < z$ and $H_\bullet(\hat{C}_\bullet)_z = 0$ otherwise.

Let $z_l = (x_l, y_l)$ for all l and $z_0 = (x_0, y_0)$, and consider the chain complex $C'_\bullet = \operatorname{colim}_y C_\bullet = \bigoplus_{i=1}^N F(x_i)$ of free one-parameter persistence modules. For each $d \geq 0$, let $\operatorname{barc} H_d(C_\bullet) = \{(b_{d,i}, d_{q_i}) \mid i \in I_d\}$ for some indexing set I_d. Let $I_d^{\mathrm{f}} := \{i \in I \mid d_{d,i} < \infty\}$ and $I_d^{\mathrm{e}} = I_d \setminus I_d^{\mathrm{f}}$. We use Algorithm 20 to compute a persistence basis

$$\{(z_{d,i} = \partial c_{d,i}, c_{d,i}) \mid i \in I_d^{\mathrm{f}}\} \cup \{z_{d,i} \mid i \in I_d^{\mathrm{e}}\}$$

of $H_d(C_\bullet)$ as in Definition 2.2.6. For each $i \in I_d^{\mathrm{f}}$, define the chain complex

$$\Delta_{d,i}\colon \qquad \cdots \longrightarrow 0 \longrightarrow F(d_{d,i}, y_0) \xrightarrow{1} (b_{d,i}, y_0) \longrightarrow 0 \longrightarrow \cdots$$

concentrated in degrees d, $d+1$. There is morphism of complexes

$$\begin{tikzcd}
\Delta_{d,i}\colon & \cdots \arrow[r] & 0 \arrow[r] & F(d_{d,i}, y_0) \arrow[r, "1"] \arrow[d, "c_{d,i}"'] & F(b_{d,i}, y_0) \arrow[r] \arrow[d, "z_{d,i}"] & 0 \arrow[r] & \cdots \\
C_\bullet\colon & \cdots \arrow[r] & C_{d+2} \arrow[r] & C_{d+1} \arrow[r] & C_d \arrow[r] & C_{d-1} \arrow[r] & \cdots .
\end{tikzcd}$$

Analogously, if $i \in I_d^{\mathrm{e}}$, define the chain complex

$$\Delta_{d,i}\colon \qquad \cdots \longrightarrow 0 \longrightarrow 0 \longrightarrow (b_{d,i}, y_0) \longrightarrow 0 \longrightarrow \cdots$$

and the morphism

$$\begin{tikzcd}
\Delta_{d,i}\colon & \cdots \arrow[r] & 0 \arrow[r] & 0 \arrow[r] \arrow[d] & F(b_{d,i}, y_0) \arrow[r] \arrow[d, "z_{d,i}"] & 0 \arrow[r] & \cdots \\
C_\bullet\colon & \cdots \arrow[r] & C_{d+2} \arrow[r] & C_{d+1} \arrow[r] & C_d \arrow[r] & C_{d-1} \arrow[r] & \cdots .
\end{tikzcd}$$

Then we get a morphism

$$e\colon \bigoplus_{d\geq 0, i\in I_d} \Delta_{d,i} \to C_\bullet .$$

and we let $\hat{C}^x_\bullet \coloneqq \operatorname{cone} e$. To make this explicit, define for each $d \geq 0$ the free module

$$G_d \coloneqq \langle \hat{z}_{d-1,i} \mid i \in I_{d-1}\rangle \oplus \langle \hat{c}_{d-2,i} \mid i \in I^{\mathrm{f}}_{d-2}\rangle$$

spanned by symbols $\hat{z}_{d,i}$ and $\hat{c}_{d-1,i}$ of grades $g(\hat{z}_{d,i}) = (b_{d,i}, y_0)$ and $g(\hat{c}_{d-1,i}) = (d_{d-1,i}, y_0)$. Then

$$\hat{C}_\bullet = \left(\begin{array}{c}
\cdots \to C_{d+2} \oplus G_{d+2} \longrightarrow C_{d+1} \oplus G_{d+1} \longrightarrow C_d \oplus G_d \to \cdots \\
\hat{c}_{d-2,i} \longmapsto c_{d-2,i} - \hat{z}_{d-2}, \\
\hat{z}_{d-2,i} \longmapsto z_{d-2,i},
\end{array}\right).$$

Repeating this process with the roles of x and y exchanged yields a chain complex $\hat{C}_\bullet$ with $\hat{C}_z = C_z$ if $z < z_0$ and $H_\bullet(\hat{C}_\bullet)_z = 0$ otherwise.

5.1.5 Sparsification

We observe that in Algorithm 10, the second for-loop takes orders of magnitude longer than the first loop. Furthermore, it has the tendency to increase the sparsity of the matrix considerably, which may incur high costs on all further operations on these matrices. In this section, we give a heuristic explanation, and devise a partial remedy.

Explanation Let K_* be a $\mathbf{Z}^2$-filtered simplicial complex and D^{d+1} represent its coboundary matrices with respect to the standard basis. We let the simplices of K_* be enumerated colexicographically, such that

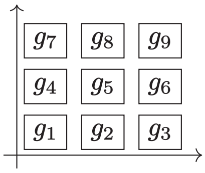
$$K_* = \{\sigma_1, \ldots, \sigma_n\} \quad \text{with} \quad g(\sigma_1) \preceq_{\text{colex}} \cdots \preceq_{\text{colex}} g(\sigma_n).$$

We obtain a $\mathbf{Z}$-filtered complex K'_* with $K'_z = \{\sigma_i \in K \mid i \leq z\}$. It has the same coboundary matrices as K_*, including row and column order. The first for-loop in Algorithm 10 can be seen as a usual one-parameter persistence left-to-right reduction scheme, preceded by re-ordering the columns of D^{d+1} by their colex grades. Recall that Algorithm 10 is invoked after performing a clearing step; see Algorithm 11.

Recall that in one-parameter persistence, if applied to a coboundary matrix D, the standard algorithm spends most of its runtime on the columns it reduces to zero (i.e., birth columns). Consequently, clearing remedies this by skipping the reduction of most of the birth columns. However, this is only true if the matrix being reduced comes from a filtration, as this imposes a certain block upper triangular shape on the matrix. Therefore, apart from the birth columns, the matrix is “almost” reduced already at the beginning.

This also holds for coboundary matrices arising from two-parameter filtrations. To illustrate this, consider the following example. Let the different grades $g_1 \preceq_{\text{colex}} \cdots \preceq_{\text{colex}} g_9$ that occur as row or column grades of a valid $\mathbf{Z}^2$-graded matrix D be arranged as in the following schematic.

Then validity of D imposes the following block structure on D, where empty blocks are zero:



$$D = \begin{array}{c} \\ g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} g_1 & g_2 & g_3 & g_4 & g_5 & \cdots \end{array} \\ \begin{pmatrix} * & * & * & * & * & \\ & * & * & & * & \\ & & * & & & \\ & & & * & * & \\ & & & & * & \\ & & & & & \ddots \end{pmatrix} \end{array}$$

Of course, block rows or columns may be empty, in case D has no rows or columns of that grade. Nevertheless, validity of D imposes that D has the above block upper triangle shape. Ordering the rows of D by their lex-pivots does not change this. To see this, note that in the above example, we obtain

$$\begin{array}{c|ccccccccc} & g_1 & g_4 & g_7 & g_2 & g_5 & g_8 & g_3 & g_6 & g_9 \\ \hline g_1 & * & * & * & * & * & * & * & * & * \\ g_2 & & & & * & * & * & * & * & * \\ g_3 & & & & & & & & * & * \\ g_4 & & * & * & & * & * & & * & * \\ g_5 & & & & & * & * & & * & * \\ g_6 & & & & & & & & * & * \\ g_7 & & & * & & & * & & & * \\ g_8 & & & & & & * & & & * \\ g_9 & & & & & & & & & * \end{array},$$

where all empty entries are zero. Although the matrix is not in block upper triangular shape anymore, we see that to the left of the column column g_j, which has block pivot g_j, no other column has an entry in that block row. This ensures that despite the re-ordering, reduction is fast.

For the second for-loop, which can be seen as a left-to-right reduction scheme, preceded by reordering the rows lexicographically by grade and the columns by colex-pivot, the performance drastically changes. We assume that this reordering renders D the coboundary matrix of a randomly ordered simplicial complex. It is known reducing the (co)boundary matrix of a simplicial complex with simplices in random order takes much longer in practice than reducing the (co)boundary matrix of a $\mathbf{Z}$-filtered complex [17, 78].

To make a concrete example, consider the following numbers. We have run our algorithm on the coboundary matrices of the full density-Rips complex of 150 points sampled from the orthogonal group $O(5)$, embedded in $\mathbf{R}^{25}$. To compute $H^3(N^\bullet(K_*))$, we have to apply Algorithm 10 to the coboundary matrix D^{d+2} of $N^\bullet$, which is a graded $551,300 \times 11,175$. After the first for loop of Algorithm 10, which takes 5 ms, the densest columns have up to 1,456 entries in the heaps representing the columns. After the second, which takes 10,350 ms, the densest columns have up to 78,875,970 entries in the heaps.[1] This is a somewhat extreme example, but illustrates the problem. This implies that all further matrix computations with this matrix (namely, minimization and cokernel computation) will be expensive.

Sparsification We provide a partial remedy for this. Recall that our goal is to compute the submodule $[V]_F$ of a free module $F = \bigoplus_{i=1}^n F(z_i)$, where V is a subspace of $\operatorname{colim} F$. The space V is given by a matrix B representing a generating system of V with respect to the standard basis of $\operatorname{colim} F$. Then Algorithm 10 computes a graded matrix $B' = \texttt{Bireduce}(B)$ that represents a basis of the free module $[V]_F$ with respect to the standard basis of F.

However, in Algorithm 11, which is where Algorithm 10 is invoked, we do not require that B' represents a basis of $[V]_M$ with respect to the standard basis of F. Instead, it suffices if B' represents a basis of $[V]_F$ with respect to *any* basis of F; see also Corollary 3.5.1. We may, therefore, choose the basis of F that suits our task best; i.e., for which B' is particularly sparse.

Definition 5.1.3. A non-zero entry of a graded matrix M is *dominant* if it is the only non-zero entry in its row that is not dominated. A non-zero entry M_{ij} is *dominated* if there is a dominant entry M_{kj} in the same column, such that $\mathrm{rg}_i^M \leq \mathrm{rg}_k^M$.

[1] Note that this shows a particular shortcoming of the heap representation: namely, using 78,875,970 to represent a matrix column with $551,300$ entries is certainly not sparse. With the vector representation, we obtain up to 1,644 non-zero entries in one column after the first for-loop (3 ms) and up to 141,026 non-zero entries after the second (971 ms). Compare also the results from [17].

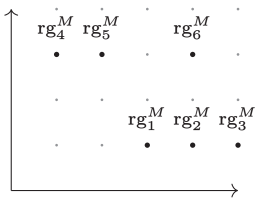
Remark (Base changes of $\mathbf{Z}^n$-graded modules). Let $M\colon F \to E$ be morphism of finite rank free modules. Let M be the valid graded matrix representing M with respect to the homogeneous basis $f_1, \dots, f_m$ of F and $e_1, \dots, e_n$ of E.

- If $g(f_i) \leq g(f_j)$, then replacing f_j by $f'_j = f_j + \lambda F_{g(f_j),g(f_i)}(f_i)$ defines another basis of F. The matrix M' representing M with respect to this basis is obtained by replacing the column M_j by $M'_j := M_j + \lambda M_{*i}$.
- If $g(e_i) \leq g(e_j)$, then replacing e_j by $e'_j = e_j + \lambda F_{g(e_j),g(e_i)}(e_i)$ defines another basis of E. The matrix M' representing M with respect to this basis is obtained by replacing the row M_{i*} by $M'_{i*} = M_{i*} - \lambda M_{j*}$.

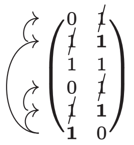
If M_{ij} is dominated by M_{kj}, then M_{ij} can be erased by the row operation $M_{i*} \leftarrow M_{i*} - \frac{M_{ij}}{M_{kj}} M_{k*}$, and this row operation only changes the entry M_{ij}. Such eliminations of dominated entries using dominating ones are therefore compatible with column sparse matrix formats.

Example. Consider the graded matrix

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{with row grades}$$



The entry $M_{6,1}$ is dominant. The row operation $M_{5*} \leftarrow M_{5,*} - M_{6,*}$ is valid because $b \leq a$, and results in setting the entry $M_{5,1}$ to zero. After the operation, also $M_{5,2}$ becomes dominant, and we perform row operation $M_{4,*} \leftarrow M_{4,*} - M_{5,*}$, which sets the entry $M_{4,2}$ to zero. No entry of $M_{3,*}$ can be removed, so the two non-zero entries in that row are not dominant. In particular, they cannot be used to set the row $M_{2,*}$ to zero without a proper row operation. Instead, the row operation $M_{2,*} \leftarrow M_{2,*} - M_{5,*}$ sets $M_{2,5}$ to zero and leaves the dominant entry $M_{2,1}$. This can be used for the last operation $M_{1,*} \leftarrow M_{1,*} - M_{2,*}$. We obtain the following sequence of row operations from bottom to top:


$\begin{pmatrix} 0 & \cancel{1} \\ \cancel{1} & \mathbf{1} \\ 1 & 1 \\ 0 & \cancel{1} \\ \cancel{1} & \mathbf{1} \\ \mathbf{1} & 0 \end{pmatrix}$

The struck through entries are set to zero in the course of the operation. Of the remaining entries, the ones printed in boldface are dominant.

We obtain the following algorithm to eliminate dominated matrix entries. We first describe the algorithm on a high level (Algorithm 21) before we describe a more implementation-oriented formulation (Algorithm 22).

Proposition 5.1.4. *Let M be a valid graded matrix such that $\mathrm{rg}_i^M \ngeq \mathrm{rg}_j^M$ if $i \leq j$. Then Algorithm 21 computes a graded matrix M' with the same row and column grades as M, such that there is an invertible valid graded matrix V with $M' = MV$ by eliminating all possible dominated entries of M.*

Proof. During the main for-loop, the algorithm maintains the property that d_j contains the indices i such that $M_{ij} \neq 0$ is dominant. During each iteration, it maintains the property that u contains the column indices j of the entries in row M_{i*} that are not dominated. Assume that $M_{ij} \neq 0$ and that there exists a $h \in d_j$ such that $\mathrm{rg}_i^M \leq \mathrm{rg}_h^M$. Then the row operation $M_{i*} \leftarrow M_{i*} - \frac{M_{ij}}{M_{hj}} M_{h*}$ is a valid base change and sets the entry M_{ij} to zero. Since M_{hj} is a

Algorithm 21: Elimination of dominated entries

Input: A graded $m \times n$-matrix M such that $\mathrm{rg}^M_i \ngeq \mathrm{rg}^M_j$ if $i \leq j$.

Output: A graded matrix $M' = VM$ for some valid invertible upper triangular matrix V, such that M' contains not more non-zero entries than M

```
$d_j \leftarrow \emptyset$ for all $j$                                  ▷ row indices of dominant
for $i = m, \dots, 1$ do                                                  entries in column $j$
    $u \leftarrow \emptyset$
    for $j = 1, \dots, n$ with $M_{ij} \neq 0$ do
        if $\exists h \in d_j : \mathrm{rg}^M_i \leq \mathrm{rg}^M_h$ then               ▷ $\exists$ a dominant entry $M_{hj}$
            $M_{ij} \leftarrow 0$
        else
            $M'_{ij} \leftarrow M_{ij}$
            $u \leftarrow u \cup \{j\}$
    if $u = \{j\}$ for some $j$ then                                   ▷ $M_{ij}$ is dominant
        $d_j \leftarrow d_j \cup \{i\}$
```

dominant entry, no other row entry of M_{i*} is affected by this row operation. Therefore, the row operation can be realized by just setting $M_{ij} \leftarrow 0$. If there is no such h, then M_{ij} is not dominated. If $u = \{j\}$ and the end of the ith iteration, then M_{ij} is the only non-zero entry in M_{i*} that is not dominated, it is dominant by definition. □

To describe a possible implementation of this algorithm, we assume that we are using the vector representation of matrices; see Section 5.1.3.1. In case we are using heap representations, one can use consolidation (Algorithm 16) to convert a heap-representation to a vector-representation of the same matrix. For simplicity, we describe the following refinement of Algorithm 21 only for matrices with coefficients in $\mathbf{F}_2$; the approach generalizes to arbitrary coefficient fields. Let the rows of M be in colexicographic[2] order with respect to their grades, and let $(v_1, \dots, v_n)$ be the vectors representing M, where each v_j contains the row indices of the non-zero entries in M_j in descending order.

Proposition 5.1.5. *In the above situation, Algorithm 22 computes the same valid graded matrix M' as Algorithm 21.*

Proof. We show that Algorithm 22 performs the same row operations as Algorithm 21. The central data structure is the priority queue Q. Its entries are triples (λ, k, j) with $j \leq n$ and $k \leq |v_j|$ and $\lambda = v_{j,k}$ is the row grade of the last but kth non-zero entry in M_j. Thus, an entry in Q represents the non-zero entry $M_{v_{kj}j}$. The priority queue Q is ordered lexicographically.

Claim 2. Repeatedly calling `PopPush`(Q) retrieves all indices (i, j) of non-zero entries of M, in order of descending row indices i.

Proof of claim. The function `PopPush`(Q) retrieves the maximal (i, k, j) from Q. Because Q is ordered by the row indices $i = v_{jk}$ first, popping the maximal elements from Q thus retrieves the elements in reverse order of the row grades. To see that the function retrieves all indices of non-zero entries of M, we argue as follows. Recall that v_j is descendingly sorted. Thus, if $k = |v_j|$, then i is the smallest row index of a non-zero entry in M_j. Therefore, it is not necessary to process any further entries of v_k. Otherwise, v_j contains more entries i' corresponding non-zero entries $M_{i'j}$ with $i' < i$. Pushing $((v_j)_{k+1}, k+1, j)$ to Q ensures that the non-zero entry $M_{i',j}$ with the largest row index $i' = (v_j)_{k+1}$ less than i is processed as the next entry from M_j. Therefore, `PopPush`(Q) eventually yields all non-zero entries of M.

Claim 3. If an entry M_{ij} is dominated, then it is dominated by the dominant entry M_{lj} with the least row index $l > i$.

[2] or lexicographic; adapt algorithm and proof accordingly.

Algorithm 22: Possible implementation of Algorithm 21.

Input: Descendingly ordered vectors $v_1, \dots, v_n$ representing a reduced valid graded $m \times n$-matrix M with entries in $\mathbf{F}_2$, with rg^M ordered colexicographically.
Output: Descendingly ordered vectors $v'_1, \dots, v'_n$ representing $M' = VM$ as in Algorithm 21.

```
    for $j = 1, \dots, n$ do $v_j \leftarrow \emptyset$                      ▷ vectors representing $M'_j$
    for $j = 1, \dots, n$ do $d_j \leftarrow -1$                             ▷ row index of last dominating
    $Q \leftarrow \{((v_j)_0, 0, j) \mid M_{ij} \neq 0\}$ as lex. ordered priority queue     entry in $M_j$
(a) function PopPush():                                                      ▷ retrieves next entry from $Q$
        $i, k, j \leftarrow$ PopMax($Q$)
        if $k < |v_j|$ then Push($Q$, $((v_j)_{k+1}, k+1, j)$)
        return $i, j$
    $i, k, j \leftarrow$ PopPush($Q$)                                        ▷ largest pivot of $M$
    append $i$ to $v'_j$
    $r \leftarrow 1$                                                         ▷ number of entries not
    while $Q \neq$ do                                                          dominated in $i$th row.
(b)     $i', j' \leftarrow$ PopPush($Q$)
        if $i \neq i'$ then
            if $r = 1$ then $d_j \leftarrow i$                               ▷ $M_{ij}$ is dominating
            $i \leftarrow i'$
            $j \leftarrow j'$
            $r \leftarrow 0$
(c)     if $d_j = -1$ or $\mathrm{rg}^M_{i'} \not\leq \mathrm{rg}^M_{d_{j'}}$ then     ▷ $M_{i'j'}$ is not dominated
            append $i'$ to $v_{j'}$
            $i \leftarrow i'$
            $j \leftarrow j'$
            $r \leftarrow r + 1$
```

Proof of claim. Assume that M_{ij} is dominated by some non-zero entry M_{kj}. This implies that $i < k$ and $\mathrm{rg}^M_i \leq \mathrm{rg}^M_k$. Recall that $z \leq z'$ in $\mathbf{Z}^2$ if and only if $z \preceq_{\mathrm{lex}} z'$ and $z \preceq_{\mathrm{colex}} z'$. In particular, $\mathrm{rg}^M_i \preceq_{\mathrm{colex}} \mathrm{rg}^M_k$ and $\mathrm{rg}^M_i \preceq_{\mathrm{lex}} \mathrm{rg}^M_k$. Let l be the least row index $l > i$ of a dominant entry in column M_j. Then $\mathrm{rg}^M_i \preceq_{\mathrm{colex}} \mathrm{rg}^M_l \preceq_{\mathrm{colex}} \mathrm{rg}^M_k$ since by assumption, the rows of M are ordered colexicographically by grade. Two distinct dominant entries necessarily have incomparable row grades, we get $\mathrm{rg}^M_i \preceq_{\mathrm{lex}} \mathrm{rg}^M_k \preceq_{\mathrm{lex}} \mathrm{rg}^M_l$. Together, this shows that $\mathrm{rg}^M_i \leq \mathrm{rg}^M_l$.

The variables i and j are initialized such that M_{ij} is the pivot of M with the largest row index. Assume by induction that at the beginning of an iteration of the main while-loop, i and j are the row and column indices of the last processed non-zero entry M_{ij}, that r is the number of non-zero entries in row M_{i*} not eliminated, and that d_j contains the row index of the last processed dominating entry in column M_j (or -1, if there is none). Let i' and j' be the next entry from Q as in line (b).

If $i \neq i'$, then all entries in row M_{i*} have been processed. If $r = 1$, there has been a single non-zero entry M_{ij} in row i. It is therefore dominant, which justifies setting $d_j \leftarrow i$. Since there have not been any previously processed entries in row i', the algorithm sets $r \leftarrow 0$, which reestablishes the induction hypothesis.

We have to check if the new entry $M_{i'j'}$ is dominated. By Claim 3, this is the case if and only if it is dominated by the last dominating entry $M_{d_j j'}$; i.e., if $\mathrm{rg}^M_i \leq \mathrm{rg}^M_{d_j}$, given that there exists any previous dominating entry in $M_{j'}$. In this case, the algorithm simply discards the entry, which corresponds to setting ot to zero. Otherwise (line (c)), the entry is kept unchanged, which corresponds to appending its row index to $v'_{j'}$. Setting $i \leftarrow i'$, $j \leftarrow j'$ and $r \leftarrow r + 1$ reestablishes the induction hypothesis. Note that the entries of v'_j are constructed in descending order, as required by the vector representation. □

5.1.6 Chunk(*) preprocessing

As mentioned before, *chunk preprocessing* is an efficient preprocessing scheme that reduces the size of the input chain complex without changing its homology isomorphism type. More concretely, given chain complex $C_\bullet$ of finite rank free two-parameter persistence modules concentrated in degrees ≥ 0, chunk preprocessing computes a minimal chain complex $C'_\bullet$ homotopy equivalent to $C_\bullet$. Assuming that $C_\bullet$ is given as a sequence $D_\bullet$ of boundary matrices, chunk preprocessing applies minimization (Algorithm 3) to all matrices $D_\bullet$; see Algorithm 4.

For chain complexes $C_\bullet$ arising from filtered simplicial complexes in practice, $C'_\bullet$ typically has much smaller rank than $C_\bullet$. It has been observed that when computing a minimal free resolution of $H_d(C_\bullet)$ using the LW-algorithm, applying chunk preprocessing drastically improves the efficiency of the entire pipeline [73, table 3]. A particular reason for this is that, while there is no efficient way known to parallelize the LW-algorithm (Algorithm 6), Algorithm 3 can be implemented in an embarrassingly parallel way; see Section 2.4.2.

For full function Rips-complexes (such as density-Rips), it has been observed that the boundary matrices $D'_\bullet$ of the the chain complex $C'_\bullet$ computed by chunk preprocessing computes, although having fewer rows and columns than $D_\bullet$, have more non-zero entries; see [73, table 2] Nevertheless, applying chunk preprocessing before running the LW-algorithm reduces the runtime.

Chunk* preprocessing Algorithm 3, which underlies chunk preprocessing, minimizes each boundary matrix D_d through left-to-right column additions. Alternatively, we propose to minimize D_d through row operations. Equivalently, this corresponds to applying Algorithm 3 to the transpose D^d of D_d. We thus propose the following preprocessing scheme, which we refer to as chunk* preprocessing: we apply Algorithm 3 to all coboundary matrices D^d, starting with D^1. To compute a minimal free resolution of homology, we transpose the result; see Algorithm 5. We expect that for boundary matrices with much more rows than columns, such as the boundary matrices of Vietoris–Rips complexes, the performance may differ.

5.2 Experiments

In this section, we evaluate the performance of the cohomology algorithm Algorithm 11 using our implementation `2pac` [94]. To do so, we compare the performance of our implementation of the cohomology algorithm (Algorithm 11) and the homology algorithm (Algorithm 9). Although the latter is implemented in the state-of-the-art software `mpfree` [85], we use our own implementation of it. We do this in order to be able to explore the influence of implementation details, such as the matrix representation or the specific preprocessing. This is justified by Table 1, which shows that `mpfree` and the version of `2pac` that uses the vector representation have similar runtimes.

5.2.1 Samples

Although our software can be used to compute minimal free resolutions of the (co)homology of any chain complex of finite rank free two-parameter persistence modules, our main motivation for computing cohomology are two-parameter Rips complexes. This is because in one-parameter persistence, cohomology (with clearing) has increased the practically tractable size of Vietoris–Rips complexes drastically. Therefore, we focus on function-Rips complexes in the experimental evaluation. We have run our experiments on (subsamples of) the following datasets, which have been generated using the Jupyter notebook `EXAMPLE.ipynb` in the repository. The data sets, which consists of the filtration values for the vertices and the distance matrix, are

Table 5.1: Data sets used in the experiments. The first four samples (spheres) consist of 200 points sampled from the actual space, and 100 points sampled from the space described in column "outliers". For the orthogonal groups and tori, we sampled 400 points from the space and 200 outliers. For details about the last one, see the text.

space	outliers	density function	bandwidth
$S^1 \subseteq \mathbf{R}^2$	$[-2, 2]^{\times 2}$	Gaussian	0.25
$S^2 \subseteq \mathbf{R}^3$	$[-1.5, 1.5]^{\times 3}$	Gaussian	0.30
$S^3 \subseteq \mathbf{R}^4$	$[-1.25, 1.25]^{\times 4}$	Gaussian	0.35
$S^4 \subseteq \mathbf{R}^5$	$[-1, 1]^{\times 5}$	Gaussian	0.40
$O(3) \subseteq \mathbf{R}^9$	$[-1.25, 1.25]^{\times 9}$	Gaussian	0.60
$O(4) \subseteq \mathbf{R}^{16}$	$[-1.25, 1.25]^{\times 16}$	Gaussian	0.85
$O(5) \subseteq \mathbf{R}^{25}$	$[-1.25, 1.25]^{\times 25}$	Gaussian	0.85
$T^2 = (S^1)^{\times 2} \subseteq \mathbf{R}^4$	$[-1.5, 1.5]^{\times 4}$	Gaussian	0.35
$T^3 = (S^1)^{\times 3} \subseteq \mathbf{R}^6$	$[-1.5, 1.5]^{\times 6}$	Gaussian	0.50
$T^4 = (S^1)^{\times 4} \subseteq \mathbf{R}^8$	$[-1.25, 1.25]^{\times 8}$	Gaussian	0.70
$C_8H_{16} \subseteq \mathbf{R}^8$	–	–	

generated prior to to the experiments, written to disk, and then read by `2pac` during the experiments to generate the function-Rips complexes. The precise code to reproduce the data files can be found in `Generate Samples.ipynb`.

Density-Rips complexes We have sampled points from spheres $S^n \subseteq \mathbf{R}^{n+1}$, tori $T^n := (S^1)^n \subseteq \mathbf{R}^{2n}$ and orthogonal groups; see Table 5.1. Arguably, the most common density functions $\rho\colon S \to \mathbf{R}$ used in persistent homology are

- the *Gaussian density function*

$$p \mapsto \sum_{q \in S \setminus \{p\}} \exp(\frac{-d(p,q)^2}{2\sigma^2}),$$

- the *ball density function*

$$p \mapsto |\{q \in S \setminus p \mid d(p,q) \leq \sigma\}|,$$

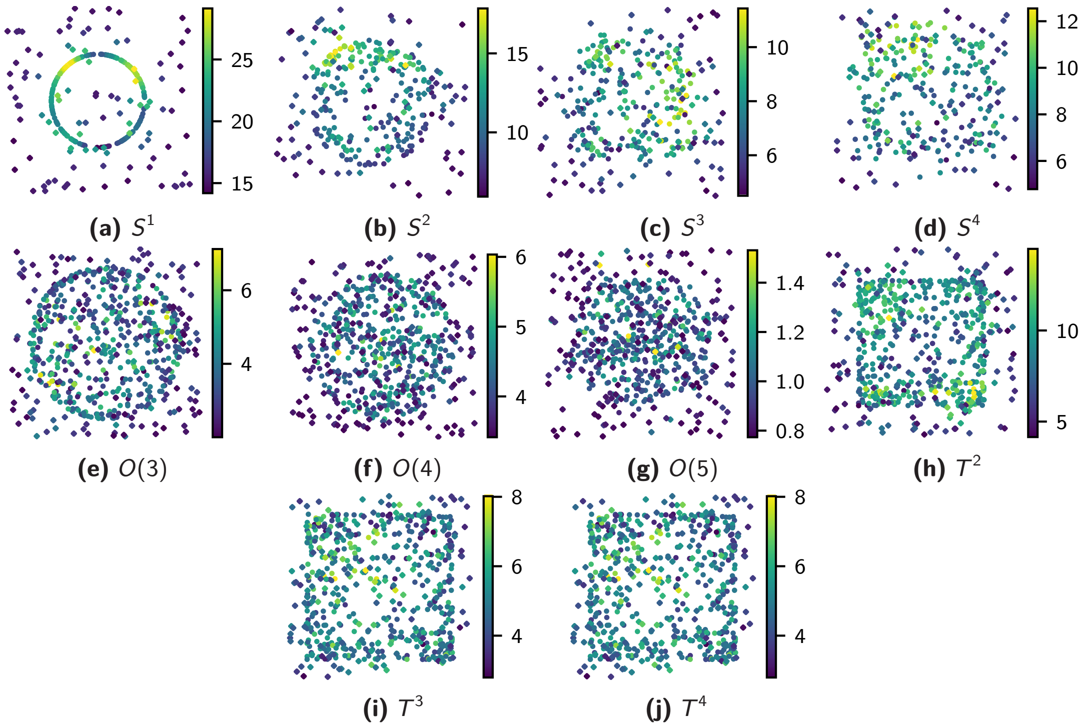
- and the *kth nearest neighbor density function* $p \mapsto d(p,q)$, where $q \in S$ satisfies

$$|\{r \in S \setminus \{p\} \mid d(p,r) < d(p,q)\}| < k.$$

Each of these relies on the choice of a bandwidth parameter $\sigma \in \mathbf{R}$ or $k \in \mathbf{N}$. We observed that usually, the Gaussian density function achieves the best distinction between sample and outliers. Using the notebook `EXAMPLE.ipynb`, we manually chose the bandwidth parameter σ for each of the data sets in Table 5.1. A projection onto the first two coordinates of $\mathbf{R}^n$ of each of the function-Rips data sets is shown in Figure 5.1.

Cyclooctane (C_8H_{16}) The cyclooctane data set, which can be obtained from [101], is sampled from the configuration space of the cyclooctane molecule. More accurately, the sample is taken from the configuration space of a linkage of eight rigid rods of equal length, with fixed equal angles of approximately 118° at the eight joints. Because of this angle, the linkage forms no planar octagon but has to "pucker" in space. In particular, three consecutive edges need not lie in the same plane.

Each of the approx. 6000 entries in the data set consists of the 24 coordinates of the eight vertices in $\mathbf{R}^3$. The data set has previously been studied in persistence [101, 102, 103]. It has been suggested that the described configuration space forms a two-dimensional subspace of $\mathbf{R}^{24}$;



(a) S^1 **(b)** S^2 **(c)** S^3 **(d)** S^4

(e) $O(3)$ **(f)** $O(4)$ **(g)** $O(5)$ **(h)** T^2

(i) T^3 **(j)** T^4

Figure 5.1: The plots show projections of the datasets on the the first two coordinates of the ambient Euclidean space. The colors encode the values of the density function. All datasets use a Gaussian density function, with the bandwidth parameters from Table 5.1.

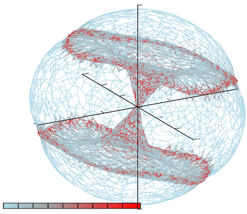


Figure 5.2: Isomap projection [121] of the cyclooctane data set from $\mathbf{R}^24$ to $\mathbf{R}^3$ with respect to the Euclidean distance. The color encodes the metric distortion of the projection (red=high distortion). The component appearing as an hourglass is a projection of the Klein Bottle.

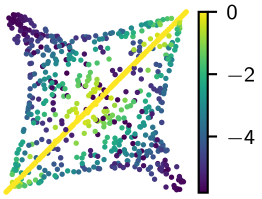


Figure 5.3: Cyclooctane data set. The plot shows the first two torsion angles of the data points. Points in the diagonal of the plot correspond to points from the spherical component of the configuration space. Points from the actual space are printed with circles, outliers with diamonds.

namely, a union of a 2-sphere and a Klein bottle that intersect in two disjoint 1-spheres. For an illustration, see the projection to $\mathbf{R}^3$ in Figure 5.2, which has been obtained with the Isomap algorithm [121].

We equip the sample with the following distance. To each consecutive four vertices v_1, v_2, v_3 and v_4, we assign the signed angle enclosed by the affine plane spanned by v_1, v_2 and v_3 and the affine plane spanned by v_2, v_3 and v_4. This angle is also called the *torsion angle* of the edge v_2v_3. This gives a vector of eight oriented angles. As the distance between two configurations, we use the Euclidean distance between their torsion angle vectors.

We equip the vertices with the function $(\alpha_1, \ldots, \alpha_8) \mapsto |(\alpha_1 - \alpha_5, \ldots, \alpha_4 - \alpha_8)|$. The reason for this choice is that as mentioned above, the configuration space of C_8H_{16} has been described as the union of a sphere and a Klein Bottle, intersecting in two circles [102]. We have observed [95] that configurations from the spherical component have the property $\alpha_i = \alpha_{i+4}$ for all $i \leq 4$. This is not the case in general for configurations from the Klein bottle component. Therefore, the above function measures the deviation from this symmetry.

5.2.2 Parameters

Our main interest lies in comparing the following three approaches:

- the *cohomology algorithm* (Algorithm 11) called by

  `./2pac -c -cone=2 -f[dataset]`,

- the *homology algorithm with chunk preprocessing* (Algorithms 4 and 9), called by

  `./2pac -n`n `-hDC`$(n+1)$ `-f[dataset]`),

- the *homology algorithm with chunk* preprocessing* (Algorithms 5 and 9), called

  `./2pac -hC`$(n+1)$ `-f[dataset]`.

Here n denotes the dimension up to which one wishes to compute a free resolution of H_n. These program calls generate the (co)boundary matrices of the full function-Rips complex K_* of the respective data set, and compute a minimal free resolution of $H_d(K_*)$ (resp., $H^{d+2}(N^\bullet(K_*))$) for $d = 0, \ldots, n$. In the case of homology, passing `-DC`$(n+1)$ (resp. `-C`$(n+1)$) applies chunk (resp. chunk*) preprocessing to all boundary matrices $D_1, \ldots, D_{n+1}$.

Remark (`mpfree`). The second approach listed above is the same that is implemented in `mpfree`. As mentioned at the beginning of this section, use our own implementation to be able to vary implementation details such as the underlying matrix type; see Table 1 for a runtime comparison. A conceptual difference is that `mpfree` is designed for computing a minimal free resolution of homology for a single degree, while `2pac` is designed to produce a sequence of of resolutions of homology in ascending dimensions. This has the effect that `2pac` reuses some data used in the resolution computation in lower homology dimensions (e.g., chunk preprocessing).

We found that this is the best choice for chunk preprocessing: applying minimization only to the first n or less boundary matrices (`-C`n) does not reduce the runtime of the following homology computation considerably, while minimizing the $(n+2)$nd boundary matrix, despite reducing the runtime of the homology algorithm drastically, is too expensive.

For the cohomology algorithm, one has to replace the input complex by an eventually acyclic one, because the function-Rips filtration is not eventually acyclic with regard to the function-parameter. To do this, we pass `-cone=2` when computing cohomology. This builds the eventually acyclic chain complex $\hat{C}_\bullet$ as described in Section 5.1.4. Alternatively, one can pass `-cone=1` to construct $\hat{C}_\bullet$ as the cone of $C_\bullet(K_*)$; see Section 3.7.1. If not stated otherwise, we use `-cone=2`, as we have observed this usually yields better results than `-cone=1`; see Section 5.3.2 below.

Remark. If the input complex is not eventually acyclic also for the second filtration parameter, one has to also pass `-Cone=1` or `-Cone=2`. This is the case, for example, for truncated function-Rips complexes.

In our experiments, we investigate the effect of

- the size of a random subsample taken from the input. The subsample size to take can be specified by passing `-s`; this chooses a random subsample with a fixed seed to ensure reproducibility;
- varying the dimension up to which we compute (co)homology (using `-n`);
- the matrix representation (matrix or heap representation; see Section 5.1.3); to change this, run either `./2pac` (for heaps) or `./2pac.vectors`;
- the strategy to cone off K_*, either using the simplicial cone (`-cone=1`), or using a homology basis (`-cone=2`); and
- combining the cohomology algorithm with chunk(*) preprocessing (using `-C` for chunk* and `-DC` for chunk).

Machine and setup The code version used for the experiments is revision `1853b6f` in the git repository [94]. The repository also contains the shell script (`run-experiments.sh`) to run the experiments, together with the Jupyter notebooks `Generate Samples.ipynb` to generate the samples used in these experiments (see above) and the notebook `Plots and tables.ipynb` used to evaluate these.

The experiments were run on a Ubuntu 22.04.2 LTS machine with 48 3.0 GHz Intel(R) Xeon(R) CPU E5-2687W v4 CPUs and 504 GiB RAM, with the software compiled using g++ 11.3.0. Some experiments are also run on a MacBook Pro 2017 with a 2.3 GHz Dual-Core Intel Core i5 CPU and 16GiB RAM, with the software compiled using clang++ 15.0.7. We refer to this as machine "M" in the captions.

Each instance was killed when exceeding five minutes runtime. If not stated otherwise, the software uses the heap representation of matrices (see Section 5.1.3.2). Algorithms 3 and 7 can be implemented in an embarrassingly parallel way, which is done in our implementation. Algorithms 6 and 10 The software was run with a thread limit `OMP_THREAD_LIMIT=2`.

5.3 Results

An overview of the runtime of the cohomology and the homology algorithm for different data sets, subsample sizes and homology dimensions is given in Figures 1 to 3. These figures show the runtime necessary to compute H_1, H_2 and H_3 (resp. $H^3(N^\bullet)$, $H^4(N^\bullet)$ and $H^5(N^\bullet)$ for the cohomology algorithm). In the case of the homology algorithm, when computing H_d, the numbers also include the time necessary for running `Minimize`(D_{d+1}) (for the chunk algorithm) or `Minimize`(D^{d+1}) (for the chunk* algorithm). The runtime does not include time for I/O, time for generating the (co)boundary matrices of the function-Rips complexes, and, in the case of the cohomology algorithm, time for coning off the input complex. We observed that in general, coning off the input complex is a rather cheap preprocessing step, compared to the total runtime.

Note that the plots use logarithmic scale both for the runtime and the number of $n+1$-simplices. Missing points in the data series occur if program has terminated within five minutes. We see that in most instances, the cohomology algorithm is the fastest method for computing H_2. For H_3, both the homology with chunk* preprocessing and the cohomology algorithm are faster than the homology algorithm with chunk preprocessing. Which one is fastest depends on the instances.

In Tables 2 to 6, we have listed the runtimes for the three algorithms for a fixed size of the subsample for two different computers. Missing entries indicate that the experiment did not terminate within five minutes. The table also includes the the quotient of the runtime of the homology algorithm by the two compared methods (homology algorithm with chunk* preprocessing, and the cohomology algorithm). We see that for H_2, we consistently get the best speedup with the cohomology algorithm. While for H_3, the cohomology algorithm is still consistently faster than the homology algorithm, it is outperformed in some cases by the homology algorithm with chunk* preprocessing.

5.3.1 Matrix representations

The performance of the algorithms depends on the chosen sparse matrix representation. This was already known from one-parameter persistence [12]. It is also known that the impact of the choice of the matrix representation is stronger for cohomological algorithms than for homological ones [12, Table 1, 17, Table 3].

In Tables 7 to 11, we collected runtime data for both versions of the homology algorithm (with chunk(*) preprocessing) and the cohomology algorithm, taken from an implementation with the matrix representation and from an implementation with the vector representation. The columns “speedup” show by which factor the former is faster.

We see that the homology algorithm with chunk preprocessing generally runs slightly slower when using heap matrices. In contrast, chunk* preprocessing can benefit from the heap representation considerably in higher homology dimensions. The cohomology algorithm apparently does not benefit from using heap matrices to the same extend. Comparing Table 8 with 9 and Table 10 with 11 suggests that the impact of the choice of the matrix representation also depends on the machine.

A possible explanation for faster runtimes of the chunk* preprocessing and the cohomology algorithm is the following. The heap representation is better for tall and narrow matrices, which is typically the shape of the coboundary matrices of a Vietoris–Rips complex. Namely, to add two columns in vector representation, one has to iterate over all entries of both summands, and write the symmetric difference of the two vectors to another vector. In contrast, the heap representation allows an efficient in-place addition by copying the entries of one column into the heap representing the other. Typically, most of the runtime goes into reducing a few columns that end up with a high number of non-zero entries. In this case, the heap representation avoids iterating over these entries for every addition to such a column.

Our software also contains code for the implicit matrix representation described in Section 5.1.3.3. However, experiments carried out at an earlier step showed no performance benefit from this matrix representation. Therefore, we did not include this approach in the experiments in this section.

5.3.2 Variants of the cohomology algorithm

We have claimed above that the cohomology algorithm works best without chunk(*) preprocessing, and with a sparsification step (Algorithm 22) added between the two for-loops of Algorithm 10. To justify this, we have run the cohomology algorithm on the same datasets. Apart from the “normal” variant, which includes these steps and which is also used for the other experiments, we

- leave out sparsification,
- leave out clearing,
- use the simplicial cone to make the complex eventually acyclic, instead of Section 5.1.4,

- do not make the input eventually acyclic at all,[3]
- apply chunk or chunk* preprocessing before we run the cohomology algorithm.

The results are listed in Tables 16 to 19. We see that sparsification may introduce an additional cost; however, it can also reduce the total runtime considerably. The last is particularly true for the vector representation; see Table 18 In general (for both matrix representations), the potential runtime savings are much higher than the additional runtime, which is why we enable sparsification as the default option. We see, however, that the benefit from sparsification is considerably smaller for $H^5(N^\bullet)$ than for $H^4(N^\bullet)$

Clearing, on the other hand, has no significant impact at all. This is because the runtime for the first for-loop in Algorithm 10, which is the step where, without clearing, reduction of columns to zero would happen, is negligible in comparison with all other steps of the cohomology algorithm; see Section 5.3.4 below.

With regard to coning off the input complex, we see that the homology-basis-based approach is typically more efficient than taking the simplicial cone. Possibly, this is because the former method constructs the smallest possible chain complex with the desired properties, while the latter method may construct an unnecessarily large complex. This is particularly true for clique complexes. Comparing these numbers to the column “no cone“, which contains runtimes of the algorithm without coning off the input complex at all, we see that this step does not introduce a significant overhead.

Lastly, we see that the algorithm does not profit from chunk(*) preprocessing. Indeed, the total runtimes of chunk(*) preprocessing and a following run of the cohomology algorithm maybe much longer than just running the cohomology algorithm. This is particularly true in higher homology dimensions. See Section 5.3.3 for details

5.3.3 Chunk preprocessing

In Tables 12 to 15, we have listed the runtimes for the chunk and the chunk* preprocessing alone, together with the runtimes of running the homology and the cohomology algorithm without this preprocessing, and in combination with the chunk(*) preprocessing; both for the heap and the vector representation. The tables confirm that the homology algorithm profits from chunk and even more from chunk* preprocessing, and that the cohomology algorithm does not run faster in combination with the chunk(*) preprocessing. In fact, we see that chunk(*) preprocessing already takes longer than the cohomology algorithm would take without any preprocessing.

A possible explanation for this is that the chunk algorithm is known, despite decreasing the number of rows and columns of the input boundary matrices, to potentially increase the total number of non-zero entries. In other words, the chunk preprocessing may increase the density of the matrix considerably. This is particularly true for full function–Rips complexes [73, Table 3]. We hypothesize the same is true for chunk* preprocessing.

5.3.4 Steps of the cohomology algorithm

In Tables 20 to 27, we have listed the runtimes for the steps in the cohomology algorithm Algorithm 11. The runtime for the call to `Bireduce`() is spread among the first three columns, listing the runtime of the first and second reduction loop of Algorithm 10, and the cost of sparsifying the matrix between the two loops (Algorithm 22). The last column lists the total runtime for computing the minimal free resolution, which may comprise smaller values not part of the preceding steps.

[3] In this case, the computed resolution of $H^\bullet(N^\bullet(K_*))$ may not be correct, and $H^\bullet(N^\bullet(K_*))$ may not be isomorphic to $H^{\bullet-2}(K_*)$. We included this case nevertheless in order to demonstrate that making the input eventually acyclic does not increase runtime too much.

From these tables, we see that the runtime is dominated by the calls to `Minimize`() and to `KerAndMgsWithKer`(). Although sparsification is called before the second reduction loop, it does not improve the performance of this step in most cases (except for $O(3)$) when using heap matrices, but only the runtime of the following calls to `Minimize`() and `KerAndMgsWithKer`(). Interestingly, the benefit arising from sparsification seems to be much bigger for $H^4(N^\bullet)$ than for $H^5(N^\bullet)$.

Summary and concluding remarks

In this thesis, we have shown that it is possible to extend several aspects of the duality between persistent homology and cohomology from the context of one-parameter persistence to two- and in parts also to multi-parameter persistence. Specifically, for any $n \geq$ and for a one-critically n-parameter filtered simplicial complex K_* we defined a chain complex $N^\bullet(K_*)$ of free modules functorial in K_* (Definition 3.1.2 and Lemma 3.2.4). In the special case $n = 1$, we have that $N^\bullet(K_*) = K^\bullet(K, K_*)$. The well-known correspondence between absolute and relative one-parameter persistent cohomology, which is induced by the long exact sequence of relative cohomology, can be generalized for any number n of parameters to a natural isomorphism

$$H^d(K_*) \cong H^{d+n}(N^\bullet(K_*)),$$

assuming that K_* is eventually acyclic, i.e., $H_\bullet(K_z) \neq 0$ for only finitely many z. This follows from the Calabi–Yau property of persistence modules (Theorem 3.2.8). The same property also allowed us to find a correspondence between minimal free resolutions of $H^d(K_*)$ and $H_d(K_*)$ (Theorem E). This is a generalization of the correspondence of barcodes of $H^d(K_*)$ and $H_d(K_*)$ in one-parameter persistence. Lastly, we used the Calabi–Yau property to devise an algorithm that efficiently computes $H^{d+n}(N^\bullet(K_*))$ for $n = 2$ (Algorithm 11).

This algorithm computes a minimal free resolution of $H^{d+2}(N^\bullet(K_*))$ from the $(d+1)$st coboundary matrix $\delta^{d+1}\colon N^d(K_*) \to N^{d+1}(K_*)$ and not, as a naive algorithm would do, from δ^{d+2} and δ^{d+3}. This generalizes a corresponding statement in one-parameter persistence, which states that if $K \coloneqq \operatorname{colim} K_*$ is acyclic, then a barcode of $H^{d+1}(K, K_*)$ can be computed just from $\delta^{d+1}\colon C^d(K, K_*) \to C^{d+1}(K, K_*)$, i.e., without considering δ^{d+2}.

Our motivation for this was to compute the persistent homology of two-parameter filtrations of Vietoris–Rips complexes. In one-parameter persistence, Vietoris–Rips complexes have become feasible in practice only through optimization strategies such as clearing, which rely on computing persistent cohomology. The fact that $H^{d+2}(N^\bullet(K_*))$ can be computed from the $(d+1)$st coboundary matrix of $N^\bullet(K_*)$ is vital for the practicality of our algorithm, because for Vietoris–Rips complexes and other clique complexes, the higher dimensional coboundary matrices would be prohibitively large. Furthermore, our algorithm offers the possibility to certain optimizations similar to clearing.

We evaluated the practicality of our algorithm in experiments, using a software implementation that we made publicly available [94]. In these experiments, we compared the runtime of our algorithm with the runtime of the state of the art algorithm for minimal free resolutions of two-parameter persistent homology, which is the Lesnick–Wright (LW) algorithm in conjunction with chunk preprocessing. These experiments show that our algorithm is practical and efficient indeed, and, when computing $H_d(K_*)$ for $d > 1$, outperforms the LW-algorithm by a factor of often more than 10. In these experiments, we also evaluated the impact of different technical details of the implementation, such as sparse matrix representations and others.

Nevertheless, the margin by which our algorithm outperforms the LW algorithm is much smaller than the difference between the Standard Algorithm and the Clearing Algorithm in one-parameter persistent cohomology. We observe that in many cases, the performance of our algorithm suffers from matrix fill-up. This is not a particularity of our algorithm, but also affects the LW algorithm. In fact, it comes from $\mathbf{Z}^2$ not being totally ordered that in practice, a (co)boundary matrix that is compatible with a $\mathbf{Z}^2$-indexed filtration may be much harder to reduce than a (co)boundary matrix coming from a **Z**-indexed filtration. We assume it is vital for future improvements in multi-parameter persistent (co)homology to overcome the performance bounds posed by matrix fill-up.

Memory-efficient strategies, such as not storing the matrix being reduced in memory, or exploiting frequent special cases that can be handled efficiently (such as apparent pairs) may come in handy. Because our software was mainly designed as a versatile framework for two-parameter persistent cohomology, and in order to investigate the practicality of the algorithms presented in this thesis, these ideas are not implemented in our software, notwithstanding their conceptual simplicity.

We also remark that there is no known algorithm that computes minimal free resolutions of n-parameter persistent homology or cohomology in polynomial time for $n > 2$. In particular, the LW algorithm does not generalize to more than two parameters. The same is true for our algorithm, although most of the underlying algebraic properties (such as the n-Calabi–Yau property of n-parameter persistence modules, which underlies the isomorphism $H^d(K_*) \cong H^{d+n}(N^\bullet(K_*))$, and the correspondence between minimal free resolutions of $H_d(K_*)$ and $H^d(K_*)$) hold for any number of parameters. We thus assume that closing this gap may increase the feasibility of multi-parameter persistence significantly.

Bibliography

[1] Manu Aggarwal and Vipul Periwal. *Dory: Overcoming Barriers to Computing Persistent Homology.* 2021. arXiv: 2103.05608.

[2] Ángel Javier Alonso, Michael Kerber, and Siddharth Pritam. "Filtration-Domination in Bifiltered Graphs." In: *2023 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, 2023, pp. 27–38. DOI: 10.1137/1.9781611977561.ch3.

[3] Ibrahim Assem, Daniel Simson, and Andrzej Skowronski. *Elements of the Representation Theory of Associative Algebras.* Vol. 1. London Mathematical Society Student Texts 65. Cambridge: Cambridge University Press, 2006. ISBN: 978-0-511-34545-6.

[4] Dominique Attali, André Lieutier, and David Salinas. "Vietoris–Rips Complexes Also Provide Topologically Correct Reconstructions of Sampled Shapes." In: *Computational Geometry* 46.4 (2013), pp. 448–465. DOI: 10.1016/j.comgeo.2012.02.009.

[5] Gorô Azumaya. "Corrections and Supplementaries to My Paper Concerning Krull-Remak-Schmidt's Theorem." In: *Nagoya Mathematical Journal* 1 (1950), pp. 117–124. DOI: 10.1017/S002776300002290X.

[6] Sergey Alexandrovich Barannikov. "The Framed Morse Complex and Its Invariants." In: *Singularities and Bifurcations.* Vol. 21. Adv. Soviet Math. Providence, RI: American Mathematical Society, 1994, pp. 93–115. DOI: 10.1090/advsov/021.

[7] Hyman Bass. "Big Projective Modules Are Free." In: *Illinois Journal of Mathematics* 7.1 (1963), pp. 24–31. DOI: 10.1215/ijm/1255637479.

[8] Ulrich Bauer. *Ripser.* Version 1.1. 2019. URL: https://github.com/Ripser/ripser.

[9] Ulrich Bauer. "Ripser: Efficient Computation of Vietoris–Rips Persistence Barcodes." In: *Journal of Applied and Computational Topology* 5.3 (2021), pp. 391–423. DOI: 10.1007/s41468-021-00071-5.

[10] Ulrich Bauer and Herbert Edelsbrunner. "The Morse Theory of Čech and Delaunay Complexes." In: *Transactions of the American Mathematical Society* 369.5 (2017), pp. 3741–3762. DOI: 10.1090/tran/6991.

[11] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. "Clear and Compress: Computing Persistent Homology in Chunks." In: *Topological Methods in Data Analysis and Visualization III.* Ed. by Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert. Cham: Springer International Publishing, 2014, pp. 103–117. DOI: 10.1007/978-3-319-04099-8_7.

[12] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. "PHAT – Persistent Homology Algorithms Toolbox." In: *Mathematical Software – ICMS 2014.* Ed. by Hoon Hong and Chee Yap. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 137–143. ISBN: 978-3-662-44199-2.

[13] Ulrich Bauer, Michael Kerber, Fabian Roll, and Alexander Rolle. "A Unified View on the Functorial Nerve Theorem and Its Variations." In: *Expositiones Mathematicae* (2023). DOI: 10.1016/j.exmath.2023.04.005.

[14] Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. "Efficient Two-Parameter Persistence Computation via Cohomology." In: *39th International Symposium on Computational Geometry (SoCG 2023)* (Dallas). Ed. by Erin W. Chambers and Joachim Gudmundsson. Leibniz International Proceedings in Informatics (LIPIcs) 258. Dagstuhl: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 15:1–15:17. DOI: 10.4230/LIPIcs.SoCG.2023.15. arXiv: 2303.11193.

[15] Ulrich Bauer and Michael Lesnick. "Induced Matchings and the Algebraic Stability of Persistence Barcodes." In: *Journal of Computational Geometry* 6.2 (2015), pp. 162–191. DOI: 10.20382/jocg.v6i2a9.

[16] Ulrich Bauer and Michael Lesnick. "Persistence Diagrams as Diagrams: A Categorification of the Stability Theorem." In: *Topological Data Analysis.* Ed. by Nils A. Baas, Gunnar E. Carlsson, Gereon Quick, Markus Szymik, and Marius Thaule. Abel Symposia. Cham: Springer International Publishing, 2020, pp. 67–96. DOI: 10.1007/978-3-030-43408-3_3.

[17] Ulrich Bauer, Talha Bin Masood, Barbara Giunti, Guillaume Houry, Michael Kerber, and Abhishek Rathod. *Keeping It Sparse: Computing Persistent Homology Revised.* 2022. arXiv: 2211.09075.

[18] Ulrich Bauer and Fabian Roll. *Connecting Discrete Morse Theory and Persistence: Wrap Complexes and Lexicographic Optimal Cycles.* 2022. arXiv: 2212.02345.

[19] Ulrich Bauer and Maximilian Schmahl. *Lifespan Functors and Natural Dualities in Persistent Homology.* 2021. arXiv: 2012.12881.

[20] Ulrich Bauer and Luis Scoccola. *Generic Two-Parameter Persistence Modules Are Nearly Indecomposable.* 2022. arXiv: 2211.15306.

[21] Christine Berkesch and Frank-Olaf Schreyer. "Syzygies, Finite Length Modules,and Random Curves." In: *Commutative Algebra and Noncommutative Algebraic Geometry.* Ed. by David Eisenbud, Srikanth B. Iyengar, Anurag K. Singh, J. Joby Stafford, and Michel Van de Bergh. Vol. 1. MSRI Publications 67. Cambridge University Press, 2015, pp. 25–52. ISBN: 978-1-107-06562-8.

[22] Silvia Biasotti, Andrea Cerri, Patrizio Frosini, Daniela Giorgi, and Claudia Landi. "Multidimensional Size Functions for Shape Comparison." In: *Journal of Mathematical Imaging and Vision* 32.2 (2008), pp. 161–179. DOI: 10.1007/s10851-008-0096-z.

[23] Andrew J. Blumberg, Itamar Gal, Michael A. Mandell, and Matthew Pancia. "Robust Statistics, Hypothesis Testing, and Confidence Intervals for Persistent Homology on Metric Measure Spaces." In: *Foundations of Computational Mathematics* 14.4 (2014), pp. 745–789. DOI: 10.1007/s10208-014-9201-4.

[24] Andrew J. Blumberg and Michael Lesnick. "Stability of 2-Parameter Persistent Homology." In: *Foundations of Computational Mathematics* (2022). DOI: 10.1007/s10208-022-09576-6.

[25] Omer Bobrowski, Sayan Mukherjee, and Jonathan E. Taylor. "Topological Consistency via Kernel Estimation." In: *Bernoulli* 23.1 (2017), pp. 288–328. DOI: 10.3150/15-BEJ744.

[26] Magnus Bakke Botnan and William Crawley-Boevey. "Decomposition of Persistence Modules." In: *Proceedings of the American Mathematical Society* 148.11 (2020), pp. 4581–4596. DOI: 10.1090/proc/14790.

[27] Magnus Bakke Botnan, Steffen Oppermann, and Steve Oudot. "Signed Barcodes for Multi-Parameter Persistence via Rank Decompositions." In: *38th International Symposium on Computational Geometry (SoCG 2022).* Ed. by Xavier Goaoc and Michael Kerber. Vol. 224. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 19:1–19:18. DOI: 10.4230/LIPIcs.SoCG.2022.19.

[28] Peter Bubenik and Jonathan A. Scott. "Categorification of Persistent Homology." In: *Discrete & Computational Geometry* 51.3 (2014), pp. 600–627. DOI: 10.1007/s00454-014-9573-x.

[29] Mickaël Buchet, Frédéric Chazal, Steve Y. Oudot, and Donald R. Sheehy. "Efficient and Robust Persistent Homology for Measures." In: *Computational Geometry* 58 (2016), pp. 70–96. DOI: 10.1016/j.comgeo.2016.07.001.

[30] Mickaël Buchet and Emerson G. Escolar. "Realizations of Indecomposable Persistence Modules of Arbitrarily Large Dimension." In: ed. by Bettina Speckmann and Csaba D. Tóth. Vol. 99. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 15:1–15:13. DOI: 10.4230/LIPIcs.SoCG.2018.15.

[31] Gunnar Carlsson. “Topology and Data.” In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308. DOI: 10.1090/S0273-0979-09-01249-X.

[32] Gunnar Carlsson, Tigran Ishkhanov, Vin de Silva, and Afra Zomorodian. “On the Local Behavior of Spaces of Natural Images.” In: *International Journal of Computer Vision* 76.1 (2008), pp. 1–12. DOI: 10.1007/s11263-007-0056-x.

[33] Gunnar Carlsson, Gurjeet Singh, and Afra Zomorodian. “Computing Multidimensional Persistence.” In: *Algorithms and Computation*. Ed. by Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 730–739. DOI: 10.1007/978-3-642-10631-6_74.

[34] Gunnar Carlsson and Afra Zomorodian. “The Theory of Multidimensional Persistence.” In: *Discrete & Computational Geometry* 42.1 (2009), pp. 71–93. DOI: 10.1007/s00454-009-9176-0.

[35] Mathieu Carrière and Andrew J. Blumberg. “Multiparameter Persistence Images for Topological Machine Learning.” In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS’20. Red Hook, NY, USA: Curran Associates Inc., 2020, pp. 22432–22444. ISBN: 978-1-71382-954-6.

[36] Wojciech Chacholski, Martina Scolamiero, and Francesco Vaccarino. *Combinatorial Presentation of Multidimensional Persistent Homology*. 2014. arXiv: 1409.7936.

[37] Wojciech Chachólski, René Corbet, and Anna-Laura Sattelberger. *The Shift-Dimension of Multipersistence Modules*. 2021. arXiv: 2112.06509.

[38] Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Mémoli, and Steve Y. Oudot. “Gromov-Hausdorff Stable Signatures for Shapes Using Persistence.” In: *Computer Graphics Forum* 28.5 (2009), pp. 1393–1403. DOI: 10.1111/j.1467-8659.2009.01516.x.

[39] Frédéric Chazal, David Cohen-Steiner, and Quentin Mérigot. “Geometric Inference for Probability Measures.” In: *Foundations of Computational Mathematics* 11.6 (2011), pp. 733–751. DOI: 10.1007/s10208-011-9098-0.

[40] Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics. Cham: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-42545-0.

[41] Frédéric Chazal, Vin de Silva, and Steve Oudot. “Persistence Stability for Geometric Complexes.” In: *Geometriae Dedicata* 173.1 (2014), pp. 193–214. DOI: 10.1007/s10711-013-9937-z.

[42] Frédéric Chazal, Brittany Fasy, Fabrizio Lecci, Bertrand Michel, Alessandro Rinaldo, Alessandro Rinaldo, and Larry Wasserman. “Robust Topological Inference: Distance to a Measure and Kernel Distance.” In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5845–5884.

[43] Frédéric Chazal, Marc Glisse, Catherine Labruère, and Bertrand Michel. “Convergence Rates for Persistence Diagram Estimation in Topological Data Analysis.” In: *Journal of Machine Learning Research* 16.110 (2015), pp. 3603–3635. URL: http://jmlr.org/papers/v16/chazal15a.html.

[44] Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. “Persistence-Based Clustering in Riemannian Manifolds.” In: *Journal of the ACM* 60.6 (2013), 41:1–41:38. DOI: 10.1145/2535927.

[45] Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. “Scalar Field Analysis over Point Cloud Data.” In: *Discrete & Computational Geometry* 46.4 (2011), pp. 743–775. DOI: 10.1007/s00454-011-9360-x.

[46] Chao Chen and Michael Kerber. “Persistent Homology Computation with a Twist.” In: 2011. URL: https://eurocg11.inf.ethz.ch/abstracts/22.pdf.

[47] Claude Cibils and Pu Zhang. “Calabi–Yau Objects in Triangulated Categories.” In: *Transactions of the American Mathematical Society* 361.12 (2009), pp. 6501–6519. DOI: 10.1090/S0002-9947-09-04682-0.

[48] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. “Stability of Persistence Diagrams.” In: *Discrete & Computational Geometry* 37 (2007), pp. 103–120. DOI: 10.1007/s00454-006-1276-5.

[49] David Cohen-Steiner, André Lieutier, and Julien Vuillamy. "Lexicographic Optimal Homologous Chains and Applications to Point Cloud Triangulations." In: *Discrete & Computational Geometry* 68.4 (2022), pp. 1155–1174. DOI: 10.1007/s00454-022-00432-6.

[50] René Corbet and Michael Kerber. "The Representation Theorem of Persistence Revisited and Generalized." In: *Journal of Applied and Computational Topology* 2.1 (2018), pp. 1–31. DOI: 10.1007/s41468-018-0015-3.

[51] René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. "Computing the Multicover Bifiltration." In: *37th International Symposium on Computational Geometry (SoCG 2021)*. Ed. by Kevin Buchin and Éric Colin de Verdière. Vol. 189. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 27:1–27:17. DOI: 10.4230/LIPIcs.SoCG.2021.27.

[52] William Crawley-Boevey. "Decomposition of Pointwise Finite-Dimensional Persistence Modules." In: *Journal of Algebra and Its Applications* 14.05 (2015), p. 1550066. DOI: 10.1142/S0219498815500668.

[53] Matija Čufar, Žiga Virk, NZ Institute for Advanced Study, Massey University, Auckland, New Zealand, and University of Ljubljana and Institute IMFM, Ljubljana, Slovenia. "Fast Computation of Persistent Homology Representatives with Involuted Persistent Homology." In: *Foundations of Data Science* 5.4 (2023), pp. 466–479. DOI: 10.3934/fods.2023006.

[54] Vin de Silva and Robert Ghrist. "Coverage in Sensor Networks via Persistent Homology." In: *Algebraic & Geometric Topology* 7.1 (2007), pp. 339–358. DOI: 10.2140/agt.2007.7.339.

[55] Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. "Dualities in Persistent (Co)Homology." In: *Inverse Problems* 27.12 (2011), p. 124003. DOI: 10.1088/0266-5611/27/12/124003.

[56] Tamal K. Dey, Woojin Kim, and Facundo Mémoli. "Computing Generalized Rank Invariant for 2-Parameter Persistence Modules via Zigzag Persistence and Its Applications." In: *38th International Symposium on Computational Geometry (SoCG 2022)*. Ed. by Xavier Goaoc and Michael Kerber. Vol. 224. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, 34:1–34:17. DOI: 10.4230/LIPIcs.SoCG.2022.34.

[57] Herbert Edelsbrunner. "Surface Reconstruction by Wrapping Finite Sets in Space." In: *Discrete and Computational Geometry*. Ed. by Boris Aronov, Saugata Basu, János Pach, and Micha Sharir. Vol. 25. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 379–404. DOI: 10.1007/978-3-642-55566-4_17.

[58] Herbert Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Providence, R.I: American Mathematical Society, 2010. 241 pp. ISBN: 978-0-8218-4925-5.

[59] Herbert Edelsbrunner and John Harer. "Persistent Homology—a Survey." In: *Contemporary Mathematics*. Ed. by Jacob E. Goodman, János Pach, and Richard Pollack. Vol. 453. Providence, Rhode Island: American Mathematical Society, 2008, pp. 257–282. DOI: 10.1090/conm/453/08802.

[60] Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. "On the Shape of a Set of Points in the Plane." In: *IEEE Transactions on Information Theory* 29.4 (1983), pp. 551–559. DOI: 10.1109/TIT.1983.1056714.

[61] Herbert Edelsbrunner and Dmitriy Morozov. "Persistent Homology." In: *Handbook of Discrete and Computational Geometry*. 3rd ed. Chapman and Hall/CRC, 2017.

[62] Herbert Edelsbrunner and Georg Osang. "A Simple Algorithm for Higher-Order Delaunay Mosaics and Alpha Shapes." In: *Algorithmica* 85.1 (2023), pp. 277–295. DOI: 10.1007/s00453-022-01027-6.

[63] Herbert Edelsbrunner and Georg Osang. "The Multi-cover Persistence of Euclidean Balls." In: *34th International Symposium on Computational Geometry (SoCG 2018)*. Ed. by Bettina Speckmann and Csaba D. T{\'o}th. Vol. 99. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 1–14. DOI: 10.4230/LIPICS.SOCG.2018.34.

[64] Edelsbrunner, Letscher, and Zomorodian. “Topological Persistence and Simplification.” In: *Discrete & Computational Geometry* 28.4 (2002), pp. 511–533. DOI: 10.1007/s00454-002-2885-2.

[65] David Eisenbud. *Commutative Algebra.* Vol. 150. Graduate Texts in Mathematics. New York, NY: Springer New York, 1995. DOI: 10.1007/978-1-4612-5350-1.

[66] Charles Epstein, Gunnar Carlsson, and Herbert Edelsbrunner. “Topological Data Analysis.” In: *Inverse Problems* 27.12 (2011), p. 120201. DOI: 10.1088/0266-5611/27/12/120201.

[67] Burcin Eröcal, Oleksandr Motsak, Frank-Olaf Schreyer, and Andreas Steenpass. “Refined Algorithms to Compute Syzygies.” In: *Journal of Symbolic Computation* 74 (2016), pp. 308–327. DOI: 10.1016/j.jsc.2015.07.004.

[68] Alberto Facchini. “The Krull-Schmidt Theorem.” In: *Handbook of Algebra.* Ed. by Michiel Hazewinkel. Vol. 3. North-Holland, 2003, pp. 357–397. DOI: 10.1016/S1570-7954(03)80066-9.

[69] Robin Forman. “A User’s Guide to Discrete Morse Theory.” In: *Séminaire Lotharingien de Combinatoire* 48 (2008). URL: https://www.emis.de/journals/SLC/wpapers/s48forman.pdf.

[70] G. Frobenius and L. Stickelberger. “Ueber Gruppen von vertauschbaren Elementen.” In: *Journal für die reine und angewandte Mathematik* 86 (1879), pp. 217–262. URL: https://eudml.org/doc/148395.

[71] Ulderico Fugacci and Michael Kerber. *Chunk Reduction for Multi-Parameter Persistent Homology.* 2019. arXiv: 1812.08580.

[72] Ulderico Fugacci and Michael Kerber. “Chunk Reduction for Multi-Parameter Persistent Homology.” In: *35th International Symposium on Computational Geometry (SoCG 2019).* Ed. by Gill Barequet and Yusu Wang. Vol. 129. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 37:1–37:14. DOI: 10.4230/LIPIcs.SoCG.2019.37.

[73] Ulderico Fugacci, Michael Kerber, and Alexander Rolle. “Compression for 2-Parameter Persistent Homology.” In: *Computational Geometry* 109 (2023), p. 101940. DOI: 10.1016/j.comgeo.2022.101940.

[74] Peter Gabriel. “Unzerlegbare Darstellungen I.” In: *manuscripta mathematica* 6.1 (1972), pp. 71–103. DOI: 10.1007/BF01298413.

[75] Oliver Gäfvert and Wojciech Chachólski. *Stable Invariants for Multiparameter Persistence.* 2021. arXiv: 1703.03632.

[76] Robert Ghrist. “Barcodes: The Persistent Topology of Data.” In: *Bulletin of the american mathematical society* 45 (2007), pp. 61–76. DOI: 10.1090/S0273-0979-07-01191-3.

[77] Victor Ginzburg. *Calabi-Yau Algebras.* 2007. arXiv: math/0612139.

[78] Barbara Giunti, Guillaume Houry, and Michael Kerber. “Average Complexity of Matrix Reduction for Clique Filtrations.” In: *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation.* Villeneuve-d’Ascq France: ACM, 2022, pp. 187–196. DOI: 10.1145/3476446.3535474.

[79] Dan Halperin, Michael Kerber, and Doron Shaharabani. “The Offset Filtration of Convex Objects.” In: *Algorithms - ESA 2015.* Ed. by Nikhil Bansal and Irene Finocchi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 705–716. ISBN: 978-3-662-48350-3.

[80] Heather A. Harrington, Nina Otter, Hal Schenck, and Ulrike Tillmann. “Stratifying Multiparameter Persistent Homology.” In: *SIAM Journal on Applied Algebra and Geometry* 3.3 (2019), pp. 439–471. DOI: 10.1137/18M1224350.

[81] Gregory Henselman-Petrusek. *Eirene.* 2021. URL: https://github.com/Eetion/Eirene.jl.

[82] Michael Höppner and Helmut Lenzing. “Projective Diagrams over Partially Ordered Sets Are Free.” In: *Journal of Pure and Applied Algebra* 20.1 (1981), pp. 7–12. DOI: 10.1016/0022-4049(81)90045-1.

[83] Irving Kaplansky. “Projective Modules.” In: *Annals of Mathematics* 68.2 (1958), pp. 372–377. DOI: 10.2307/1970252. JSTOR: 1970252.

[84] Bernhard Keller. "Calabi–Yau Triangulated Categories." In: *Trends in Representation Theory of Algebras and Related Topics*. Ed. by Andrzej Skowroński. 1st ed. EMS Series of Congress Reports. EMS Press, 2008, pp. 467–489. DOI: 10.4171/062-1/11.

[85] Michael Kerber. *Mpfree*. 2021. URL: https://bitbucket.org/mkerber/mpfree.

[86] Michael Kerber and Alexander Rolle. "Fast Minimal Presentations of Bi-graded Persistence Modules." In: *2021 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*. Ed. by Martin Farach-Colton and Sabine Storandt. Proceedings. Society for Industrial and Applied Mathematics, 2021, pp. 207–220. DOI: 10.1137/1.9781611976472.16.

[87] Keunsu Kim and Jae-Hun Jung. *Exact Multi-Parameter Persistent Homology of Time-Series Data: Fast and Variable One-Dimensional Reduction of Multi-Parameter Persistence Theory*. 2022. arXiv: 2211.03337.

[88] Woojin Kim and Facundo Memoli. "Spatio-Temporal Persistent Homology for Dynamic Metric Spaces." In: *Discrete & Computational Geometry* 66.3 (2021), pp. 831–875. DOI: 10.1007/s00454-019-00168-w.

[89] Woojin Kim and Facundo Mémoli. "Generalized Persistence Diagrams for Persistence Modules over Posets." In: *Journal of Applied and Computational Topology* 5.4 (2021), pp. 533–581. DOI: 10.1007/s41468-021-00075-1.

[90] Woojin Kim and Samantha Moore. *Bigraded Betti Numbers and Generalized Persistence Diagrams*. 2022. arXiv: 2111.02551.

[91] Kevin P. Knudson. "A Refinement of Multi-Dimensional Persistence." In: *Homology, Homotopy and Applications* 10.1 (2008), pp. 259–281. DOI: 10.4310/HHA.2008.v10.n1.a11.

[92] Roland Kwitt, Stefan Huber, Marc Niethammer, Weili Lin, and Ulrich Bauer. "Statistical Topological Data Analysis - a Kernel Perspective." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 3070–3078.

[93] Roberto La Scala and Mike E. Stillman. "Strategies for Computing Minimal Free Resolutions." In: *Journal of Symbolic Computation* 26.4 (1998), pp. 409–431. DOI: 10.1006/jsco.1998.0221.

[94] Fabian Lenzen. *2pac. (2-Parameter Persistent Cohomology)*. A C++ software package for two-parameter persistent cohomology. 2023. URL: https://gitlab.com/flenzen/2-parameter-persistent-cohomology.

[95] Fabian Lenzen. *Cell Structure for the Cyclooctane Conformation Space*. inpreparation. 2023.

[96] Michael Lesnick. "The Theory of the Interleaving Distance on Multidimensional Persistence Modules." In: *Foundations of Computational Mathematics* 15.3 (2015), pp. 613–650. DOI: 10.1007/s10208-015-9255-y.

[97] Michael Lesnick and Michael Kerber. *Scc2020: A File Format for Sparse Chain Complexes in TDA*. 2021. URL: https://bitbucket.org/mkerber/chain_complex_format.

[98] Michael Lesnick and Matthew Wright. *Computing Minimal Presentations and Bigraded Betti Numbers of 2-Parameter Persistent Homology*. 2019. arXiv: 1902.05708.

[99] Michael Lesnick and Matthew Wright. "Computing Minimal Presentations and Bigraded Betti Numbers of 2-Parameter Persistent Homology." In: *SIAM Journal on Applied Algebra and Geometry* 6.2 (2022), pp. 267–298. DOI: 10.1137/20M1388425.

[100] Michael Lesnick and Matthew Wright. *Interactive Visualization of 2-D Persistence Modules*. 2015. arXiv: 1512.00180. Implemented in *RIVET*. Version 1.1.0. 2020. URL: https://rivet.readthedocs.io/en/latest/about.html.

[101] Shawn Martin. *NMTRI: Non-Manifold Surface Reconstruction*. Software by Shawn Martin. 2012. URL: http://www.cs.otago.ac.nz/homepages/smartin/software.php.

[102] Shawn Martin, Aidan Thompson, Evangelos A. Coutsias, and Jean-Paul Watson. "Topology of Cyclo-Octane Energy Landscape." In: *The Journal of Chemical Physics* 132 (2010), p. 234115. DOI: 10.1063/1.3445267.

[103] Shawn Martin and Jean-Paul Watson. "Non-Manifold Surface Reconstruction from High Dimensional Point Cloud Data." In: *Computational Geometry* 44.8 (2011), pp. 427–441. DOI: `10.1016/j.comgeo.2011.05.002`.

[104] Ezra Miller. *Data Structures for Real Multiparameter Persistence Modules.* 2020. arXiv: `1709.08155`.

[105] Dmitriy Morozov. *Persistence Algorithm Takes Cubic Time in the Worst Case.* 2005. URL: `https://www.mrzv.org/publications/worst-case/biogeometry`.

[106] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. "Finding the Homology of Submanifolds with High Confidence from Random Samples." In: *Discrete & Computational Geometry* 39.1-3 (2008), pp. 419–441. DOI: `10.1007/s00454-008-9053-2`.

[107] Nina Otter, Mason A. Porter, Ulrike Tillmann, Peter Grindrod, and Heather A. Harrington. "A Roadmap for the Computation of Persistent Homology." In: *EPJ Data Science* 6.1 (2017), p. 17. DOI: `10.1140/epjds/s13688-017-0109-5`.

[108] Steve Oudot. *Persistence Theory: From Quiver Representations to Data Analysis.* Vol. 209. Mathematical Surveys and Monographs. Providence, Rhode Island: American Mathematical Society, 2015. DOI: `10.1090/surv/209`.

[109] Amit Patel. "Generalized Persistence Diagrams." In: *Journal of Applied and Computational Topology* 1.3-4 (2018), pp. 397–419. DOI: `10.1007/s41468-018-0012-6`.

[110] Irena Peeva. *Graded Syzygies.* London: Springer London, 2011. DOI: `10.1007/978-0-85729-177-6`.

[111] Julián Burella Pérez, Sydney Hauke, Umberto Lupo, Matteo Caorsi, and Alberto Dassatti. *Giotto-Ph: A Python Library for High-Performance Computation of Persistent Homology of Vietoris–Rips Filtrations.* 2021. arXiv: `2107.05412`.

[112] Jeff M. Phillips, Bei Wang, and Yan Zheng. "Geometric Inference on Kernel Density Estimates." In: *31st International Symposium on Computational Geometry (SoCG 2015).* Ed. by Lars Arge and János Pach. Vol. 34. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 857–871. DOI: `10.4230/LIPIcs.SOCG.2015.857`.

[113] Daniel Quillen. "Projective Modules over Polynomial Rings." In: *Inventiones Mathematicae* 36.1 (1976), pp. 167–171. DOI: `10.1007/BF01390008`.

[114] Raphael Reinauer, Matteo Caorsi, and Nicolas Berkouk. *Persformer: A Transformer Architecture for Topological Machine Learning.* 2022. arXiv: `2112.15210`.

[115] Bastian Rieck, Filip Sadlo, and Heike Leitte. "Topological Machine Learning with Persistence Indicator Functions." In: *Topological Methods in Data Analysis and Visualization V.* Ed. by Hamish Carr, Issei Fujishiro, Filip Sadlo, and Shigeo Takahashi. Mathematics and Visualization. Cham: Springer International Publishing, 2020, pp. 87–101. DOI: `10.1007/978-3-030-43036-8_6`.

[116] Alexander Rolle and Luis Scoccola. *Stable and Consistent Density-Based Clustering.* 2021. arXiv: `2005.09048`.

[117] Sara Scaramuccia, Federico Iuricich, Leila De Floriani, and Claudia Landi. "Computing Multiparameter Persistent Homology through a Discrete Morse-based Approach." In: *Computational Geometry* 89 (2020), p. 101623. DOI: `10.1016/j.comgeo.2020.101623`.

[118] Donald R. Sheehy. "A Multicover Nerve for Geometric Inference." In: *Proceedings of the 24th Canadian Conference on Computational Geometry, CCCG 2012, Charlottetown, Prince Edward Island, Canada, August 8-10, 2012.* 2012, pp. 309–314. URL: `http://2012.cccg.ca/papers/paper52.pdf`.

[119] Jacek Skryzalin. "Numeric Invariants from Multidimensional Persistence." Stanford University, 2016. URL: `https://purl.stanford.edu/gv738xh9880`.

[120] Andrei Suslin. "Projective Modules over Polynomial Rings Are Free." In: *Soviet Mathematics* 17.4 (1976), pp. 1160–1164.

[121] J. B. Tenenbaum. "A Global Geometric Framework for Nonlinear Dimensionality Reduction." In: *Science* 290.5500 (2000), pp. 2319–2323. DOI: 10.1126/science.290.5500.2319.

[122] The GUDHI Project. *GUDHI*. 2021. URL: https://gudhi.inria.fr.

[123] The RIVET Developers. *RIVET*. Version 1.1.0. 2020. URL: https://github.com/rivetTDA/rivet/.

[124] Jean-Louis Verdier. "Des Catégories Triangulées et Des Categories Derivées." PhD thesis. Société Mathématique de France, 1996. URL: http://webusers.imj-prg.fr/~georges.maltsiniotis/jlv.html.

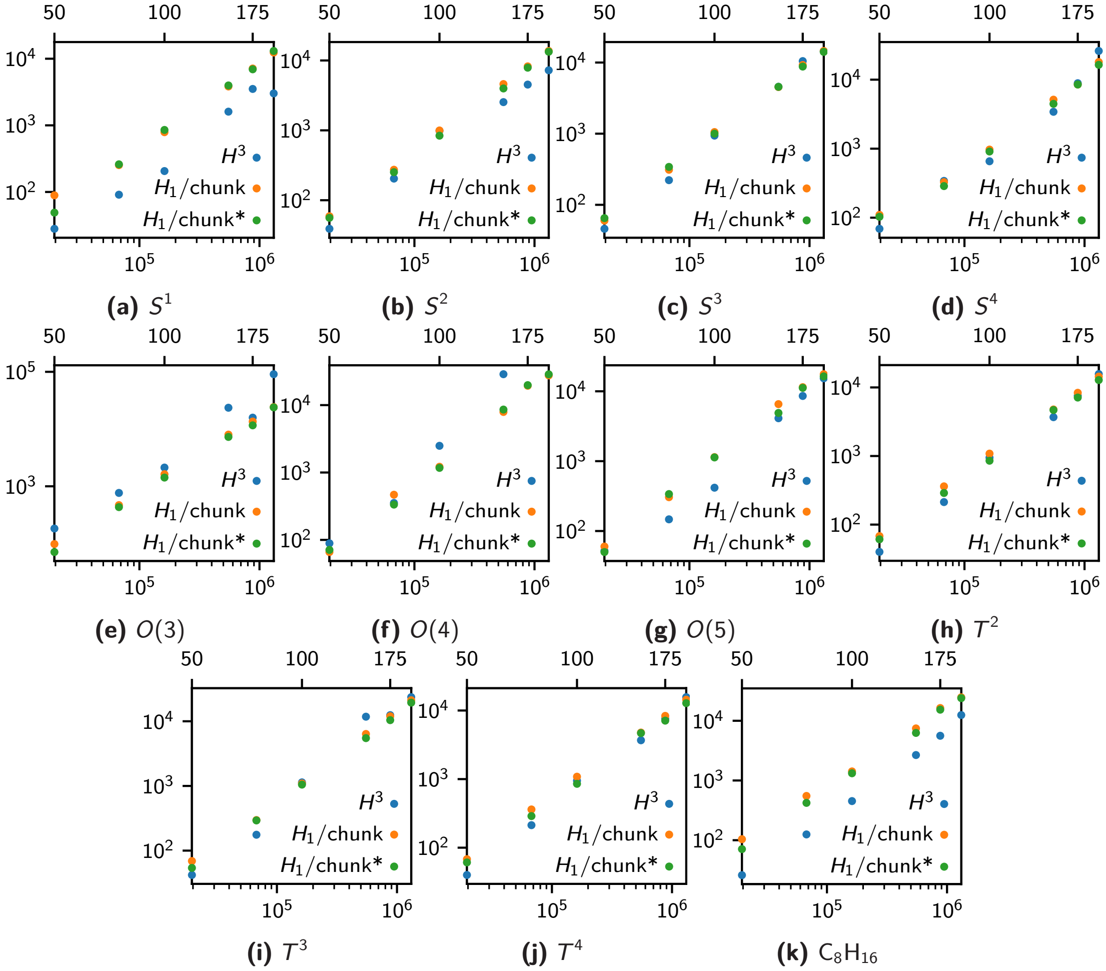
[125] Leopold Vietoris. "Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen." In: *Mathematische Annalen* 97.1 (1927), pp. 454–472. DOI: 10.1007/BF01447877.

[126] Cary Webb. "Decomposition of Graded Modules." In: *Proceedings of the American Mathematical Society* 94.4 (1985), pp. 565–571. DOI: 10.2307/2044864. JSTOR: 2044864.

[127] Charles A. Weibel. *An Introduction to Homological Algebra*. Reprint. 1997, transf. to digital print. Cambridge Studies in Advanced Mathematics 38. Cambridge: Cambridge Univ. Press, 2003. 450 pp. ISBN: 978-0-521-55987-4.

[128] Afra Zomorodian and Gunnar Carlsson. "Computing Persistent Homology." In: *Discrete & Computational Geometry* 33 (2005), pp. 249–274. DOI: 10.1007/s00454-004-1146-y.

Runtime data tables

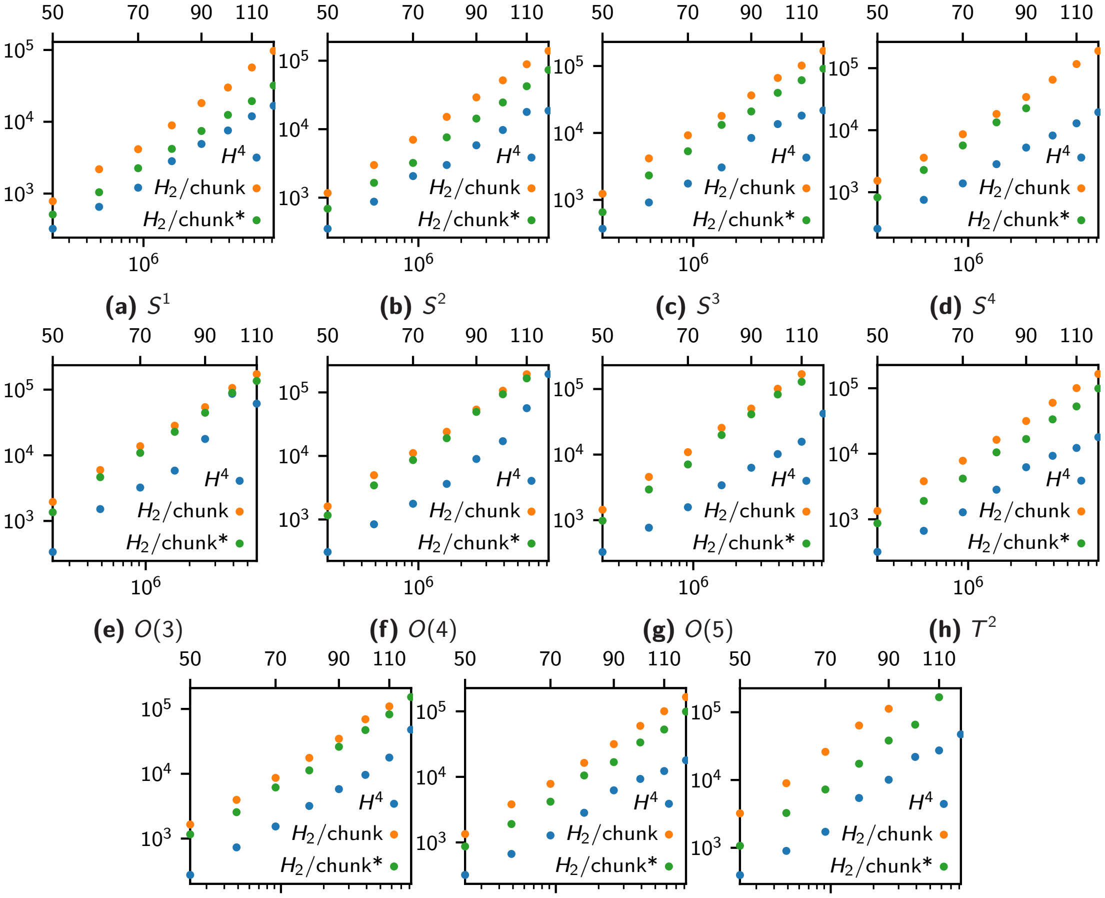


(a) S^1 **(b)** S^2 **(c)** S^3 **(d)** S^4

(e) $O(3)$ **(f)** $O(4)$ **(g)** $O(5)$ **(h)** T^2

(i) T^3 **(j)** T^4 **(k)** C_8H_{16}

Figure 1: Runtimes (ms) of the homology and cohomology algorithm for H_1, plottet over the number of 2-simplices (bottom) and vertices (top) of the subsample. Command line parameters were `-hn1 -cone=2` for the cohomology algorithm, `-cn1 -DC2` for the homology algorithm with chunk and `-cn1 -C2` with chunk* preprocessing.

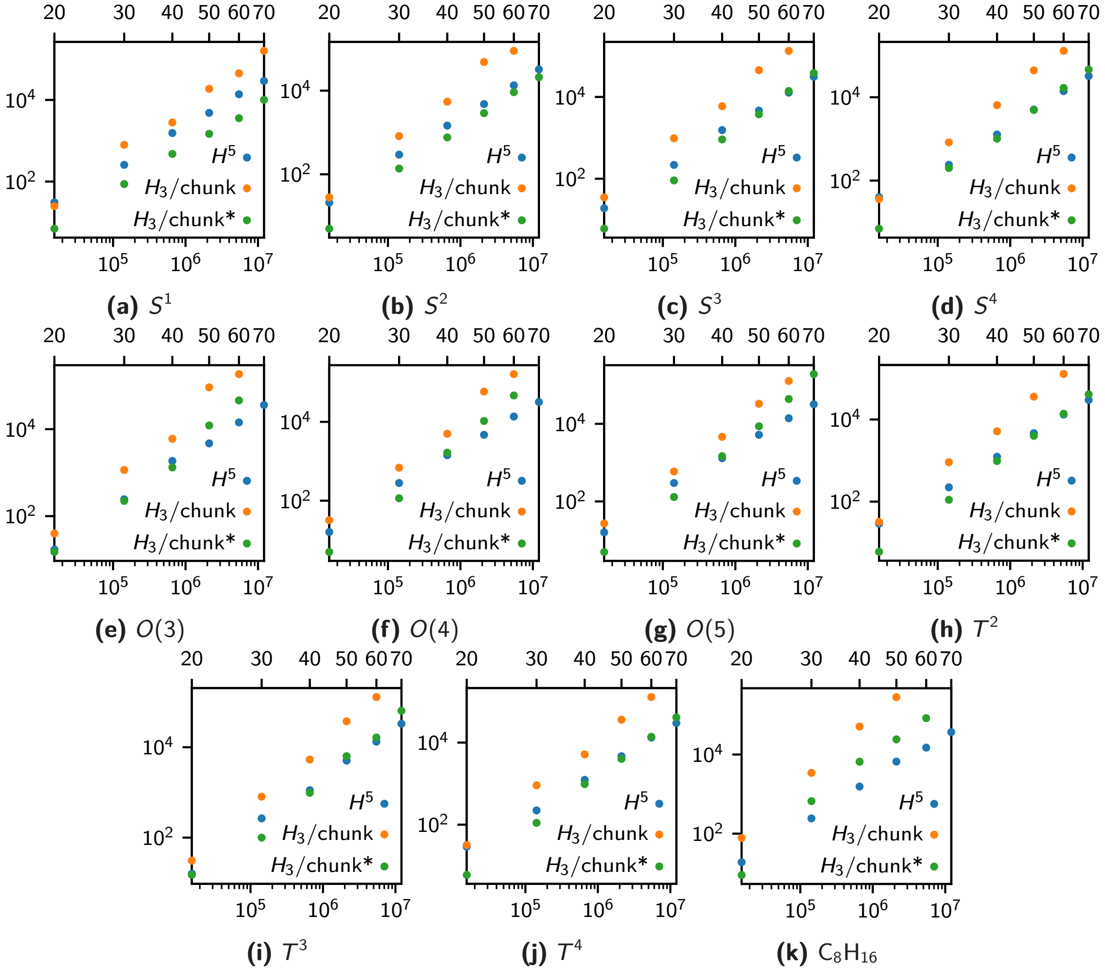


(a) S^1 **(b)** S^2 **(c)** S^3 **(d)** S^4

(e) $O(3)$ **(f)** $O(4)$ **(g)** $O(5)$ **(h)** T^2

(i) T^3 **(j)** T^4 **(k)** C_8H_{16}

Figure 2: Runtimes (ms) of the homology and cohomology algorithm for H_2, plottet over the number of 3-simplices (bottom) and vertices (top) of the subsample. Command line parameters were `-hn2 -cone=2` for the cohomology algorithm `-cn2 -DC3` for the homology algorithm with chunk and `-cn2 -C3` with chunk* preprocessing.



(a) S^1 **(b)** S^2 **(c)** S^3 **(d)** S^4

(e) $O(3)$ **(f)** $O(4)$ **(g)** $O(5)$ **(h)** T^2

(i) T^3 **(j)** T^4 **(k)** C_8H_{16}

Figure 3: Runtimes (ms) of the homology and cohomology algorithm for H_3, plottet over the number of 4-simplices (bottom) and vertices (top) of the subsample. Command line parameters were `-hn3 -cone=2` for the cohomology algorithm `-cn3 -DC4` for the homology algorithm with chunk and `-cn3 -C4` with chunk* preprocessing.

Table 1: Runtime comparison between `mpfree` and our reimplementation of the homology algorithm Algorithm 9 in `2pac`. The table shows the run time to compute a minimal free resolution of H_2 of the function-Rips complex of a subsample of size 80 of the datasets from Table 5.1. Here, we compare `mpfree` with the version of `2pac` that uses the vector representation. Although `mpfree` often is slightly faster, the runtimes do not differ much. In contrast to the other experiments, these were run on a MacBook Pro 2017 with a 2.3 GHz Dual-Core Intel Core i5 CPU and 16 GiB RAM.Runtime comparison between `mpfree` and `2pac`.

	mpfree		2pac	
	chunk	H_2	chunk	H_2
sample				
1-sphere	7 348	471	9 038	1 067
2-sphere	11 095	1 870	11 978	2 564
3-sphere	10 861	3 114	11 333	3 817
4-sphere	13 016	5 884	15 843	7 672
O3	12 740	8 692	12 529	9 633
O4	11 561	6 923	11 187	8 359
O5	13 195	7 351	13 466	11 287
2-torus	12 085	2 137	12 470	2 567
3-torus	14 437	6 707	14 080	8 111
4-torus	16 675	8 575	18 060	9 699
cyclooctane	78 816	11 533	58 328	7 183

Table 2: Comparison of the runtimes of the homology algorithm (with chunk and chunk* preprocessing) and the cohomology algorithm. Each row contains the runtime (ms) for computing a minimal free resolution of $H_1(-)$ of the density-Rips complex of subsamples of size 200. The 6th and 8th column contain the ratio of the respective runtimes.

	chunk	H_1	chunk*	H_1	$\frac{\text{chunk} + H_1}{\text{chunk*} + H_1}$	H^3	$\frac{\text{chunk} + H_1}{H^3}$
S^1	4 804	7 580	5 042	8 035	0.95	3 017	4.10
S^2	3 155	10 526	3 283	10 048	1.03	7 230	1.89
S^3	2 796	11 750	2 840	11 269	1.03	14 463	1.01
S^4	3 138	14 872	2 926	13 474	1.10	26 061	0.69
$O(3)$	2 316	21 932	1 804	22 100	1.01	90 836	0.27
$O(4)$	1 828	25 762	1 543	27 091	0.96	–	–
$O(5)$	2 170	15 398	1 815	14 541	1.07	15 355	1.14
T^2	3 344	11 028	2 638	10 172	1.12	15 672	0.92
T^3	2 930	18 197	2 343	17 074	1.09	23 625	0.89
T^4	2 932	17 193	2 137	16 017	1.11	15 711	1.28
C_8H_{16}	5 456	19 263	4 009	19 910	1.03	12 459	1.98

Table 3: Same as Table 2, but for H_2, and with subsamples of size 100.

	chunk	H_2	chunk*	H_2	$\frac{\text{chunk} + H_2}{\text{chunk*} + H_2}$	H^4	$\frac{\text{chunk} + H_2}{H^4}$
S^1	27 414	2 578	10 034	2 422	2.41	7 569	3.96
S^2	42 190	9 524	15 576	8 983	2.11	9 733	5.31
S^3	47 248	19 103	21 650	18 087	1.67	13 564	4.89
S^4	45 497	19 590	19 696	–	–	8 206	7.93
$O(3)$	45 057	61 030	31 625	58 126	1.18	86 347	1.23
$O(4)$	39 831	66 573	31 397	62 109	1.14	17 077	6.23
$O(5)$	45 235	55 750	28 783	53 817	1.22	10 183	9.92
T^2	44 220	15 815	18 794	14 733	1.79	9 304	6.45
T^3	43 711	25 648	21 984	25 279	1.47	9 615	7.21
T^4	53 143	51 937	28 762	49 007	1.35	14 444	7.27
C_8H_{16}	–	–	28 193	37 339	–	21 910	–

Table 4: Same as Table 3, but run on the machine M.

	chunk	H_2	chunk*	H_2	$\frac{\text{chunk} + H_2}{\text{chunk*} + H_2}$	H^4	$\frac{\text{chunk} + H_2}{H^4}$
S^1	37 289	5 200	12 155	4 638	2.53	8 744	4.86
S^2	52 294	14 242	18 764	13 568	2.06	7 982	8.34
S^3	50 261	23 272	19 211	25 089	1.66	15 998	4.60
S^4	50 206	25 615	21 198	29 454	1.50	8 964	8.46
$O(3)$	55 566	67 668	24 786	66 158	1.36	21 742	5.67
$O(4)$	49 257	50 104	22 577	52 270	1.33	8 980	11.06
$O(5)$	49 861	59 751	25 093	55 153	1.37	11 150	9.83
T^2	44 758	13 240	16 377	13 137	1.97	7 921	7.32
T^3	51 814	44 898	22 597	39 410	1.56	17 216	5.62
T^4	54 126	60 019	25 565	54 947	1.42	16 546	6.90
C_8H_{16}	209 094	47 583	29 800	47 482	3.32	32 365	7.93

Table 5: Same as Table 2, but for H_3, and with subsamples of size 60.

	chunk	H_3	chunk*	H_3	$\frac{\text{chunk} + H_3}{\text{chunk*} + H_3}$	H^5	$\frac{\text{chunk} + H_3}{H^5}$
S^1	42 826	1 993	1 401	2 169	12.55	13 759	3.26
S^2	86 650	2 771	6 365	2 846	9.71	13 373	6.69
S^3	128 907	5 184	9 707	4 272	9.59	12 753	10.51
S^4	123 745	5 981	10 916	5 815	7.75	14 015	9.26
$O(3)$	164 526	20 048	28 119	17 735	4.03	14 266	12.94
$O(4)$	143 490	19 329	28 167	18 563	3.48	13 632	11.94
$O(5)$	106 272	20 293	24 470	19 070	2.91	13 895	9.11
T^2	124 135	4 578	9 806	4 005	9.32	13 209	9.74
T^3	123 169	5 940	11 526	4 886	7.87	13 301	9.71
T^4	171 873	16 470	21 924	15 203	5.07	13 154	14.32
C_8H_{16}	–	–	71 454	11 535	–	14 958	–

Table 6: Same as Table 5, but run on the machine M.

	chunk	H_3	chunk*	H_3	$\frac{\text{chunk} + H_3}{\text{chunk*} + H_3}$	H^5	$\frac{\text{chunk} + H_3}{H^5}$
S^1	52 476	2 731	2 989	2 702	9.70	14 428	3.83
S^2	98 841	3 188	7 815	3 121	9.33	14 703	6.94
S^3	112 448	6 480	9 329	6 241	7.64	11 943	9.96
S^4	121 720	9 062	14 025	9 456	5.57	14 838	8.81
$O(3)$	116 542	13 710	24 206	14 847	3.34	14 621	8.91
$O(4)$	130 011	14 386	26 825	13 728	3.56	14 843	9.73
$O(5)$	118 761	24 942	25 274	21 144	3.10	14 736	9.75
T^2	108 809	5 216	9 573	5 188	7.72	13 102	8.70
T^3	139 316	14 464	22 173	12 603	4.42	13 040	11.79
T^4	166 287	13 722	20 985	14 603	5.06	12 897	13.96
C_8H_{16}	–	–	116 980	30 016	–	15 421	–

Table 7: Runtime comparison of heap and vector representation. The table lists runtimes of the homology algorithm with chunk, chunk*, and the cohomology algorithm, each with the heap and the vector representation, The column "speedup" lists the runtime with the vector representation, divided by the runtime of the heap representation. The runtimes are taken for computing H_1 for subsamples of size 200 of the respective samples.

		chunk+H_1			chunk*+H_1			H^3	
	heaps	vectors	speedup	heaps	vectors	speedup	heaps	vectors	speedup
S^1	12 384	10 912	0.88	13 077	12 551	0.96	3 017	3 746	1.24
S^2	13 681	12 029	0.88	13 331	11 124	0.83	7 230	13 028	1.80
S^3	14 546	11 784	0.81	14 109	11 711	0.83	14 463	30 507	2.11
S^4	18 010	13 720	0.76	16 400	12 636	0.77	26 061	22 234	0.85
$O(3)$	24 248	18 193	0.75	23 904	17 924	0.75	90 836	39 414	0.43
$O(4)$	27 590	24 795	0.90	28 634	22 854	0.80	–	53 629	–
$O(5)$	17 568	14 153	0.81	16 356	12 843	0.79	15 355	14 342	0.93
T^2	14 372	11 298	0.79	12 810	10 001	0.78	15 672	9 839	0.63
T^3	21 127	15 751	0.75	19 417	15 442	0.80	23 625	31 116	1.32
T^4	20 125	15 790	0.78	18 154	15 081	0.83	15 711	13 794	0.88
C_8H_{16}	24 719	14 119	0.57	23 919	15 194	0.64	12 459	16 058	1.29

Table 8: Same as Table 7, but for computing H_2, and with subsamples of size 100.

		chunk+H_2			chunk*+H_2			H^4	
	heaps	vectors	speedup	heaps	vectors	speedup	heaps	vectors	speedup
S^1	29 992	30 115	1.00	12 456	132 855	10.67	7 569	7 566	1.00
S^2	51 714	42 938	0.83	24 559	118 823	4.84	9 733	20 546	2.11
S^3	66 351	46 422	0.70	39 737	115 088	2.90	13 564	53 948	3.98
S^4	65 087	51 741	0.79	–	114 482	–	8 206	7 274	0.89
$O(3)$	106 087	76 740	0.72	89 751	148 436	1.65	86 347	185 573	2.15
$O(4)$	106 404	64 582	0.61	93 506	156 135	1.67	17 077	28 919	1.69
$O(5)$	100 985	58 988	0.58	82 600	161 102	1.95	10 183	13 966	1.37
T^2	60 035	51 770	0.86	33 527	104 008	3.10	9 304	17 221	1.85
T^3	69 359	50 702	0.73	47 263	135 179	2.86	9 615	11 014	1.15
T^4	105 080	65 117	0.62	77 769	161 908	2.08	14 444	30 770	2.13
C_8H_{16}	–	171 381	–	65 532	–	–	21 910	170 360	7.78

Table 9: Same as Table 8, but run on machine M.

	chunk+H_2			chunk*+H_2			H^4		
	heaps	vectors	speedup	heaps	vectors	speedup	heaps	vectors	speedup
S^1	42 489	51 606	1.21	16 793	183 232	10.91	8 744	26 296	3.01
S^2	66 536	80 296	1.21	32 332	205 232	6.35	7 982	8 232	1.03
S^3	73 533	78 732	1.07	44 300	173 794	3.92	15 998	105 595	6.60
S^4	75 821	81 801	1.08	50 652	193 228	3.81	8 964	12 446	1.39
$O(3)$	123 234	121 360	0.98	90 944	257 593	2.83	21 742	80 548	3.70
$O(4)$	99 361	93 838	0.94	74 847	217 728	2.91	8 980	10 427	1.16
$O(5)$	109 612	107 905	0.98	80 246	248 209	3.09	11 150	12 253	1.10
T^2	57 998	72 185	1.24	29 514	192 961	6.54	7 921	7 036	0.89
T^3	96 712	99 274	1.03	62 007	196 956	3.18	17 216	91 363	5.31
T^4	114 145	121 173	1.06	80 512	240 337	2.99	16 546	37 658	2.28
C_8H_{16}	256 677	–	–	77 282	–	–	32 365	–	–

Table 10: Same as Table 7, but for computing H_3, and with subsamples of size 50.

	chunk+H_3			chunk*+H_3			H^5		
	heaps	vectors	speedup	heaps	vectors	speedup	heaps	vectors	speedup
S^1	18 728	12 189	0.65	1 476	5 223	3.54	4 821	4 346	0.90
S^2	48 394	42 502	0.88	2 896	20 528	7.09	4 769	4 273	0.90
S^3	45 226	35 858	0.79	3 765	18 240	4.84	4 643	4 277	0.92
S^4	44 188	41 964	0.95	4 940	21 607	4.37	5 062	3 924	0.78
$O(3)$	91 427	82 780	0.91	12 176	58 628	4.82	4 708	4 079	0.87
$O(4)$	58 547	53 700	0.92	10 541	54 100	5.13	4 691	4 150	0.88
$O(5)$	32 781	28 006	0.85	8 644	32 805	3.80	5 208	4 662	0.90
T^2	36 027	29 737	0.83	4 000	16 295	4.07	4 655	4 353	0.94
T^3	37 427	33 144	0.89	6 347	34 430	5.42	5 063	3 932	0.78
T^4	54 739	50 096	0.92	11 006	44 921	4.08	4 946	4 289	0.87
C_8H_{16}	281 220	–	–	24 105	164 893	6.84	6 626	4 560	0.69

Table 11: Same as Table 10, but run on machine M.

	chunk+H_3			chunk*+H_3			H^5		
	heaps	vectors	speedup	heaps	vectors	speedup	heaps	vectors	speedup
S^1	23 202	29 425	1.27	1 955	10 586	5.41	4 920	3 798	0.77
S^2	29 830	45 988	1.54	3 003	24 803	8.26	4 998	4 268	0.85
S^3	64 478	97 334	1.51	5 726	39 121	6.83	5 066	4 447	0.88
S^4	58 205	78 114	1.34	7 152	39 748	5.56	5 147	3 762	0.73
$O(3)$	60 287	110 063	1.83	12 881	104 865	8.14	5 142	4 064	0.79
$O(4)$	44 307	63 782	1.44	11 668	94 886	8.13	5 072	3 814	0.75
$O(5)$	43 197	65 849	1.52	8 908	81 485	9.15	4 967	3 923	0.79
T^2	44 114	70 518	1.60	3 331	23 829	7.15	4 137	3 796	0.92
T^3	37 744	61 395	1.63	9 031	66 894	7.41	4 449	3 955	0.89
T^4	62 315	101 003	1.62	12 963	68 146	5.26	4 133	3 971	0.96
C_8H_{16}	–	–	–	32 654	166 399	5.10	5 223	4 572	0.88

Table 12: Runtime and effect of chunk and chunk* preprocessing. The reported runtime of chunk (resp. chunk*) preprocessing is the time needed to minimize (Algorithm 3) the 3th boundary (resp. coboundary) matrix. The two other groups show the runtime of the homology and the cohomology algorithm, each without preprocessing, after chunk and after chunk* preprocessing. The run times were measured for subsamples of size 100.

				H_2			H^4	
	chunk	chunk*	w/o	w. chunk	w. chunk*	w/o	w. chunk	w. chunk*
S^1	27 414	10 034	118 209	2 578	2 422	7 569	3 636	3 371
S^2	42 190	15 576	178 587	9 524	8 983	9 733	8 419	7 306
S^3	47 248	21 650	–	19 103	18 087	13 564	28 563	–
S^4	45 497	19 696	–	19 590	–	8 206	7 781	7 534
$O(3)$	45 057	31 625	–	61 030	58 126	86 347	–	–
$O(4)$	39 831	31 397	–	66 573	62 109	17 077	127 683	126 768
$O(5)$	45 235	28 783	–	55 750	53 817	10 183	13 905	12 904
T^2	44 220	18 794	262 684	15 815	14 733	9 304	10 394	9 965
T^3	43 711	21 984	–	25 648	25 279	9 615	12 625	–
T^4	53 143	28 762	–	51 937	49 007	14 444	25 265	–
C_8H_{16}	–	28 193	–	–	37 339	21 910	44 551	42 301

Table 13: Same as Table 12, but for the vector representation.

				H_2			H^4	
	chunk	chunk*	w/o	w. chunk	w. chunk*	w/o	w. chunk	w. chunk*
S^1	27 779	130 466	209 212	2 336	2 389	7 566	3 231	3 247
S^2	36 583	112 898	–	6 355	5 925	20 546	4 944	4 433
S^3	35 817	104 276	–	10 605	10 812	53 948	8 433	7 668
S^4	40 265	102 946	–	11 476	11 536	7 274	4 418	4 423
$O(3)$	38 574	116 482	–	38 166	31 954	185 573	26 989	26 027
$O(4)$	31 466	121 588	–	33 116	34 547	28 919	10 519	10 123
$O(5)$	31 092	132 402	–	27 896	28 700	13 966	6 836	7 533
T^2	42 311	95 259	–	9 459	8 749	17 221	5 341	4 727
T^3	35 576	117 914	–	15 126	17 265	11 014	5 866	6 087
T^4	38 868	137 073	–	26 249	24 835	30 770	10 828	10 055
C_8H_{16}	155 365	–	–	16 016	–	170 360	13 254	–

Table 14: Same as Table 12, but for H_3, and with a subsample size of 60.

				H_3			H^5	
	chunk	chunk*	w/o	w. chunk	w. chunk*	w/o	w. chunk	w. chunk*
S^1	42 826	1 401	–	1 993	2 169	13 759	4 485	4 359
S^2	86 650	6 365	–	2 771	2 846	13 373	4 418	5 026
S^3	128 907	9 707	–	5 184	4 272	12 753	5 170	5 446
S^4	123 745	10 916	–	5 981	5 815	14 015	5 645	5 417
$O(3)$	164 526	28 119	–	20 048	17 735	14 266	8 918	8 308
$O(4)$	143 490	28 167	–	19 329	18 563	13 632	8 778	7 747
$O(5)$	106 272	24 470	–	20 293	19 070	13 895	8 862	7 602
T^2	124 135	9 806	–	4 578	4 005	13 209	5 978	5 265
T^3	123 169	11 526	–	5 940	4 886	13 301	6 045	6 007
T^4	171 873	21 924	–	16 470	15 203	13 154	7 692	7 053
C_8H_{16}	–	71 454	–	–	11 535	14 958	–	6 564

Table 15: Same as Table 14, but for the vector representation.

				H_3			H^5	
	chunk	chunk*	w/o	w. chunk	w. chunk*	w/o	w. chunk	w. chunk*
S^1	41 923	4 406	–	2 067	2 389	13 672	4 462	4 743
S^2	78 721	108 905	–	2 573	2 401	11 646	4 614	4 548
S^3	104 420	98 118	–	3 635	3 727	11 515	4 363	4 405
S^4	125 403	147 877	–	4 471	4 218	12 496	4 472	4 930
$O(3)$	150 888	–	–	10 088	–	13 287	5 429	–
$O(4)$	130 892	–	–	9 847	–	13 169	4 988	–
$O(5)$	97 518	–	–	11 479	–	13 625	5 656	–
T^2	120 352	130 758	–	3 361	3 371	12 234	4 751	4 822
T^3	112 293	170 990	–	3 990	4 229	12 795	5 023	4 595
T^4	161 083	261 555	–	9 196	–	11 792	5 249	4 890
C_8H_{16}	–	–	–	–	–	14 347	–	–

Table 16: Run times of different variants of the cohomology algorithm. Each row contains the runtime for computing $H^4(N^\bullet(-))$ ($\hat{=}$ $H_2(K_*)$) of the density-Rips complex of a subsample of size 100. The first column contains the runtime of the cohomology algorithm as used in all other tables, including sparsification (Algorithm 22) between the two loops of Algorithm 10, and coning off the input complex (Section 5.1.4 with respect to the density parameter of the input. The column "no sparsification" contains runtimes without this sparsification step. The column "no clearing" contains runtimes without the clearing step in line (a) of Algorithm 11. The columns "simplicial cone" contain runtimes when using the simplicial cone to make the input eventually acyclic (Section 3.7.1). The column "no cone" contains run times without any preprocessing that makes the input eventually acyclic. Note that in this case, one cannot infer H_2 from $H^4(N^\bullet$. The last two columns show the runtime of the cohomology algorithm when combined with chunk and chunk* preprocessing.

	normal	no sparsification	no clearing	simp cone	no cone	chunk	chunk*
S^1	7 569	5 461	7 548	8 162	8 174	33 870	13 521
S^2	9 733	6 909	8 104	10 290	8 058	49 117	22 266
S^3	13 564	17 332	12 212	14 630	12 173	73 401	–
S^4	8 206	14 786	6 845	8 358	7 732	55 021	27 297
$O(3)$	86 347	137 011	79 025	59 528	39 186	–	–
$O(4)$	17 077	69 926	16 916	24 738	17 097	169 454	155 382
$O(5)$	10 183	60 882	9 080	11 394	12 076	56 908	38 418
T^2	9 304	10 797	8 891	9 807	9 275	59 798	31 761
T^3	9 615	22 271	9 148	9 789	9 730	54 777	–
T^4	14 444	37 775	15 921	15 041	12 647	80 866	–
C_8H_{16}	21 910	28 445	24 618	23 587	18 000	237 735	73 353

Table 17: Same as Table 16, but for $H^5(N^\bullet)$ ($\hat{=}\ H_3(K_*)$), and with a subsample size of 60.

	normal	no sparsification	no clearing	simp cone	no cone	chunk	chunk*
S^1	13 759	7 043	6 432	14 885	12 717	47 747	5 704
S^2	13 373	7 897	6 697	14 597	13 219	92 410	10 999
S^3	12 753	8 067	8 443	15 179	12 529	135 566	13 571
S^4	14 015	8 744	8 492	14 440	14 263	135 491	16 280
$O(3)$	14 266	20 644	21 233	15 004	14 865	166 358	37 476
$O(4)$	13 632	15 970	18 958	14 940	13 220	146 703	34 616
$O(5)$	13 895	22 545	24 174	15 365	13 725	113 428	31 973
T^2	13 209	8 757	8 524	13 754	13 531	136 668	15 008
T^3	13 301	9 081	8 790	14 310	13 357	122 560	17 718
T^4	13 154	14 965	15 977	16 297	13 212	182 112	27 347
C_8H_{16}	14 958	13 784	13 958	17 776	15 317	–	81 506

Table 18: Same as Table 16, but with the vector representation.

	normal	no sparsification	no clearing	simp cone	no cone	chunk	chunk*
S^1	7 566	5 538	7 286	8 897	11 102	32 189	139 049
S^2	20 546	34 070	19 168	23 135	18 363	41 672	109 123
S^3	53 948	136 226	52 552	57 022	56 479	47 938	110 360
S^4	7 274	89 539	7 462	8 776	7 981	43 085	108 275
$O(3)$	185 573	–	181 390	242 204	203 544	61 565	135 672
$O(4)$	28 919	–	28 862	36 612	29 574	44 218	128 983
$O(5)$	13 966	–	13 988	15 431	15 614	43 037	147 286
T^2	17 221	52 698	16 407	18 989	16 095	45 278	93 917
T^3	11 014	140 227	11 185	11 984	11 950	40 512	123 177
T^4	30 770	191 980	27 653	27 534	25 933	55 042	138 676
C_8H_{16}	170 360	262 912	156 842	194 073	133 231	185 177	–

Table 19: Same as Table 17, but with the vector representation.

	normal	no sparsification	no clearing	simp cone	no cone	chunk	chunk*
S^1	4 346	2 193	2 038	5 054	3 599	12 840	5 694
S^2	4 273	3 133	2 973	5 655	3 968	42 198	19 179
S^3	4 277	2 534	2 490	5 380	4 864	37 125	17 032
S^4	3 924	4 225	3 381	5 732	4 194	37 132	20 253
$O(3)$	4 079	12 985	12 336	5 013	4 508	76 542	52 926
$O(4)$	4 150	9 209	9 164	5 312	3 942	46 173	56 870
$O(5)$	4 662	9 261	9 214	5 267	4 830	25 404	32 764
T^2	4 353	6 916	7 221	4 850	4 307	28 604	16 321
T^3	3 932	6 196	5 534	4 972	4 554	30 047	35 352
T^4	4 289	8 042	7 636	5 262	4 362	45 320	39 200
C_8H_{16}	4 560	20 191	19 873	5 517	4 985	–	169 944

Table 20: Run times for the steps of the cohomology algorithm (Algorithm 11). The columns mean, from left to right, the first and second reduction for-loop in Algorithm 10, the sparsification step (Algorithm 22) between the two loops, and the call to `Minimize`() and and `KerAndMgsWithKer`() in Algorithm 11. The numbers are from computing H^4 with a subsample size of 100.

	first	second	sparsify	minimize	kernel	res
S^1	45	122	3 071	724	1 716	7 569
S^2	62	424	3 710	1 155	2 189	9 733
S^3	73	503	3 431	2 991	3 230	13 564
S^4	64	92	3 513	786	1 793	8 206
$O(3)$	48	17 971	3 661	41 758	14 611	86 347
$O(4)$	79	1 374	3 162	3 800	5 207	17 077
$O(5)$	45	404	3 265	1 779	2 514	10 183
T^2	63	271	3 624	1 149	2 032	9 304
T^3	63	209	3 679	1 298	2 083	9 615
T^4	67	370	3 968	2 767	4 048	14 444
C_8H_{16}	99	1 227	4 694	6 233	5 319	21 910

Table 21: Same as Table 20, but for the vector representation.

	first	second	sparsify	minimize	kernel	res
S^1	54	489	3 020	668	1 564	7 566
S^2	53	8 351	3 131	4 784	2 097	20 546
S^3	53	16 069	3 153	28 474	2 986	53 948
S^4	51	173	2 801	853	1 597	7 274
$O(3)$	73	34 128	3 552	129 835	10 389	185 573
$O(4)$	61	4 745	3 424	12 840	4 445	28 919
$O(5)$	60	1 896	3 265	4 374	2 163	13 966
T^2	54	5 282	3 229	4 360	2 049	17 221
T^3	56	1 195	3 076	2 612	1 954	11 014
T^4	57	4 535	3 301	15 659	3 989	30 770
C_8H_{16}	130	36 454	4 069	120 199	5 142	170 360

Table 22: Same as Table 20, but without the sparsification step.

	first	second	sparsify	minimize	kernel	res
S^1	103	156	–	595	2 294	5 461
S^2	60	471	–	1 547	2 396	6 909
S^3	60	689	–	6 895	4 626	17 332
S^4	57	161	–	4 954	4 966	14 786
$O(3)$	62	27 688	–	72 867	23 456	137 011
$O(4)$	51	4 533	–	27 675	25 453	69 926
$O(5)$	61	763	–	18 851	29 556	60 882
T^2	54	297	–	3 297	3 543	10 797
T^3	61	252	–	7 984	7 408	22 271
T^4	73	581	–	12 601	15 574	37 775
C_8H_{16}	90	1 554	–	12 229	8 231	28 445

Table 23: Same as Table 22, but without the sparsification step.

	first	second	sparsify	minimize	kernel	res
S^1	64	1 400	–	571	1 623	5 538
S^2	52	17 181	–	12 120	2 273	34 070
S^3	62	19 626	–	106 965	4 686	136 226
S^4	61	1 212	–	77 378	5 616	89 539
$O(3)$	70	46 602	–	–	–	–
$O(4)$	69	11 093	–	–	–	–
$O(5)$	72	3 848	–	255 013	–	–
T^2	54	10 602	–	34 891	3 404	52 698
T^3	72	2 321	–	124 181	6 764	140 227
T^4	73	11 325	–	158 941	12 718	191 980
C_8H_{16}	138	47 227	–	202 075	7 261	262 912

Table 24: Same as Table 20, but for H^5 and a subsample size of 60.

	first	second	sparsify	minimize	kernel	res
S^1	362	254	6 669	1 154	2 295	13 759
S^2	345	209	7 011	1 018	2 095	13 373
S^3	326	203	6 582	1 019	2 088	12 753
S^4	342	228	6 855	1 183	2 319	14 015
$O(3)$	425	300	6 992	1 354	2 206	14 266
$O(4)$	382	292	6 859	1 214	2 106	13 632
$O(5)$	387	247	6 867	1 231	2 359	13 895
T^2	334	240	6 599	1 101	2 213	13 209
T^3	372	248	6 669	1 223	2 018	13 301
T^4	348	211	6 496	1 040	2 235	13 154
C_8H_{16}	678	232	8 400	1 004	1 936	14 958

Table 25: Same as Table 24, but for the vector representation.

	first	second	sparsify	minimize	kernel	res
S^1	217	155	6 810	914	2 363	13 672
S^2	179	132	5 748	846	2 101	11 646
S^3	186	156	5 811	878	1 961	11 515
S^4	314	144	5 939	929	2 227	12 496
$O(3)$	197	233	6 305	1 249	2 271	13 287
$O(4)$	221	173	6 632	1 055	2 279	13 169
$O(5)$	218	181	6 888	1 073	2 441	13 625
T^2	191	160	6 116	885	2 139	12 234
T^3	194	144	6 410	1 062	2 211	12 795
T^4	192	152	5 842	874	2 078	11 792
C_8H_{16}	509	160	7 744	794	2 135	14 347

Table 26: Same as Table 24, but without the sparsification step.

	first	second	sparsify	minimize	kernel	res
S^1	312	192	–	1 129	2 495	7 043
S^2	325	218	–	1 275	2 716	7 897
S^3	314	200	–	1 837	2 405	8 067
S^4	326	254	–	1 818	2 802	8 744
$O(3)$	375	335	–	8 036	5 387	20 644
$O(4)$	329	326	–	4 975	4 789	15 970
$O(5)$	353	383	–	8 071	6 535	22 545
T^2	317	229	–	2 098	2 569	8 757
T^3	430	293	–	1 937	2 694	9 081
T^4	325	261	–	4 603	4 471	14 965
C_8H_{16}	688	275	–	5 995	2 725	13 784

Table 27: Same as Table 26, but for the vector representation.

	first	second	sparsify	minimize	kernel	res
S^1	177	149	–	882	2 113	6 154
S^2	161	163	–	766	2 138	5 986
S^3	202	203	–	8 376	2 414	14 307
S^4	221	231	–	8 982	2 508	15 218
$O(3)$	205	667	–	203 171	4 549	214 265
$O(4)$	206	251	–	54 535	3 989	63 785
$O(5)$	232	320	–	193 038	5 554	205 314
T^2	254	371	–	31 209	2 839	38 330
T^3	194	212	–	13 717	2 440	19 907
T^4	204	272	–	47 274	3 940	56 157
C_8H_{16}	525	151	–	102 046	2 768	109 022