Efficient Two-Parameter Persistence Computation via Cohomology

Ulrich Bauer
Department of Mathematics, TUM School of Computation, Information and Technology, and Munich Data Science Institute, Technical University of Munich, Germany

Fabian Lenzen
Department of Mathematics, TUM School of Computation, Information and Technology, Technical University of Munich, Germany

Michael Lesnick
Department of Mathematics, SUNY Albany, NY, USA

Abstract

Clearing is a simple but effective optimization for the standard algorithm of persistent homology (PH), which dramatically improves the speed and scalability of PH computations for Vietoris–Rips filtrations. Due to the quick growth of the boundary matrices of a Vietoris–Rips filtration with increasing dimension, clearing is only effective when used in conjunction with a dual (cohomological) variant of the standard algorithm. This approach has not previously been applied successfully to the computation of two-parameter PH.

We introduce a cohomological algorithm for computing minimal free resolutions of two-parameter PH that allows for clearing. To derive our algorithm, we extend the duality principles which underlie the one-parameter approach to the two-parameter setting. We provide an implementation and report experimental run times for function-Rips filtrations. Our method is faster than the current state-of-the-art by a factor of up to 20.

2012 ACM Subject Classification Mathematics of computing → Algebraic topology; Theory of computation → Computational geometry

Keywords and phrases Persistent homology, persistent cohomology, two-parameter persistence, clearing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2023.15

Related Version *Full Version*: `https://arxiv.org/abs/2303.11193` [8]

Supplementary Material *Software (Source Code)*: `https://gitlab.com/flenzen/2-parameter-persistent-cohomology` [33]
archived at `swh:1:dir:7ef1c55bee7980f9a68507241eeb461e0af7d99e`

Funding *Ulrich Bauer*: Supported by the German Research Foundation (DFG) through the Collaborative Research Center SFB/TRR 109 *Discretization in Geometry and Dynamics* – 195170736.

1 Introduction

Motivation. Persistent homology [19, 50, 37] analyzes how the homology of a filtered topological space changes as the filtration parameter increases. By assigning filtered spaces (e.g., Vietoris–Rips filtrations) to data sets, it provides simple signatures of the data called *barcodes*, which encode multi-scale information about the shape of the data. Thanks to recent advances in PH computation and software [4, 49, 1, 39, 45, 26, 27], PH has become popular for practical data applications [25]. A well-known stability result [16, 9, 12] guarantees that small perturbations (in the Gromov–Hausdorff distance) of the input data lead to small perturbations (in the bottleneck distance) of the barcodes of the PH of Vietoris–Rips filtrations. However, Vietoris–Rips PH is notoriously unstable to outliers. Besides other strategies [12, Section 1.7], a commonly proposed remedy for this is the introduction of a

second filtration parameter controlling the local density of the point cloud, which leads to the notions of *two-* and *multi-parameter* PH [14, 43, 17, 34, 41, 13]. A central computational problem of two-parameter PH, somewhat akin to the computation of a barcode, is the computation of a *minimal free presentation* or a *minimal free resolution* (MFR) of the PH module. While a resolution contains more information than a presentation, the underlying algorithmic problems are essentially the same, and we focus on the computation of a MFR in this work. Such a resolution is often quite small in practice [23], and computing it is a natural first step in computing invariants or metrics in the multi-parameter setting [13].

Computing a MFR of two-parameter PH is more involved than in the one parameter case. The problem can be solved by classical Gröbner basis algorithms, which work in much greater generality but do not scale well enough for practical TDA applications [35]. Recently, a specialized algorithm was introduced [35, 32], which is far more efficient than the Gröbner basis approach, both in theory and in practice. This has substantially lowered the barrier to practical data analysis with two-parameter persistence [30]. For recent applications, see, e.g., [11, 46].

Nevertheless, existing software for two-parameter PH is much slower and less scalable than one-parameter PH implementations such as [4, 1, 39, 49]. The approach of [35, 32] boils down to a matrix reduction scheme similar to the standard algorithm of one-parameter PH [50], and has the same asymptotic run time, cubic in the size of the complex. However, modern one-parameter PH algorithms incorporate several critical optimizations. In particular, it is known that for Vietoris–Rips filtrations, *clearing* [5, 15] leads to major performance gains when combined with a dual (cohomological) variant of the standard persistence algorithm [18, 5]. All state-of-the-art software for computing Vietoris–Rips PH employ this strategy. In two parameters, however, working with persistent cohomology (PC) is more challenging, essentially because, in contrast to one parameter, relative simplicial cochains of filtered complexes do not form free modules.

Contributions. In order to compute MFRs of PH of function-Rips bifiltrations more efficiently, we introduce a cohomological variant of the algorithm of [35, 32], which we outline now.

Let K_* be a finite simplicial $\mathbf{Z}^n$-filtration with $K = \bigcup_{z\in\mathbf{Z}^n} K_z$. Assume that K_* is one-critical, i.e., the set $\{z \,|\, \sigma \in K_z\}$ has a unique minimal element $g(\sigma)$ for every $\sigma \in K$. We define a certain cochain complex $N^\bullet(K_*)$ of free persistence modules. In this paper, $H_\bullet(K)$ always denotes the reduced simplicial homology of K. If $H_d(K_z) = 0$ for all d but finitely many indices $z \in \mathbf{Z}^n$ (which can easily be ensured by adding additional simplices to K_*), then $H^{d+n}(N^\bullet(K_*))$ is isomorphic to the dual module of $H_d(K_*)$ for all d (see Proposition 6). This can be seen as a generalization of a corresponding statement for one-parameter persistence, in which case $N^\bullet(K_*)$ equals the relative cochain complex $C^\bullet(K, K_*)$ [18, Theorem 2.4]; see also [10].

Given a (minimal) free resolution $F_\bullet$ of an n-parameter persistence module M and a choice of basis for each module of $F_\bullet$, we show that the matrices representing $F_\bullet$ also represent a (minimal) injective resolution of M; see Theorem 10. In particular, this allows us to easily convert a (minimal) free resolution of a module (e.g., $H^{d+n}(N^\bullet(K_*))$) to a (minimal) free resolution of its dual (i.e., $H_d(K_*)$); see Corollary 14.

For $n = 2$, we propose a method to compute a MFR of $H^{d+2}(N^\bullet(K_*))$ (and thus $H_d(K_*)$) solely from the coboundary map $\delta^{d+1}\colon N^d(K_*) \to N^{d+1}(K_*)$; see Section 3.4. At the core of this method is an algorithm for the following problem: given a morphism $f\colon F \to F'$ of free persistence modules and a basis of the vector space $\operatorname{colim}\operatorname{im} f$, compute a basis of the free persistence module $\ker f$; see Theorem 19. The algorithm is compatible with the clearing optimization, which improves its performance considerably.

We have implemented our approach [33] and report timing results from computational experiments with function-Rips bifiltrations. On most instances considered, our approach is significantly faster than the approach [23] used in `mpfree`, and on certain instances, our implementation is able to outperform the approach of by a factor of up to 20.

A number of recent methods for computation of multiparameter persistence focus on decreasing the size of the input complex without changing its homology [22, 42, 23, 2]. These methods can be used as a preprocessing step to the computation of a minimal free resolution. In our computational experiments, we explore the effect of the chunk preprocessing method of [23] on the efficiency of our method. We find that in our experiments, our method generally performs better without this preprocessing. In contrast, we observe that the preprocessing is very helpful for the the approach of [35, 32], as previously reported [23]. We also observe that applying the chunk algorithm on cochain complexes instead of chain complexes may significantly increase the performance even for homology computation.

2 Background

2.1 Persistence modules

Let k be a field, let $n \in \mathbf{N}$, let **vec** denote the category of finite dimensional k-vector spaces, and consider $\mathbf{Z}^n$ as a poset with the usual product partial order. A (pointwise finite dimensional) *$\mathbf{Z}^n$-persistence module*, also called an *n-parameter persistence module*, is a functor $M\colon \mathbf{Z}^n \to \mathbf{vec}$. The maps $M_{z\leq z'}\colon M_z \to M_{z'}$ are called the *structure maps* of M. If $m \in M_z$, we call z the *grade* of m, denoted by $g(m)$. The *total dimension* of M is $\sum_{z\in\mathbf{Z}^n} \dim M_z$. We write **$\mathbf{Z}^n$-pers** for the abelian category of pointwise finite dimensional $\mathbf{Z}^n$-persistence modules. Its morphisms are natural transformations. **$\mathbf{Z}^n$-pers** is equivalent to a full subcategory of the category of multigraded modules over the ring $k[x_1, \dots, x_n]$; see [14]. The algebra $k[x_1, \dots, x_n]$ is not a principal ideal domain unless $n = 1$; therefore, $\mathbf{Z}^n$-persistence modules cannot be described by a barcode for $n > 1$.

Let $V^* = \mathrm{Hom}_k(V, k)$ denote the dual of a k-vector space V. The *dual* of a $\mathbf{Z}^n$-persistence module M is the $\mathbf{Z}^n$-persistence module M^* with $(M^*)_z = (M_{-z})^*$ and $(M^*)_{z\leq z'} = (M_{-z'\leq -z})^*$. An object M of an abelian category $\mathcal{C}$ is *projective* (respectively *injective*) if the functor $\mathrm{Hom}_{\mathcal{C}}(M, -)$ (respectively $\mathrm{Hom}_{\mathcal{C}}(-, M)$) is exact. The duality $M \mapsto M^*$ is an exact contravariant equivalence of categories and thus maps projective to injective modules and vice versa.

For $z \in \mathbf{Z}^n$, let $F(z)$ be the module with $F(z)_w = \begin{cases} k \text{ if } z \leq w, \\ 0 \text{ otherwise,} \end{cases}$ and $F(z)_{w\leq w'} = \mathrm{id}_k$ if $z \leq w \leq w'$. A module F is *free* if there are elements $(z_i)_{i\in I} \subseteq \mathbf{Z}^n$, for some indexing set I, such that there is an isomorphism $b\colon \bigoplus_{i\in I} F(z_i) \to F$. Every finitely generated projective persistence module is free [40, 44, 28]. Let e_i denote the element $1 \in F(z_i)_{z_i}$ of the component of $\bigoplus_{i\in I} F(z_i)$ indexed by i. The set $\{b(e_i) \mid i \in I\}$ is a *basis* of F. The multiset $\mathrm{rk}\, F := \{z_i \mid i \in I\}$ is uniquely determined by F and called its *(graded) rank*. A module M is *finitely generated* if there is a pointwise surjection $F \to M$ from a free module F of finite rank; the image of a basis of F under such a map is called a *generating system* of M.

A *graded matrix* M is a matrix with entries $M_{ij} \in k$ whose rows and columns are decorated with row grades rg^M_* and column grades cg^M_*. The *graded transpose* of a graded $m \times n$-matrix M is the graded $n \times m$-matrix M^T with entries $(M^T)_{ij} = M_{m+1-j,n+1-i}$, row grades $\mathrm{rg}^{M^T}_i = -\mathrm{cg}^M_{n+1-i}$ and column grades $\mathrm{cg}^{M^T}_j = -\mathrm{rg}^M_{m+1-i}$. A morphism $f\colon F \to F'$ of finite rank free modules F and F' with respective bases $b_1, \dots, b_n$ and $b'_1, \dots, b'_m$ is uniquely represented by a graded $m \times n$-matrix M with $\mathrm{cg}^M_j = g(b_j)$, $\mathrm{rg}^M_i = g(b'_i)$, and entries M_{ij} such that $f(b_j) = \sum_i M_{ij} F'_{g(b'_i)\leq g(b_j)}(b'_i)$ for all j.

▶ **Lemma 1.** *A graded matrix M represents a morphism of finite rank free modules iff $M_{ij} = 0$ whenever $\mathrm{rg}_i^M \not\leq \mathrm{cg}_j^M$.*

Proof. This follows from the fact that $\mathrm{Hom}(F(z), F(z')) = \begin{cases} k \text{ if } z \geq z', \\ 0 \text{ otherwise.} \end{cases}$ ◀

If bases of free modules F, F' are fixed, we identify a morphism $F \to F'$ with the graded matrix representing it. A *free resolution* (resp., *injective* resolution) of a module M is a chain complex $F_\bullet \colon \cdots \to F_1 \to F_0$ of free modules (resp., cochain complex $I^\bullet \colon I^0 \to I^1 \to \cdots$ of injective modules) concentrated in non-negative degrees that is quasi-isomorphic to M. A *(homological) d-ball* is an acyclic chain complex of the form $\cdots \to 0 \to F(z) \xrightarrow{\mathrm{id}} F(z) \to 0 \to \cdots$ for some $z \in \mathbf{Z}^n$, concentrated in degrees $d, d-1$. A free resolution, and, more generally, a chain complex of free modules, is called *minimal* if it contains no direct summand isomorphic to a ball. An injective resolution is *minimal* if its dual is minimal. A morphism $F_1 \to F_0$ of free modules is called a *(minimal) free presentation* of a module M if it extends to a (minimal) free resolution of M.

▶ **Theorem 2** (see [21, Theorem 20.2], [38, Theorem 7.5])**.** *Every finitely generated module has a* MFR*. Every free resolution is isomorphic to the direct sum of a* MFR *with a direct sum of homological balls. In particular, a* MFR *is unique up to isomorphism of chain complexes.*

Thus, letting $F_\bullet$ be a MFR of a finitely generated persistence module M, the graded ranks $\beta_q(M) \coloneqq \mathrm{rk}\, F_q$, called the *graded Betti numbers* of M, are independent of the choice of $F_\bullet$.

▶ **Theorem 3** (Hilbert's Syzygy theorem [38, Theorem 15.2], [21, Corollary 19.7])**.** *Every $\mathbf{Z}^n$-persistence module has a* MFR *of length at most n.*

2.2 Filtrations

For P any poset, a (simplicial) *P-filtration* is a functor K_* from $\mathbf{Z}^n$ to the category of simplicial complexes such that the simplicial maps $K_{z \leq z'}$ are inclusions. We write $K = \bigcup_{z \in \mathbf{Z}^n} K_z$.

▶ **Example 4.** Let S be a metric space and let $\operatorname{diam}\sigma = \max_{s,t \in \sigma} d(s,t)$ for every finite, non-empty $\sigma \subseteq S$. The *Vietoris–Rips filtration* $\widehat{VR}_*(S)$ associated to S is the $\mathbf{R}$-filtration given by $\widehat{VR}_r(S) = \{\sigma \subseteq S \mid 0 < |\sigma| < \infty, \operatorname{diam}\sigma \leq r\}$. If S is finite and non-empty, let $r_1 < r_2 < \cdots < r_n$ be the distinct values $\operatorname{diam}\sigma$ can attain for $\sigma \subseteq S$. By setting

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

we obtain a $\mathbf{Z}$-filtration $VR_*(S)$, which we also call a Vietoris–Rips filtration.

If K_* is a $\mathbf{Z}^n$-filtration, its absolute and relative simplicial chains $C_\bullet(K_*)$ and $C_\bullet(K, K_*)$ (with coefficients in k) form chain complexes of $\mathbf{Z}^n$-persistence modules, and the respective cycles $Z_d(-)$, boundaries $B_d(-)$ and homology $H_d(-)$ are $\mathbf{Z}^n$-persistence modules for all d. The dual cochain complex $C^\bullet \coloneqq (C_\bullet)^*$ of a chain complex $C_\bullet$ has components $C^d = (C_d)^*$. Its cocycles $Z^d(C^\bullet)$, coboundaries $B^d(C^\bullet)$ and cohomology $H^d(C^\bullet)$ are $\mathbf{Z}^n$-persistence modules for all d. Because $(-)^*$ is exact, there is a natural isomorphism $H^d(C^\bullet) \to H_d(C_\bullet)^*$ for all d. Our indexing convention is that a chain complex $C_\bullet$ has the boundary morphisms $\partial_d \colon C_d \to C_{d-1}$ and a cochain complex $C^\bullet$ has coboundary morphisms $\delta^d = (\partial_d)^* \colon C^{d-1} \to C^d$. This differs from the standard convention, but is chosen such that $C^\bullet = (C_\bullet)^*$ has $\delta^d = (\partial_d)^*$.

A $\mathbf{Z}^n$-filtration K_* is *one-critical* if for every $\sigma \in K$ the set $\{z \in \mathbf{Z}^n \mid \sigma \in K_z\}$ has a unique minimal element $g(\sigma)$, called the *grade* of σ. In this case, $C_\bullet(K_*) = \bigoplus_{\sigma \in K_*} F(g(\sigma))$ is free, with a basis $\{e_\sigma \mid \sigma \in K_*\}$ satisfying $g(e_\sigma) = g(\sigma)$. In particular, $\partial_\bullet$ can be represented by a graded matrix $[\partial_\bullet]$.

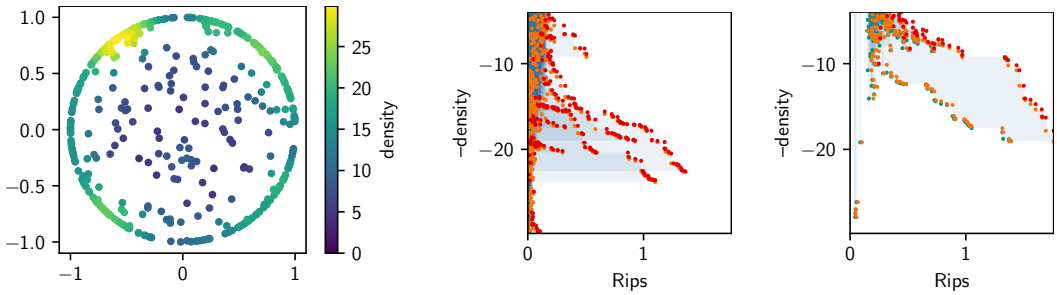


Figure 1 A point set S (left) with $|S| = 400$, with density function $\rho(p) := \sum_{q \in S \setminus \{p\}} \exp\left(-\frac{d(p,q)^2}{2\sigma^2}\right)$ for $\sigma = 0.15$; graded Betti numbers (teal: β_0, red: β_1, orange: β_2) and Hilbert function (shades of blue increasing from dim = 0 to dim ≥ 10) of $H_0(VR^\rho_*(S))$ (middle) and $H_1(VR^\rho_*(S))$ (right).

▶ **Example 5.** The *function-Rips bifiltration* $VR^f_*(S)$ associated to a finite metric space S and a function $f\colon S \to \mathbf{Z}$ is the one-critical $\mathbf{Z}^2$-filtration with $VR^f_{x,y} = \{\sigma \in VR_y(S) \mid \max_{s\in\sigma} f(s) \leq x\}$. Figure 1 illustrates the Hilbert function and graded Betti-numbers of the PH of a function-Rips bifiltration, where the function is a density function.

2.3 One-parameter persistence and clearing

We next turn attention to persistence modules over $\mathbf{Z}$. For $-\infty \leq b_i < d_i \leq \infty$, let $I(b,d)$ be the *interval module* with $I(b,d)_z = \begin{cases} k \text{ if } b \leq z < d, \\ 0 \text{ otherwise} \end{cases}$ and $I(b,d)_{z\leq z'} = \mathrm{id}_k$ if $b \leq z \leq z' < d$. Every pointwise finite-dimensional $\mathbf{Z}$-persistence module is isomorphic to an essentially unique direct sum $\bigoplus_{i\in I} I(b_i, d_i)$ [47, 50]. The collection of the pairs (b_i, d_i) is called the *barcode* of M.

Given a finite $\mathbf{Z}$-filtered complex K_*, one is usually interested in computing the barcode of $H_\bullet(K_*)$. Since $k[x]$ is a principal ideal domain, the submodules $Z_d(K_*), B_d(K_*) \subseteq C_d(K_*)$ are free for all d. The *standard algorithm* [20, §3] computes bases of $Z_d(K_*)$ and $B_d(K_*)$ and thus the barcode of $H_\bullet(K_*)$ by applying an order-respecting Gaussian column reduction scheme to each graded matrix $[\partial_d]$. Each relative cochain module $C^d(K, K_*)$ is also free, so the same algorithm computes the barcode of $H^\bullet(K, K_*)$ from the graded matrices $[\delta^d] = [\partial_d]^T$ representing the coboundary operators δ^d. The barcodes of $H_\bullet(K_*)$ and $H^\bullet(K, K_*)$ determine each other in a simple way, as is seen by considering the long exact sequence of the pair (K, K_*) [18, 10].

It has been observed that for Vietoris–Rips filtrations, computing $H^\bullet(K, K_*)$ instead of $H_\bullet(K_*)$ is far more efficient. This increase in efficiency hinges on the use of the *clearing* optimization scheme [5, 15, 4], which we now explain. The *pivot* of a matrix column is the largest row index of a non-zero entry in that column. The standard algorithm applies left-to-right column additions to bring $[\delta^{d+1}]$ into *reduced* form R^{d+1}, meaning that all columns of R^{d+1} have pairwise distinct pivots. If a column R^d_j is non-zero with pivot i, then $R^{d+1}_i = 0$. Therefore, if R^d is known from previous computations, the reduction of $[\delta^{d+1}]_j$ to zero can be skipped. As the standard algorithm would typically spend most of its run time on the columns of $[\delta^{d+1}]$ that are reduced to zero, skipping most of these accelerates the algorithm considerably.

If the reduced homology $H_d(K)$ is zero for all d, then the long exact sequence of the pair (K, K_*) shows that $H^{d+1}(K, K_*)^* \cong H_d(K_*)$ for all d, so one would expect that they can be computed from the same data. Indeed, one can use clearing to compute $H^{d+1}(K, K_*)$ (respectively $H_d(K_*)$) from δ^{d+1} (respectively ∂_{d+1}) alone; see Algorithm 3 in the full version.

2.4 Computation of 2-parameter persistence

The LW-Algorithm. Assume that $C_\bullet$ is a chain complex of free $\mathbf{Z}^2$-persistence modules of finite rank; e.g., $C_\bullet = C_\bullet(K_*)$ for a one-critical $\mathbf{Z}^2$-filtration K_*, and let D_d be the matrix representing $\partial_d \colon C_d \to C_{d-1}$ for all d. Theorems 2 and 3 imply that the kernel of a morphism of finitely generated free $\mathbf{Z}^2$-modules is free. In particular, $Z_d(C_\bullet)$ is free for all d, so the sequence $0 \to Z_{d+1}(C_\bullet) \xrightarrow{i_{d+1}} C_{d+1} \xrightarrow{p_{d+1}} Z_d(C_\bullet)$ is a free resolution of $H_d(K_*)$. From D_d, the *LW-Algorithm* [35, 32] (see Algorithm 4 in the full version) computes a graded matrix I_d representing $i_d \colon Z_d(C_\bullet) \hookrightarrow C_d$. A variant of that algorithm (see Algorithm 5 in the full version) computes from D_{d+1} a graded matrix $D'_{d+1} \colon C'_{d+1} \to C_d$, whose columns represent a minimal generating system of $B_d(C_\bullet)$, together with a graded matrix I'_{d+1} representing the kernel Z'_{d+1} of the morphism represented by D'_{d+1}. There is a unique graded matrix P'_{d+1} such that $D'_{d+1} = I_d P'_{d+1}$, which can be obtained by Algorithm 9 in the full version. Then I'_{d+1} and P'_{d+1} represent a free resolution

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

of $H_d(C_\bullet)$. To obtain a MFR, it remains to split off summands from (1) that are isomorphic to homological balls. There is an embarrassingly parallel algorithm that computes a minimal chain complex quasi-isomorphic to a given one; see Remark 23 in the full version. In particular, this algorithm can be used to convert a free resolution to a minimal one. It can also be used to split off balls from the input complex $C_\bullet$. This is known as *chunk preprocessing* and typically improves performance of the LW-algorithm by a considerable amount [22, 32, 23].

3 Cohomology computation

Let K_* be a one-critical $\mathbf{Z}^n$-filtration. If $n > 1$, then neither $C^\bullet(K_*)$ nor $C^\bullet(K, K_*)$ are complexes of free modules. Since the LW-algorithm assumes that the input complex is a complex of free modules, the strategy from Section 2.4 cannot be used to compute $H^\bullet(K_*)$ or $H^\bullet(K, K_*)$ directly. Instead, we consider a cochain complex $N^\bullet(K_*)$ that can be used to compute $H_d(K_*)$.

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

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.

Fix a total order on the simplices of K_*, so that the boundary map $\partial_\bullet$ of the chain complex $C_\bullet(K_*) = \bigoplus_{\sigma \in K_*} F(g(\sigma))$ is represented by the graded matrix $[\partial_\bullet]$. Let $\epsilon = (1, \dots, 1) \in \mathbf{Z}^n$. Let

$$N^\bullet(K_*) = \bigoplus_\sigma F(-g(\sigma) + \epsilon)$$

be the cochain complex whose coboundary operator $\delta_N^\bullet$ is represented by $[\partial_\bullet]^T\langle -\epsilon\rangle$ with respect to the standard basis. It follows from Lemma 1 that this is a well-defined cochain complex. The key property of this chain complex is summarized in the following proposition, whose proof is deferred to the next subsection.

▶ **Proposition 6.** *If $H_d(K_*)$ has finite total dimension for all d, then there is a natural isomorphism $H_d(K_*) \cong H^{d+n}(N^\bullet(K_*))^*$ for all d.*

▶ **Corollary 7.** *If $H_\bullet(K_*)$ has finite total dimension and $F_\bullet$ is a free resolution of $H^{d+n}(N^\bullet(K_*))$, then $(F_\bullet)^*$ is an injective resolution of $H_d(K_*)$.*

3.2 The Calabi–Yau-property of persistence modules

Besides proving Proposition 6, we will need to convert the injective resolution of $H_d(K_*)$ from Corollary 7 into a free resolution of $H_d(K_*)$. Both will follow from Theorem 10, which establishes a property of persistence modules known as the *Calabi–Yau property* in some areas of algebra [24]; see [29, Lemma 4.1] for a proof in a more general context. As it turns out, there is a close correspondence between injective and free resolutions that we explore in this section. For $z \in \mathbf{Z}^n$, we define the injective module $I(z) = F(-z)^*$; i.e., $I(z)_w = \begin{cases} k \text{ if } w \le z, \\ 0 \text{ otherwise,} \end{cases}$ and $I(z)_{w \le w'} = \mathrm{id}$ if $w \le w' \le z$.

▶ **Definition 8.** *For persistence modules M, N, let $\mathsf{Hom}(M,N)$ be the persistence module with components $\mathsf{Hom}(M,N)_z = \mathrm{Hom}(M, N\langle z\rangle)$.*

Let $\mathcal{P}_{\mathbf{Z}^n\text{-pers}}$ and $\mathcal{I}_{\mathbf{Z}^n\text{-pers}}$ be the full subcategories of $\mathbf{Z}^n$-**pers** consisting of free and injective modules, respectively.

▶ **Lemma 9** (see [3, Proposition 2.10 in Chapter III])**.** *The* Nakayama functor

$$\nu := \mathsf{Hom}(-, F(0))^* \colon \mathcal{P}_{\mathbf{Z}^n\text{-pers}} \to \mathcal{I}_{\mathbf{Z}^n\text{-pers}}$$

is an equivalence of categories with quasi-inverse $\nu^{-1} = \mathsf{Hom}(I(0)^*, -)$.

One checks that $\nu F(z) = I(z)$ and $\nu^{-1} I(z) = F(z)$. Therefore, $N^\bullet(K_*) = (\nu C_\bullet(K_*)\langle\epsilon\rangle)^*$. For a chain complex $C_\bullet$ and $i \in \mathbf{Z}$, let $C_\bullet[i]$ be the chain complex whose dth module is $(C_\bullet[i])_d = C_{i+d}$. Analogously, for a cochain complex $C^\bullet$, let $C^\bullet[i]$ be the cochain complex with $(C^\bullet[i])^d = C^{i+d}$. Note that $C_\bullet[i]^* = (C_\bullet)^*[i]$.

▶ **Theorem 10.** *If $F_\bullet$ is a complex of free $\mathbf{Z}^n$-persistence modules such that $H_d(F_\bullet)$ has finite total dimension for all d, then $F_\bullet$ and $\nu F_\bullet[n]\langle\epsilon\rangle$ are naturally quasi-isomorphic.*

Proof. For $z \in \mathbf{Z}^n$, we write $z = (z_1, \dots, z_n)$. For $n \in \mathbf{N}$, let $[n] := \{1, \dots, n\}$ and let $\binom{[n]}{k} := \{S \subseteq [n] \mid |S| = k\}$. For $S = \{s_1 < \dots < s_k\} \in \binom{[n]}{k}$ and $z \in \mathbf{Z}^n$, $w \in \mathbf{Z}^k$, we let

$$z|_w^S := (z_1, \dots, z_{s_1 - 1}, w_1, z_{s_1+1}, \dots, z_{s_k-1}, w_k, z_{s_k+1}, \dots, z_n)$$

be the n-tuple obtained from z by replacing the components indexed by S by the entries of w. For any module $M \in \mathbf{Z}^n$-**pers** and $S \subseteq [n]$, we let $\mathrm{Colim}_S M$ be the module with

$$(\mathrm{Colim}_S M)_z = \mathrm{colim}_{w \in \mathbf{Z}^k} M_{z|_w^S}, \qquad (\mathrm{Colim}_S M)_{z \le z'} = \mathrm{colim}_{w \in \mathbf{Z}^k} M_{z|_w^S \le z'|_w^S}.$$

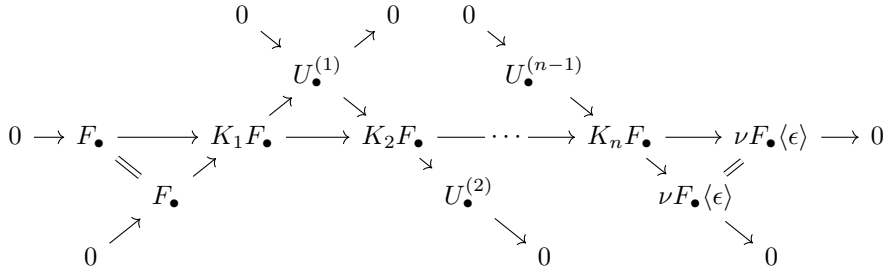
For example, for $n = 3$ we get $(\mathrm{Colim}_{\{1,3\}} M)_{(z_1,z_2,z_3)} = \mathrm{colim}_{(w_1,w_2)\in\mathbf{Z}^2} M_{(w_1,z_2,w_2)}$. The module $\mathrm{Colim}_S M$ is constant along the axes specified by S. In particular, $\mathrm{Colim}\, M = \mathrm{Colim}_{[n]} M$ is the module that is constantly $\mathrm{colim}\, M$. For a module M, we define the modules $K_k M = \bigoplus_{S \in \binom{[n]}{k}} \mathrm{Colim}_S M$ for each k. If $S \subseteq S'$, then there is a canonical morphism $\mathrm{Colim}_S M \to \mathrm{Colim}_{S'} M$. For a free module $F(z)$, these assemble to an exact sequence

$$0 \to F(z) \to K_1 F(z) \to \cdots \to K_n F(z) \to I(z)\langle\epsilon\rangle \to 0, \tag{2}$$

called the *Koszul complex* of $F(z)$. The last morphism is the canonical morphism $K_nF(z) = \operatorname{Colim} F(z) = \operatorname{Lim} I(z)\langle\epsilon\rangle \to I(z)\langle\epsilon\rangle$. Let $F_\bullet$ be a bounded complex of free modules. Using $\nu F(z) = I(z)$, we get an exact sequence

$$K_\bullet\colon \quad 0 \to F_\bullet \to K_1F_\bullet \to \cdots \to K_nF_\bullet \to \nu F_\bullet\langle\epsilon\rangle \to 0 \tag{3}$$

of chain complexes, given by taking a shifted copy of (2) for every $F(z)$ in $F_\bullet$. We unsplice (3) into short exact sequences



(4)

with chain complexes $U_\bullet^{(k)}$ for each k. Each of these short exact sequences gives a triangle in the derived category $\mathcal{D}^{\mathrm{b}}(\mathbf{Z}^n\text{-}\mathbf{pers})$ [48, §10.4.9]. We obtain connecting homomorphisms

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

in $\mathcal{D}^{\mathrm{b}}(\mathbf{Z}^n\text{-}\mathbf{pers})$. These fit into the long exact sequences

$$\begin{aligned}
&\cdots \to H_{d+1}(K_1F_\bullet) \to H_{d+1}(U_\bullet^1) \xrightarrow{\partial^{(1)}} H_d(F_\bullet) \to H_d(K_1F_\bullet) \to \cdots,\\
&\cdots \to H_{d+2}(K_2F_\bullet) \to H_{d+2}(U_\bullet^2) \xrightarrow{\partial^{(2)}} H_{d+1}(U_\bullet^{(1)}) \to H_{d+1}(K_2F_\bullet) \to \cdots,\\
&\qquad\qquad\qquad\qquad\vdots\\
&\cdots \to H_{d+n}(K_nF_\bullet) \to H_{d+n}(\nu F_\bullet\langle\epsilon\rangle) \xrightarrow{\partial^{(n-1)}} H_{d+n-1}(U_\bullet^{(n-1)}) \to H_{d+n-1}(K_nF_\bullet) \to \cdots,
\end{aligned} \tag{5}$$

induced by the short exact sequences (4). Since $H_d(F_\bullet)$ is of finite total dimension for all d, we have $\operatorname{Colim}_S H_d(F_\bullet) = 0$ if $|S| > 0$. The functor Colim_S is exact for all S because it is a directed colimit. In particular, $H_d(K_kF_\bullet) = H_d\big(\bigoplus_{|S|=k} \operatorname{Colim}_S F_\bullet\big) = 0$ for all $k > 0$. Therefore, the long exact sequences (5) show that all connecting homomorphisms $\partial^{(k)}$ are quasi-isomorphisms. Thus, $\partial^{(1)} \circ \cdots \circ \partial^{(n-1)}\colon \nu F_\bullet[n]\langle\epsilon\rangle \longrightarrow F_\bullet$ is a quasi-isomorphism. ◀

▶ **Corollary 11.** *Let $M \in \mathbf{Z}^n$-**pers** be of finite total dimension.*
1. *If $F_\bullet$ is a free resolution of M, then $\nu F_\bullet[n]\langle\epsilon\rangle$ is an injective resolution of M.*
2. *If $I^\bullet$ is an injective resolution of M, then $\nu^{-1}I^\bullet[-n]\langle-\epsilon\rangle$ is a free resolution of M.*

Proof of Proposition 6. With $N^\bullet(K_*) = (\nu C_\bullet(K_*)\langle\epsilon\rangle)^*$, Theorem 10 gives

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

▶ **Lemma 12.** *A graded matrix M represents a morphism $\bigoplus_j I(\mathrm{cg}_j^M) \to \bigoplus_i I(\mathrm{rg}_i^M)$ iff $M_{ij} = 0$ whenever $\mathrm{rg}_i^M \not\leq \mathrm{cg}_j^M$.*

Proof. This follows from $\operatorname{Hom}(I(z), I(z')) = \begin{cases} k \text{ if } z \geq z' \\ 0 \text{ otherwise} \end{cases}$ ◀

In particular, a graded matrix represents a morphism of free modules iff it represents a morphism of injective modules (cf. Lemma 1).

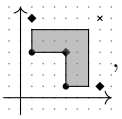
▶ **Lemma 13.** *Let* $f\colon \bigoplus_{j=1}^{n} F(z_j) \to \bigoplus_{i=1}^{m} F(z'_i)$ *be a morphism of free modules represented by the graded matrix* $[f]$*. Then the morphism* $\nu f\colon \bigoplus_{j=1}^{n} I(z_j) \to \bigoplus_{i=1}^{m} I(z'_i)$ *is represented by the same graded matrix* $[\nu f] = [f]$*.*

▶ **Corollary 14.** *For* $M \in \mathbf{Z}^n$**-pers** *of finite total dimension and graded matrices* $U_1, \dots, U_n$*, the following are equivalent:*

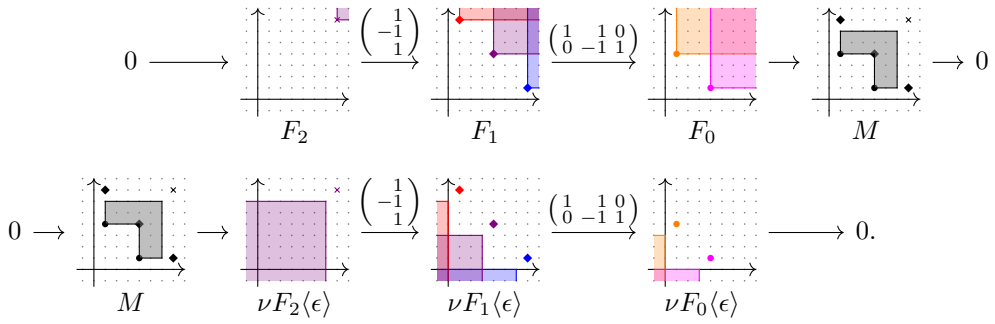
1. $U_1, \dots, U_n$ *represent a free resolution* $\cdots 0 \to F_n \xrightarrow{U_n} \cdots \xrightarrow{U_1} F_0$ *of* M*,*
2. $U_1\langle\epsilon\rangle, \dots, U_n\langle\epsilon\rangle$ *represent an injective resolution* $I^0 \xrightarrow{U_n\langle\epsilon\rangle} \cdots \xrightarrow{U_1\langle\epsilon\rangle} I^n \to 0 \cdots$ *of* M*,*
3. $U_1\langle\epsilon\rangle^T, \dots, U_n\langle\epsilon\rangle^T$ *represent a free resolution* $\cdots 0 \to G_n \xrightarrow{U_n\langle\epsilon\rangle^T} \cdots \xrightarrow{U_1\langle\epsilon\rangle^T} G_0$ *of* M^**.*

In this case $I^q = \nu F_{n-q}\langle\epsilon\rangle = G_q^*$ *for all* q*.*

▶ **Example 15.** Consider the module

$M =$  ,

where $M_z = k$ if z lies in the shaded region, $M_z = 0$ otherwise, and all structure morphisms between non-zero vector spaces of M being identities. The first line of the following diagram exhibits $F_\bullet$ as a free resolution of M, and the second line exhibits $\nu F_\bullet[2]\langle\epsilon\rangle$ as an injective resolution of M:



3.3 Pulling back modules from the colimit

From now on, we consider $\mathbf{Z}^2$-persistence modules only. It remains to explain how we compute $H^\bullet(N^\bullet(K_*))$. In principle, this could be done by a procedure analogous to the LW-Algorithm described in Section 2.4: the horizontal sequence in the commutative diagram

$$\begin{array}{ccccccccc} & & N^d(K_*) & & & & & & \\ & & \downarrow & \searrow^{\delta^{d+1}} & & & & & \\ 0 & \longrightarrow & Z^{d+1}(N^\bullet(K_*)) & \xrightarrow[i^{d+1}]{} & N^{d+1}(K_*) & \xrightarrow{p^{d+1}} & Z^{d+2}(N^\bullet(K_*)) & & \\ & & & & & \searrow^{\delta^{d+2}} & \downarrow i^{d+2} & & \\ & & & & & & N^{d+2}(K_*) & & \\ & & & & & & & \searrow^{\delta^{d+3}} & \\ & & & & & & & & N^{d+3}(K_*) \end{array} \tag{6}$$

is a free resolution of $H^{d+2}(N^\bullet(K_*))$, and we can obtain matrices representing this resolution as described in Section 2.4. This would, however, involve the coboundary maps δ^{d+2} and

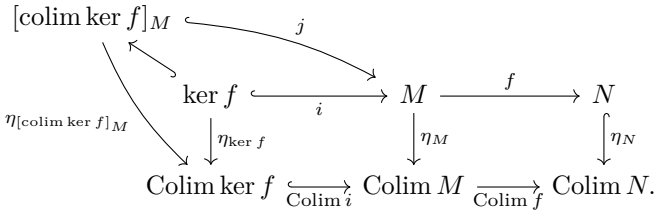
δ^{d+3}, leading to a very expensive computation, especially for function-Rips bifiltrations. Instead, we propose a method that computes a free resolution of $H^{d+2}(N^\bullet(K_*))$ from δ^{d+1} only.

For a vector space V, denote by ΔV the persistence modules with components $(\Delta V)_z = V$, such that all structure morphisms of ΔV are the identity. Let $\operatorname{Colim} M = \Delta \operatorname{colim} M$.

▶ **Definition 16.** *For a module $M \in \mathbf{Z}^2$-**pers** and a vector space $V \subseteq \operatorname{colim} M$, we let $[V]_M \in \mathbf{Z}^2$-**pers** be the preimage of ΔV under the canonical map $\eta_M \colon M \to \operatorname{Colim} M$.*

▶ **Lemma 17.** *If $f \colon M \to N$ is a morphism and N is free, then $\ker f = [\operatorname{colim} \ker f]_M$.*

Proof. If N is free, then $\eta_N \colon N \to \operatorname{Colim} N$ is injective. For every submodule $L \subseteq M$, we have $L \subseteq \eta_M^{-1}(\operatorname{Colim} L) = [\operatorname{colim} L]_M$, so $\ker f \subseteq [\operatorname{colim} \ker f]_M$. It remains to show the other inclusion $[\operatorname{colim} \ker f]_M \subseteq \ker f$. Consider the commutative diagram



The functor Colim is a directed colimit and thus exact. Therefore, $\operatorname{Colim} \ker f = \ker \operatorname{Colim} f$. This implies $\eta_N \circ f \circ j = \operatorname{Colim} f \circ \operatorname{Colim} i \circ \eta_{[\operatorname{colim} \ker f]_M} = 0$. Since η_N is injective, we obtain $f \circ j = 0$. Therefore, j factors uniquely through $\ker f$. This proves the claim. ◀

The *lexicographic order* $\preceq_{\text{lex}}$ and the *colexicographic order* $\preceq_{\text{colex}}$ are the total orders on $\mathbf{Z}^2$ defined as

$$(x, y) \preceq_{\text{lex}} (x', y') \quad \text{iff either } x < x' \text{ or } x = x' \text{ and } y \leq y',$$
$$(x, y) \preceq_{\text{colex}} (x', y') \text{ iff either } y < y' \text{ or } y = y' \text{ and } x \leq x'.$$

Two grades $z_1, z_2 \in \mathbf{Z}^2$ satisfy $z_1 \leq z_2$ iff $z_1 \preceq_{\text{colex}} z_2$ and $z_1 \preceq_{\text{lex}} z_2$.

▶ **Definition 18.** *For $b \in k^m$ and $r \in (\mathbf{Z}^2)^m$, the* lex pivot *of b with respect to r,* l-piv(b)*, is the smallest index i such that $b_i \neq 0$ and r_i takes its maximum value with respect to $\preceq_{\text{lex}}$. The* colex piot*,* c-piv(b)*, is defined analogously. For $0 \in k^m$, we let* l-piv$(0) =$ c-piv$(0) = 0$.

▶ **Theorem 19.** *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 $[V]_M$ is free, and Algorithm 1 calculates a graded matrix representing a basis of $[V]_M$.*

For a tuple $r \in (\mathbf{Z}^2)^m$ and a matrix M with m rows, let $[M]_r$ be the graded matrix with $\operatorname{rg}_i^{[M]_r} = r_i$ and $\operatorname{cg}_j^{[M]_r} = \bigvee_{M_{ij} \neq 0} r_i$. Then $[M]_r$ has the least possible column grades for which $[M]_r$ represents a map of free modules.

Proof of Theorem 19. Let $m_1, \ldots, m_s$ be a basis of M and $r = (g(m_1), \ldots, g(m_s))$. Without loss of generality, we assume that B represents a basis of V. The first for-loop in Algorithm 1 is a standard reduction scheme. In each iteration, the pivot index of one column decreases, so the loop terminates. When it does, all columns have distinct colex-pivots. During each iteration of the second for-loop, the lex-pivot of a column decreases. When it terminates, all columns have distinct lex-pivots. During the second loop, line $(*)$ ensures that no column

Algorithm 1 Computes a basis of $[V]_M$, where $M = \bigoplus_{i=1}^{m} F(r_i)$ and $V \subseteq \operatorname{colim} M$.

Data: An $m \times n$-matrix B representing a generating set of V, $r = (r_1, \dots, r_m) \in (\mathbf{Z}^2)^m$.
Result: A graded $m \times n$-matrix whose nonzero columns represent a basis of $[V]_M$.
function `Bireduce`(B)**:**
  $p \leftarrow 0 \in [m]^n$
  for $j = 1, \dots, n$ **do**
    while $i \leftarrow \text{c-piv}(B_j) \neq 0$ **do**
      if $p_i = 0$ **then** $p_i \leftarrow j$; **break**
      $B_j \leftarrow B_j + B_{p_i}$
  $p \leftarrow 0 \in [m]^n$
  for $j' = 1, \dots, n$ **do**
    $j \leftarrow j'$
    while $i \leftarrow \text{l-piv}(B_j) \neq 0$ **do**
      if $p_i = 0$ **then** $p_i \leftarrow j$; **break**
(*)   **if** $\text{c-piv}(B_j) < \text{c-piv}(B_{p_i})$ **then** swap p_i and j
      $B_j \leftarrow B_j + B_{p_i}$
  return $[B]_r$

is added to another column with a smaller colex-pivot. Since all columns have distinct colex-pivots after the first for-loop, the colex-pivots of the columns thus do not change during the second for-loop. Therefore, when the algorithm terminates, all columns of B have pairwise distinct lex- and colex-pivots.

Let $A = [B]_r$ for the state of B when the algorithm terminates. Then A represents a basis $\alpha_1, \dots, \alpha_t$ of a free submodule N of M, with $g(\alpha_j) = \operatorname{cg}_j^A$. It remains to show that $N = [V]_M$. Since all column operations performed by Algorithm 1 are invertible, A represents a basis of V. Therefore, $\operatorname{colim} N = V$, which implies $N \subseteq [V]_M$. Let $v \in [V]_M$. Then there are unique coefficients ξ_j such that $\eta_M(v) = \sum_{j=1}^{t} \xi_j \eta_M(\alpha_j)$. Since η_M is injective, $M_{g(v)\leq z}(v) = \sum_{j=1}^{t} \xi_j M_{g(\alpha_j)\leq z}(\alpha_j)$ for all $z \geq g(v) \vee \bigvee_{\xi_j \neq 0} \operatorname{cg}_j^A$. Since all columns of A have distinct lex- and colex-pivots, v cannot have smaller grade than $\bigvee_{\xi_j \neq 0} \operatorname{cg}_j^A$, so $v \in N$. This proves the claim. ◀

3.4 The free resolution of cohomology

Assume $C^\bullet$ is a cochain complex of free modules such that $\operatorname{colim} H^\bullet(C^\bullet) = 0$, and recall the commutative diagram (6). A matrix $[\delta^{d+1}]$ representing δ^{d+1} is a generating system for $\operatorname{colim} Z^{d+1}(C^\bullet)$, and Lemma 17 states that $Z^{d+1}(C^\bullet) = [\operatorname{colim} Z^{d+1}(C^\bullet)]_{C^{d+1}}$. Applying Algorithm 1 to $[\delta^{d+1}]$ thus yields a graded matrix $[i^{d+1}]$ representing a basis of $Z^{d+1}(C^\bullet)$.

▶ **Lemma 20.** *If* $0 \to F_2 \xrightarrow{f_2} F_1 \xrightarrow{f_1} F_0$ *is a free resolution of a module of finite total dimension, then* $(\nu F_0)^* = \ker(\nu f_2)^*$.

Proof. The sequence $0 \to F_2 \xrightarrow{f_2} F_1 \xrightarrow{f_1} F_0$ is exact. By Theorem 10, the sequence $\nu F_2 \xrightarrow{\nu f_2} \nu F_1 \xrightarrow{\nu f_1} \nu F_0 \to 0$ and therefore also the dual sequence $0 \to (\nu F_0)^* \xrightarrow{(\nu f_1)^*} (\nu F_1)^* \xrightarrow{(\nu f_2)^*} (\nu F_2)^*$ are exact. ◀

Thus, if a matrix $[f_2]$ representing f_2 is known, then the matrix $[f_1]^T = [(\nu f_1)^*]$ can be computed by applying the LW-Algorithm (Algorithm 4 in the full version) to $[f_2]^T = [(\nu f_2)^*]$. In particular, if $H_d(C^\bullet)$ has finite total dimension, then so has $H^{d+2}(N^\bullet(K_*))$, so Lemma 20 can be applied to the free resolution (6) of $H^{d+2}(N^\bullet(K_*))$. This shows that $[p^{d+1}]^T$ can be computed by applying the LW-Algorithm to $[i^{d+1}]^T$.

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

Input: Graded matrices $[\delta^\bullet]$ representing $C^\bullet$.
Output: Pairs of graded matrices representing a free resolution of $H^d(C^\bullet)$ for $d = 0, 1, \dots$.
$q \leftarrow \emptyset$ ▷ *pivots for clearing*
for $d = 0, 1, \dots$ **do**
  for $j \in q$ **do** $[\delta^{d+1}]_j \leftarrow 0$ ▷ *clearing*
  $[i^{d+1}] \leftarrow$ `Bireduce`$([\delta^{d+1}])$
  $n \leftarrow$ #columns of $[i^{d+1}]$
  $q \leftarrow \{\mathrm{piv}[i^{d+1}]_1, \dots, \mathrm{piv}[i^{d+1}]_n\}$
  $[i^{d+1}]^T, [p^{d+1}]^T \leftarrow$ `MGSWithKer`$([i^{d+1}]^T)$ ▷ *See Algorithm 5 in the full version*
  yield `MinimizeChainComplex`$([i^{d+1}], [p^{d+1}])$ ▷ MFR of $H^{d+2}(\nu C^\bullet\langle\epsilon\rangle) \cong H^d(C^\bullet)$

▶ **Corollary 21.** *Let $C^\bullet$ be a cochain complex of free modules such that* $\dim H^\bullet(C^\bullet)$ *is finite. Then Algorithm 2 computes free resolutions of $H^d(C^\bullet)$.*

▶ Remark 22 (Clearing). In general, $[\delta^{d+1}]$ is not injective. As in one-parameter persistent cohomology, the first loop in Algorithm 1 spends a significant amount of time on reducing the columns of $[\delta^{d+1}]$ that are eventually reduced to zero. The computation can be accelerated considerably by using the pivots of the reduced matrix $[\delta^d]$ to implement a clearing scheme before invoking Algorithm 1. This is implemented in Algorithm 2.

4 Experiments

We have implemented our cohomology algorithm in C++ [33]. We have also implemented the algorithm [23] used in `mpfree` [31], in order to vary the implementation details. Where applicable, the run time of our clone is similar to the one of `mpfree`. We have run our implementation to compute MFRs of the PH of various function-Rips bifiltrations.

4.1 Setup

All computations are done with coefficients in $k = \mathbf{F}_2$. Matrix columns are implemented as binary heaps [7]. Our code also implements an alternative representation of columns as dynamically allocated arrays. We have run our code on a MacBook Pro 2017 with a 2.3 GHz Dual-Core Intel Core i5 and 16GB RAM. The code is compiled using clang++ 15.0.7. Each instance of our program may run four threads in parallel.

The run time of the homology algorithm for minimal presentation computation [23] is dominated by chunk preprocessing and the LW-Algorithm. While it is standard to implement chunk preprocessing in an embarrassingly parallel way, no way is known to parallelize the LW-Algorithm. While the minimization step in Algorithm 2 is parallelized in our implementation, the bigraded reduction (Algorithm 1) is not, although we hypothesize it could be parallelized analogously to [36] or [6]. We found that the minimization is not a performance bottleneck of our algorithm, so one would expect similar performance on a single core.

Datasets. We have generated point clouds S by sampling n-spheres S^n, n-tori $S^1 \times \cdots \times S^1$ and orthogonal groups $O(n)$. Additionally, we use some of the point clouds from [37]. To each point $p \in S$, we associate the value $\sum_{q \in S \setminus \{p\}} \exp\left(-\frac{d(p,q)^2}{2\sigma^2}\right)$ for a manually chosen parameter σ. These values and a distance matrix are written to a file, from which the program generates the coboundary matrices of the associated full function-Rips bifiltration.

Chunk preprocessing. The LW-Algorithm works most efficiently if combined with chunk preprocessing [22, 23]; this is the approach implemented in `mpfree`. Chunk preprocessing (Algorithm 7 in the full version) applies a certain column operation scheme to the matrices representing the chain complex $C_\bullet$. As an alternative, we propose to manipulate $C_\bullet(K_*)$ by row operations. Equivalently, one can see this procedure as column operations on the matrices representing the cochain complex $\nu C^\bullet(K_*)$; hence, we refer to this approach as *cochain complex chunk preprocessing*; see Algorithm 8 in the full version.

Coning off. To ensure that $H^{d+2}(N^\bullet(K_*)) \cong H_d(K_*)^*$, Corollary 21 requires that $C_\bullet$ has homology of finite total dimension. Therefore, our implementation offers the ability to cone off the complex as follows. Let $C_\bullet$ be a chain complex of free $\mathbf{Z}^2$-persistence modules. The assignment $C'_d\colon y \mapsto \operatorname{colim}_{x\in\mathbf{Z}}(C_d)_{xy}$ defines a chain complex $C'_\bullet$ of free $\mathbf{Z}$-persistence modules. We compute the barcode of $H_d(C'_\bullet)$ using the cohomological standard algorithm with clearing (Algorithm 3 in the full version). This can be used to implement a clearing mechanism in the homological standard algorithm, which we use to compute representatives for the homology classes in the barcode of $H_d(C'_\bullet)$. Let y_0 such that $y_0 \geq g_y(\sigma)$ for all σ, where $g(\sigma) = (g_x(\sigma), g_y(\sigma))$. Let $\hat{C}_\bullet = C_\bullet$. For every bar (b, d) of $H_d(C'_\bullet)$ of non-zero length represented by a q-cycle $c \in C'_q$ of grade $g(c) = b$, we add a basis element $\hat{c}$ of grade $g(\hat{c}) = (b, y_0)$ to $\hat{C}_{q+1}$ with $\partial_{q+1}(\hat{c}) = c$. If $d < \infty$, then c bounds a chain c' with $g(c') = d$, and we add a basis element $\hat{c}'$ of grade $g(\hat{c}') = (d, y_0)$ to $\hat{C}_{q+2}$ with $\partial_{q+2}(\hat{c}') = c' - \hat{c}$. The resulting chain complex $\hat{C}_\bullet$ satisfies $H_\bullet(\hat{C}_\bullet)_{xy} = 0$ for $y \geq y_0$. If not stated otherwise, cohomology computation is done with this preprocessing applied to the density parameter.

Sparsification. We observe that the second for-loop in Algorithm 1 runs considerably longer than the first. The loop also increases the matrix density, which many incur a high cost on the subsequent steps in Algorithm 2. For an interpretation, see Remark 24 in the full version. As a remedy, we have added a step that decreases the sparsity of the matrix using row operations that are compatible with the column sparse matrix implementation. Specifically, if a row contains only a single entry, any row addition from this row to another affects only a single entry. Therefore, an entry in a row with grade g can be eliminated directly if there is a row with grade $g' \geq g$ containing only a single entry in the same column; see Algorithm 10 in the full version. All cohomology computation run times are reported with this sparsification scheme applied.

4.2 Results

An overview of the results is given in Table 1. We report only the time needed to compute the MFR and, if applicable, to apply the chunk preprocessing. We do not report the time necessary to set up the (co)boundary matrices. In all cases with $d \geq 2$, computing $H^{d+2}(N^\bullet)$ (without chunk preprocessing) was faster than computing H_d with chunk preprocessing. Our cohomology approach does not benefit from chunk preprocessing. The speedup of the cohomology approach increases with dimension. For two instances, computation of $H_d(K_*)$ did not terminate within five minutes, while computing $H^{d+2}(N^\bullet(K_*))$ was no problem. We also observe that the cohomology algorithm uses less memory for almost all instances with $d \geq 2$.

Matrix representations. The efficiency of the LW-Algorithm and of chunk preprocessing does not vary very much depending on the matrix implementation; see Table 1 and Table 2 in the full version. In contrast, our cohomology algorithm runs faster in the implementation

Table 1 Run times (in milliseconds) comparing our implementation of [35, 23] (including chunk preprocessing) and our cohomology algorithm, applied a density-Rips filtration on 300 vertices ($d = 1$), 100 vertices ($d = 2$) and 60 vertices ($d = 3$). RSS is peak resident memory as measured by `time`. Speed up is the run time of the homology computation (including chunk preprocessing), divided by the run time of the cohomology algorithm. The program has been killed after exceeding five minutes of run time.

d	sample	chunk	H_d	sum	RSS	H^{d+2}	RSS	speedup
1	c. elegans	5,457	40,841	46,298	6,423,600	119,444	6,526,976	0.39
	2-torus	11,480	19,875	31,355	6,620,404	5,032	2,827,912	6.23
	4-torus	6,342	28,627	34,969	5,916,816	50,607	3,384,472	0.69
	dragon	9,721	18,489	28,210	5,774,492	5,064	2,829,620	5.57
	2-sphere	8,657	29,180	37,837	6,421,312	25,021	4,098,268	1.51
	4-sphere	7,699	33,619	41,318	6,642,260	47,355	7,154,632	0.87
	$O(3)$	7,023	33,874	40,897	6,315,708	42,702	3,816,124	0.96
2	c. elegans	28,583	7,484	36,067	4,428,440	5,655	2,371,324	6.38
	2-torus	39,630	2,191	41,821	3,216,372	5,054	2,334,712	8.27
	4-torus	33,788	19,875	53,663	5,538,232	5,969	2,425,136	8.99
	dragon	19,023	2,379	21,402	2,557,124	5,188	2,367,488	4.13
	2-sphere	32,611	12,416	45,027	5,099,604	6,417	2,426,924	7.02
	4-sphere	29,272	25,357	54,629	6,039,576	8,637	2,505,664	6.32
	$O(3)$	31,780	29,123	60,903	6,654,692	6,796	2,445,996	8.96
3	c. elegans	38,349	2,393	40,742	3,515,708	8,984	4,227,820	4.53
	2-torus	>300,000	–	–	7,141,648	11,725	5,072,192	>25.59
	4-torus	>300,000	–	–	10,843,356	9,930	4,358,580	>30.21
	dragon	67,463	2,334	69,797	6,666,732	9,782	4,900,800	7.14
	2-sphere	59,385	3,110	62,495	4,280,112	9,051	4,185,036	6.90
	4-sphere	92,365	8,577	100,942	5,526,344	8,818	4,197,636	11.45
	$O(3)$	204,263	25,284	229,547	7,966,600	10,851	4,365,864	21.15

with binary heaps. We observe that the vector based implementations generally use less memory than the heap based ones. This happens because, in contrast to vectors, heaps may contain multiple entries for the same row index.

Cochain chunk preprocessing. For the homology computation, the cochain complex chunk preprocessing described above often is more efficient than chunk preprocessing if combined with the heap implementation of matrix columns, see Table 3 in the full version. This is true in particular for higher homology dimensions. If combined with vector-based matrices, cochain chunk preprocessing is less efficient than conventional chunk preprocessing in almost all cases, and does not terminate at all within five minutes.

References

1 Manu Aggarwal and Vipul Periwal. Dory: Overcoming Barriers to Computing Persistent Homology, March 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)*, pages 27–38, 2023. `doi:10.1137/1.9781611977561.ch3`.

3 Ibrahim Assem, Daniel Simson, and Andrzej Skowronski. *Elements of the Representation Theory of Associative Algebras*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2006.

4 Ulrich Bauer. Ripser: efficient computation of Vietoris–Rips persistence barcodes. *J. Appl. Comput. Topol.*, 5(3):391–423, 2021. doi:10.1007/s41468-021-00071-5.

5 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Clear and Compress: Computing Persistent Homology in Chunks. In Peer-Timo Bremer, Ingrid Hotz, Valerio Pascucci, and Ronald Peikert, editors, *Topological Methods in Data Analysis and Visualization III*, pages 103–117. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-04099-8_7.

6 Ulrich Bauer, Michael Kerber, and Jan Reininghaus. Distributed Computation of Persistent Homology. In Catherine C. McGeoch and Ulrich Meyer, editors, *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 31–38. Society for Industrial and Applied Mathematics, Philadelphia, PA, May 2014. doi:10.1137/1.9781611973198.4.

7 Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat—persistent homology algorithms toolbox. *J. Symbolic Comput.*, 78:76–90, 2017. doi:10.1016/j.jsc.2016.03.008.

8 Ulrich Bauer, Fabian Lenzen, and Michael Lesnick. Efficient two-parameter persistence computation via cohomology, March 2023. arXiv:2303.11193.

9 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, 6(2):162–191, March 2015. doi:10.20382/jocg.v6i2a9.

10 Ulrich Bauer and Maximilian Schmahl. Lifespan Functors and Natural Dualities in Persistent Homology, October 2021. To appear in Homology, Homotopy and Applications. arXiv:2012.12881.

11 Katherine Benjamin, Aneesha Bhandari, Zhouchun Shang, Yanan Xing, Yanru An, Nannan Zhang, Yong Hou, Ulrike Tillmann, Katherine R Bull, and Heather A Harrington. Multiscale topology classifies and quantifies cell types in subcellular spatial transcriptomics. *arXiv preprint arXiv:2212.06505*, 2022.

12 Andrew J. Blumberg and Michael Lesnick. Stability of 2-Parameter Persistent Homology. *Foundations of Computational Mathematics*, October 2022. doi:10.1007/s10208-022-09576-6.

13 Magnus Bakke Botnan and Michael Lesnick. An Introduction to Multiparameter Persistence, March 2022. arXiv:2203.14289.

14 Gunnar Carlsson and Afra Zomorodian. The Theory of Multidimensional Persistence. *Discrete & Computational Geometry*, 42(1):71–93, July 2009. doi:10.1007/s00454-009-9176-0.

15 Chao Chen and Michael Kerber. Persistent Homology Computation with a Twist. In *27th European Workshop on Computational Geometry*, 2011. URL: https://eurocg11.inf.ethz.ch/abstracts/22.pdf.

16 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37:103–120, January 2007. doi:10.1007/s00454-006-1276-5.

17 René Corbet, Michael Kerber, Michael Lesnick, and Georg Osang. Computing the multicover bifiltration. In Kevin Buchin and Éric Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:17, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2021.27.

18 Vin de Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, December 2011. doi:10.1088/0266-5611/27/12/124003.

19 Edelsbrunner, Letscher, and Zomorodian. Topological Persistence and Simplification. *Discrete & Computational Geometry*, 28(4):511–533, November 2002. doi:10.1007/s00454-002-2885-2.

20 Herbert Edelsbrunner and John Harer. Persistent homology—a survey. In Jacob E. Goodman, János Pach, and Richard Pollack, editors, *Contemporary Mathematics*, volume 453, pages 257–282. American Mathematical Society, Providence, Rhode Island, 2008. doi:10.1090/conm/453/08802.

21 David Eisenbud. *Commutative Algebra*, volume 150 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 1995. doi:10.1007/978-1-4612-5350-1.

22 Ulderico Fugacci and Michael Kerber. Chunk reduction for multi-parameter persistent homology. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 37:1–37:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2019.37.

23 Ulderico Fugacci, Michael Kerber, and Alexander Rolle. Compression for 2-parameter persistent homology. *Computational Geometry*, 109:101940, February 2023. doi:10.1016/j.comgeo.2022.101940.

24 Victor Ginzburg. Calabi–Yau algebras, 2007. arXiv:math/0612139.

25 Barbara Giunti and Jānis Lazovskis. TDA-Applications (An online database of papers on applications of TDA outside of math), 2021. URL: https://www.zotero.org/groups/2425412/tda-applications.

26 Gregory Henselman and Robert Ghrist. Matroid Filtrations and Computational Persistent Homology, October 2017. arXiv:1606.00199.

27 Gregory Henselman-Petrusek. Eirene, 2021. URL: https://github.com/Eetion/Eirene.jl.

28 Michael Höppner and Helmut Lenzing. Projective diagrams over partially ordered sets are free. *Journal of Pure and Applied Algebra*, 20(1):7–12, January 1981. doi:10.1016/0022-4049(81)90045-1.

29 Bernhard Keller. Calabi–Yau triangulated categories. In Andrzej Skowroński, editor, *Trends in Representation Theory of Algebras and Related Topics*, EMS Series of Congress Reports, pages 467–489. EMS Press, first edition, September 2008. doi:10.4171/062-1/11.

30 Michael Kerber. Multi-parameter persistent homology is practical. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. URL: https://openreview.net/pdf?id=TDU6UycGYxR.

31 Michael Kerber. mpfree, 2021. URL: https://bitbucket.org/mkerber/mpfree.

32 Michael Kerber and Alexander Rolle. Fast Minimal Presentations of Bi-graded Persistence Modules. In Martin Farach-Colton and Sabine Storandt, editors, *2021 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, Proceedings, pages 207–220. Society for Industrial and Applied Mathematics, January 2021. doi:10.1137/1.9781611976472.16.

33 Fabian Lenzen. 2-parameter persistent cohomology, 2023. URL: https://gitlab.com/flenzen/2-parameter-persistent-cohomology.

34 Michael Lesnick and Matthew Wright. Interactive Visualization of 2-D Persistence Modules, December 2015. arXiv:1512.00180.

35 Michael Lesnick and Matthew Wright. Computing Minimal Presentations and Bigraded Betti Numbers of 2-Parameter Persistent Homology. *SIAM Journal on Applied Algebra and Geometry*, 6(2):267–298, June 2022. doi:10.1137/20M1388425.

36 Dmitriy Morozov and Arnur Nigmetov. Towards Lockfree Persistent Homology. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, pages 555–557, Virtual Event USA, July 2020. ACM. doi:10.1145/3350755.3400244.

37 Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(1):17, December 2017. doi:10.1140/epjds/s13688-017-0109-5.

38 Irena Peeva. *Graded Syzygies*. Springer London, London, 2011. doi:10.1007/978-0-85729-177-6.

39 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, August 2021. arXiv:2107.05412.

40 Daniel Quillen. Projective modules over polynomial rings. *Inventiones Mathematicae*, 36(1):167–171, December 1976. doi:10.1007/BF01390008.

41 Alexander Rolle. Multi-parameter hierarchical clustering and beyond. In *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020. URL: https://openreview.net/pdf?id=g0-tBxQTPRy.

42 Sara Scaramuccia, Federico Iuricich, Leila De Floriani, and Claudia Landi. Computing multiparameter persistent homology through a discrete Morse-based approach. *Computational Geometry*, 89:101623, August 2020. doi:10.1016/j.comgeo.2020.101623.

43 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*, pages 309–314, 2012. URL: http://2012.cccg.ca/papers/paper52.pdf.

44 Andrei Suslin. Projective modules over polynomial rings are free. *Soviet Mathematics*, 17(4):1160–1164, 1976.

45 The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.6.0 edition, 2022. URL: https://gudhi.inria.fr/doc/3.6.0/.

46 Oliver Vipond, Joshua A Bull, Philip S Macklin, Ulrike Tillmann, Christopher W Pugh, Helen M Byrne, and Heather A Harrington. Multiparameter persistent homology landscapes identify immune cell spatial patterns in tumors. *Proceedings of the National Academy of Sciences*, 118(41):e2102166118, 2021.

47 Cary Webb. Decomposition of Graded Modules. *Proceedings of the American Mathematical Society*, 94(4):565–571, 1985. doi:10.2307/2044864.

48 Charles A. Weibel. *An Introduction to Homological Algebra*. Number 38 in Cambridge Studies in Advanced Mathematics. Cambridge Univ. Press, Cambridge, reprint. 1997, transf. to digital print edition, 2003.

49 Simon Zhang, Mengbai Xiao, and Hao Wang. GPU-Accelerated Computation of Vietoris-Rips Persistence Barcodes. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 70:1–70:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2020.70.

50 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33:249–274, February 2005. doi:10.1007/s00454-004-1146-y.