Computational Topology for Data Analysis: Notes from Book by

Tamal Krishna Dey
Department of Computer Science
Purdue University
West Lafayette, Indiana, USA 46907

Yusu Wang
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, California, USA 92093
Topic 14: Multiparameter Persistence Module Decomposition

In previous chapters, we have considered filtrations that are parameterised by a single parameter such as $\mathbb{Z}$ or $\mathbb{R}$. Naturally, they give rise to a 1-parameter persistence module. In this chapter, we generalize the concept and consider persistence modules that are parameterized by one or more parameters such as $\mathbb{Z}^d$ or $\mathbb{R}^d$. They are called multiparameter persistence modules in general. Multiparameter persistence modules naturally arise from filtrations that are parameterized by multiple values such as the one shown in Figure 14.1 over two parameters.

![Figure 14.1: A bi-filtration parameterized over curvature and radius; picture taken from [6].](image)

The classical algorithm of Edelsbrunner et al. [11] presented earlier provides a unique decomposition of the 1-parameter persistence module over $\mathbb{Z}$ implicitly generated by an input simplicial filtration. Similarly, a multiparameter persistence module $M$ over the grid $\mathbb{Z}^d$ can be implicitly given by an input multiparameter finite simplicial filtration and we look for computing a decomposition (Definition 10) $M = \bigoplus_i M_i$. The modules $M_i$ are the counterparts of bars in the 1-parameter case and are called indecomposables. These indecomposables are more complicated and cannot be completely characterized as in the one-parameter case. Nonetheless, for finitely generated persistence modules defined over $\mathbb{Z}^d$, their existence is guaranteed by the Krull-Schmidt theorem [2]. Figure 14.2 illustrates indecomposables of some modules.

An algorithm for decomposing a multiparameter persistence module can be derived from the so-called Meataxe algorithm which applies to much more general modules than we consider in TDA at the expense of high computational cost. Sacrificing this generality and still encompassing a large class of modules that appear in TDA, we can design a much more efficient algorithm. Specifically, we present an algorithm that can decompose a finitely presented module with a time complexity that is much better than the Meataxe algorithm though we lose the generality as the module needs to be distinctly graded as explained later.

For measuring algorithmic efficiency, it is imperative to specify how the input module is pre-
Assuming an index set of size $m$ and vector spaces of dimension $O(m)$, a 1-parameter persistence module can be presented by a set of matrices of dimensions $O(m) \times O(m)$ each representing a linear map $M_i \rightarrow M_{i+1}$ between two consecutive vector spaces $M_i$ and $M_{i+1}$. This input format is costly as it takes $O(m^3)$ space ($O(m^2)$-size matrix for each index) and also does not appear to offer any benefit in time complexity for computing the bars. An alternative presentation is obtained by considering the persistence module as a graded module over a polynomial ring $\mathbb{k}[t]$ and presenting it with the so-called generators $\{g_i\}$ of the module and relations $\sum_i \alpha_i g_i = 0 \mid \alpha_i \in \mathbb{k}[t]$ among them. A presentation matrix encoding the relations in terms of the generators characterizes the module completely. Then, a matrix reduction algorithm akin to the persistence algorithm MatPersistence that we covered earlier provides the desired decomposition\(^1\). Figure 14.3 illustrates the advantage of this presentation over the other costly presentation. In practice, when the 1-parameter persistence module is given by an implicit simplicial filtration, one can apply the matrix reduction algorithm directly on a boundary matrix rather than first computing a presentation matrix from it and then decomposing it. If there are $O(n)$ simplices constituting the filtration, the algorithm runs in $O(n^3)$ time with simple matrix reductions and in $O(n^\omega)$ time with more sophisticated matrix multiplication techniques where $\omega < 2.373$ is the exponent for matrix multiplication.

The Meataxe algorithm for multiparameter persistence modules follows the costly approach analogous to the one in the 1-parameter case that expects the presentation of each individual linear map explicitly. In particular, it expects the input $d$-parameter module $M$ over a finite subset of $\mathbb{Z}^d$ to be given as a large matrix in $\mathbb{k}^{D \times D}$ with entries in a fixed field $\mathbb{k} = \mathbb{Z}_q$, where $D$ is the sum of dimensions of vector spaces over all points in $\mathbb{Z}^d$ supporting $M$. The time complexity of the Meataxe algorithm is $O(D^6 \log q)$ [15]. In general, $D$ might be quite large. It is not clear what is

\(^1\)Notice that persistence algorithm takes a filtration as input whereas here we are considering a module presented with matrices as input.
the most efficient way to transform an input that specifies generators and relations (or a simplicial filtration) to a representation matrix required by the Meatax algorithm. A naive approach is to consider the minimal sub-grid in $\mathbb{Z}^d$ that supports the non-trivial maps. In the worst-case, with $N$ being the total number of generators and relations, one has to consider $O\left(\binom{N}{d}\right) = O(N^d)$ grid points in $\mathbb{Z}^d$ each with a vector space of dimension $O(N)$. Therefore, $D = O(N^{d+1})$ giving a worst-case time complexity of $O(N^{6d(d+1)} \log q)$. Even allowing approximation, the algorithm runs in $O(N^{3d+1} \log q)$ time [16].

In this chapter, we take the alternate approach where the module is treated as a finitely presented graded module over multivariate polynomial ring $R = \mathbb{k}[t_1, \cdots, t_d]$ [7] and presented with a set of generators and relations graded appropriately. Given a presentation matrix encoding relations with generators, our algorithm computes a diagonalization of the matrix giving a presentation of each indecomposable which the input module decomposes into. Compared to the 1-parameter case, we have to cross two main barriers for computing the indecomposables. First, we need to allow row operations along with column operations for reducing the input matrix. In 1-parameter case, row operations become redundant because column operations already produce the bars. Second, unlike in 1-parameter case, we cannot allow all left-to-right column or bottom-to-top row operations for the matrix reduction because the parameter space $\mathbb{Z}^d$, $d > 1$, unlike $\mathbb{Z}$ only induces a partial order on these operations. These two difficulties are overcome by an incremental approach combined with a linearization trick. Given a presentation matrix with a total of $O(N)$ generators and relations that are graded distinctly, the algorithm runs in $O(N^{2d+1})$ time. Surprisingly, the complexity does not depend on the parameter $d$.

Computing presentation matrix from a multiparameter simplicial filtration is not easy. For $d$-parameter filtrations with $n$ simplices, a presentation matrix of size $O(n^d) \times O(n^d)$ can be computed in $O(n^{2d+1})$ time by an algorithm presented in [10]. We will not present this construction here. Instead, we focus on the two cases, 2-parameter persistence modules where the homology groups could be multi-dimensional and $d$-parameter persistence modules where the homology group is only zero dimensional. For these two cases, we can compute the presentation matrices more efficiently. For the 2-parameter case, Lesnick and Wright [22] gives an efficient $O(n^3)$ algorithm for computing a presentation matrix from an input filtration. In this case, it turns out that $N$, the total number of generators and relations, is $O(n)$. For the 0-th homology groups, presentation matrices are given by the boundary matrices straightforwardly as detailed in Section 14.5.2 giving
We define persistence modules in this chapter differently using the definition of graded modules in algebra. Graded module structures provide an appropriate framework for defining the multiparameter persistence, in particular, for the decomposition algorithm that we present. Also, navigating between the simplicial filtration and the module induced by it becomes natural with the graded module structure.

### 14.1 Multiparameter Persistence modules

We define persistence modules in this chapter differently using the definition of graded modules in algebra. Graded module structures provide an appropriate framework for defining the multiparameter persistence, in particular, for the decomposition algorithm that we present. Also, navigating between the simplicial filtration and the module induced by it becomes natural with the graded module structure.

#### 14.1.1 Persistence modules as graded modules

First, we recollect the definition of modules. It requires a ring. We consider a module where the ring action satisfies that

\[ \forall m \in M, \forall r \in R, \quad r \cdot (m + n) = r \cdot m + r \cdot n \]

and

\[ \forall m, n \in M, \forall r \in R, \quad (r_1 + r_2) \cdot m = r_1 \cdot m + r_2 \cdot m \]

\[ \forall m, n \in M, \forall r \in R, \quad r \cdot (m \cdot n) = (r \cdot m) \cdot n = m \cdot (r \cdot n) \]

Definition 1 (Polynomial ring). Given a variable \( t \) and a field \( \mathbf{k} \), the set of polynomials given by

\[ \mathbf{k}[t] = \{ a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \mid n \geq 0, a_i \in \mathbf{k} \} \]

forms a ring with usual polynomial addition and multiplication operations. The definition can be extended to multivariate polynomials

\[ \mathbf{k}[t] = \mathbf{k}[t_1, \cdots, t_k] = \{ \Sigma_{i_1, \cdots, i_k} a_{i_1, \cdots, i_k} t_1^{i_1} \cdots t_k^{i_k} \mid i_1, \cdots, i_k \geq 0, a_{i_1, \cdots, i_k} \in \mathbf{k} \}. \]

We use polynomial ring to define multiparameter persistence modules. Specifically, let \( R = \mathbf{k}[t_1, \cdots, t_d] \) be the \( d \)-variate Polynomial ring for some \( d \in \mathbb{Z}_+ \) with \( \mathbf{k} \) being a field. Throughout this chapter, we assume coefficients are in \( \mathbf{k} \). Hence homology groups are vector spaces.

Definition 2 (Graded module). A \( \mathbb{Z}^d \)-graded \( R \)-module (graded module in brief) is an \( R \)-module \( M \) that is a direct sum of \( \mathbf{k} \)-vector spaces \( M_u \) indexed by \( u = (u_1, u_2, \ldots, u_d) \in \mathbb{Z}^d \), i.e. \( M = \bigoplus_u M_u \), such that the ring action satisfies that \( \forall i, \forall v \in \mathbb{Z}^d, t_i \cdot M_u \subseteq M_{u+e_i} \), where \( \{e_i\}_{i=1}^d \) is the standard basis in \( \mathbb{Z}^d \). The indices \( u \in \mathbb{Z}^d \) are called grades.

Another interpretation of graded module is that, for each \( u \in \mathbb{Z}^d \), the action of \( t_i \) on \( M_u \) determines a linear map \( t_i \cdot : M_u \rightarrow M_{u+e_i} \) by \( (t_i \cdot ) (m) = t_i \cdot m \). So, we can also describe a graded module equivalently as a collection of vectors spaces \( \{ M_u \}_{u \in \mathbb{Z}^d} \) with a collection of linear maps \( \{ t_i \cdot : M_u \rightarrow M_{u+e_i}, \forall i, \forall u \} \) where the commutative property \( (t_i \cdot ) (t_j \cdot .) = (t_j \cdot .) (t_i \cdot ) \) holds. The commutative diagram in Figure 14.4 shows a graded module for \( d = 2 \), also called a bigraded module.

Definition 3 (Graded module \( R \)). There is a special graded module \( M \) where \( M_u \) is the \( \mathbf{k} \)-vector space generated by \( t_i^u = t_1^{u_1} t_2^{u_2} \cdots t_d^{u_d} \) and the ring action is given by the ring \( R \). We denote it with \( R \) not to be confused with the ring \( R \) which is used to define it.

Before we introduce persistence modules as instances of graded modules, we extend the notion of simplicial filtration to the multiparameter framework.
Figure 14.4: A graded 2-parameter module. All sub-diagrams of maps and compositions of maps are commutative.

**Definition 4** $(d$-parameter filtration). A $(d$-parameter) simplicial filtration is a family of simplicial complexes \( \{X_u\}_{u \in \mathbb{Z}^d} \) such that for each grade \( u \in \mathbb{Z}^d \) and each \( i = 1, \ldots, d \), \( X_u \subseteq X_{u + e_i} \).

Figure 14.5 shows an example of a 2-parameter filtration and various graded modules associated with it. The module resulting with the homology group at the bottom right is a persistence module. The figure also shows other graded modules of chain groups (left) and boundary groups (middle).

**Definition 5** $(d$-parameter persistence module). We call a \( \mathbb{Z}^d \)-graded \( \mathbb{R} \)-module \( M \) a $d$-parameter persistence module when \( M_u \) for each \( u \in \mathbb{Z}^d \) is a homology group defined over a field and the linear maps corresponding to ring actions among them are induced by inclusions in a $d$-parameter simplicial filtration. We call \( M \) finitely generated if there exists a finite set of elements \( \{g_1, \ldots, g_n\} \subseteq M \) such that each element \( m \in M \) can be written as an \( \mathbb{R} \)-linear combination of these elements, i.e. \( m = \sum_{i=1}^{n} \alpha_i g_i \) with \( \alpha_i \in \mathbb{R} \). We call this set \( \{g_i\} \) a generating set or generators of \( M \). A generating set is called minimal if its cardinality is minimal among all generating sets. The \( \mathbb{R} \)-linear combinations \( \sum_{i=1}^{n} \alpha_i g_i \) that are 0 are called relations. We will see later that a module can be represented by a set of generators and relations.

In this exposition, we assume that all modules are finitely generated. Such modules always admit a minimal generating set. In our example in Figure 14.5, the vertex set \( \{v_p, v_r, v_g\} \) is a minimal generating set for the module of zero-dimensional homology groups.

**Definition 6** (Morphism). A graded module morphism, called morphism in short, between two modules \( M \) and \( N \) is defined as an \( \mathbb{R} \)-linear map \( f : M \to N \) preserving grades: \( f(M_u) \subseteq N_{u + e_i} \). Equivalently, it can also be described as a collection of linear maps \( \{f_u : M_u \to N_u\} \) which gives the following commutative diagram for each \( u \) and \( i \):

\[
\begin{array}{ccc}
M_u & \xrightarrow{f_u} & M_{u + e_i} \\
| & & | \\
\downarrow & & \downarrow \quad f_{u+e_i} \\
N_u & \xrightarrow{f_u} & N_{u + e_i}
\end{array}
\]
Figure 14.5: (top) An example of a 2-parameter simplicial filtration. Each square box indicates what is the current (filtered) simplicial complex at the bottom left grid point of the box. (bottom) We show different modules considering different abelian groups arising out of the complexes with the ring actions on the arrows (see Section 14.5 for details): (a) The module of 0-th chain groups $C_0, A = \begin{pmatrix} t_1 & 0 & 0 \\ 0 & t_1 & 0 \\ 0 & 0 & t_1 \end{pmatrix}$ and $B = \begin{pmatrix} t_2 & 0 & 0 \\ 0 & t_2 & 0 \\ 0 & 0 & t_2 \end{pmatrix}$. (b) The module of 0-th boundary groups $B_0, C = \begin{pmatrix} t_1 & 0 \\ 0 & t_1 \end{pmatrix}$ and $D = \begin{pmatrix} t_2 & 0 \\ 0 & t_2 \end{pmatrix}$. (c) The module of the 0-th homology groups $H_0$, it has one connected component in 0-th homology groups at grades except $(0,0)$ and $(1,1)$, and has two connected components at grade $(1,1)$.

Two graded modules $M, N$ are isomorphic if there exist two morphisms $f : M \rightarrow N$ and $g : N \rightarrow M$ such that $g \circ f$ and $f \circ g$ are identity maps. We denote it as $M \cong N$.

**Definition 7** (Shifted module). For a graded module $M$ and some $u \in \mathbb{Z}^d$, define a shifted graded module $M_{\rightarrow u}$ by setting $(M_{\rightarrow u})_v = M_{v-u}$ for each $v$.

**Definition 8** (Free module). We say a graded module is free if it is isomorphic to the direct sum of a collection of $R^I$, denoted as $\bigoplus I R^I$, where each $R^I = R_{-u_j}$ for some $u_j \in \mathbb{Z}^d$. Here $R$ is the special graded module in definition 3.

**Definition 9** (Homogeneous element). We say an element $m \in M$ is homogeneous if $m \in M_u$ for some $u \in \mathbb{Z}^d$. We denote $\text{gr}(m) = u$ as the grade of such homogeneous element. To emphasize
the grade of a homogeneous element, we also write \( m^{gr(m)} := m \).

A minimal generating set of a free module is called a basis. We usually further require that all the elements (generators) in a basis are homogeneous. For a free module \( F \cong \bigoplus_j R^j \) such a basis exists. Specifically, \( \{e_j : j = 1, 2, \cdots\} \) is a homogeneous basis of \( F \), where \( e_j \) indicates the multiplicative identity in \( R^j \). The generating set \( \{e_j : j = 1, 2, \cdots\} \) is often referred to as the standard basis of \( \bigoplus_j R^j = \langle e_j : j = 1, 2, \cdots \rangle \).

14.2 Presentations of persistence modules

**Definition 10** (Decomposition). For a finitely generated module \( M \), we call \( M \cong \bigoplus M^i \) a decomposition of \( M \) for some collection of modules \( \{M^i\} \). We say \( M \) is indecomposable if \( M \cong M^1 \oplus M^2 \implies M^1 = 0 \) or \( M^2 = 0 \) where \( 0 \) denotes a trivial module. By the Krull-Schmidt theorem [2], there exists an essentially unique (up to permutation and isomorphism) decomposition \( M \cong \bigoplus M^i \) with every \( M^i \) being indecomposable. We call it the total decomposition of \( M \).

For example, the free module \( R \) in Definition 3 is generated by \( \langle e_1^{(0,0)} \rangle \) and the free module \( R_{(0,1)} \oplus R_{(1,0)} \) is generated by \( \langle e_1^{(0,1)}, e_2^{(1,0)} \rangle \). A free module \( M \) generated by \( \langle e_j^u : j = 1, 2, \cdots \rangle \) has a (total) decomposition \( M \cong \bigoplus_j R_{u} \).

**Definition 11** (Isomorphic morphisms). Two morphisms \( f : M \to N \) and \( f' : M' \to N' \) are isomorphic, denoted as \( f \cong f' \), if there exist isomorphisms \( g : M \to M' \) and \( h : N \to N' \) such that the following diagram commutes:

\[
\begin{array}{ccc}
M & \xrightarrow{f} & N \\
\downarrow{g} & & \downarrow{h} \\
M' & \xrightarrow{f'} & N'
\end{array}
\]

Essentially, like isomorphic modules, two isomorphic morphisms can be considered the same. For two morphisms \( f_1 : M^1 \to N^1 \) and \( f_2 : M^2 \to N^2 \), there exists a canonical morphism \( g : M^1 \oplus M^2 \to N^1 \oplus N^2, g(m_1, m_2) = (f_1(m_1), f_2(m_2)) \), which is essentially uniquely determined by \( f_1 \) and \( f_2 \) and is denoted as \( f_1 \oplus f_2 \). A module is trivial if it has only the element \( 0 \) at every grade. We denote a trivial morphism by \( 0 : 0 \to 0 \). Analogous to the decomposition of a module, we can also define a decomposition of a morphism.

**Definition 12** (Morphism decomposition). A morphism \( f \) is indecomposable if \( f \cong f_1 \oplus f_2 \implies f_1 \) or \( f_2 \) is the trivial morphism \( 0 : 0 \to 0 \). We call \( f \cong \bigoplus f_i \) a decomposition of \( f \). If each \( f_i \) is indecomposable, we call it a total decomposition of \( f \).

Like decompositions of modules, the total decompositions of a morphism is also essentially unique.
Computational Topology for Data Analysis: Notes from book by T. K. Dey and Y. Wang

14.2.1 Presentation and its decomposition

To study total decompositions of persistence modules that are treated as graded modules, we draw upon the idea of presentations of graded modules and build a bridge between decompositions of persistence modules and corresponding presentations. Decompositions of presentations can be transformed to a matrix reduction problem with possibly nontrivial constrains which we will introduce in Section 14.3. We first state a result saying that there are one to one correspondences between persistence modules, presentations, and presentation matrices. Recall that, by assumption, all modules are finitely generated. A graded module hence a persistence module accommodates a description called its presentation that aids finding its decomposition. We remind the reader that a sequence of maps is exact if the image of one map equals the kernel of the next map.

Definition 13 (Presentation). A presentation of a graded module \(H\) is an exact sequence

\[ F^1 \xrightarrow{f} F^0 \xrightarrow{g} H \rightarrow 0 \]

where \(F^1, F^0\) are free.

We call \(f\) a presentation map. We say a graded module \(H\) is finitely presented if there exists a presentation of \(H\) with both \(F^1\) and \(F^0\) being finitely generated.

The exactness of the sequence implies that \(\text{im} f = \text{ker} g\) and \(\text{im} g = H\). The double arrows on the second map in the sequence signifies the surjection of \(g\). It follows that \(\text{coker} f \cong H\) and the presentation is determined by the presentation map \(f\).

Remark 1. Presentations of a given graded module are not unique. However, there exists an essentially unique (up to isomorphism) presentation \(f\) of a graded module in the sense that any presentation \(f'\) of that module can be written as \(f' \cong f \oplus f''\) with \(\text{coker} f'' = \{0\}\). We call this unique presentation the minimal presentation. See more details of the construction and properties of minimal presentation in [10].

Definition 14 (Presentation matrix). Given a presentation \(F^1 \xrightarrow{f} F^0 \rightarrow H\), fixed bases of \(F^1\) (relations) and \(F^0\) (generators) provide a matrix form \([f]\) of the presentation map \(f\), which we call a presentation matrix of \(H\). It has entries in \(R\). In the special case that \(H\) is a free module with \(F^1\) being a zero module, we define the presentation matrix \([f]\) of \(H\) to be a null column matrix with matrix size \(\ell \times 0\) for some \(\ell \in \mathbb{N}\).

In Figure 14.6, we illustrate the presentation matrix of the module \(H_0\) consisting of zero dimensional homology groups induced by the filtration shown in Figure 14.5. We will see later that, in this case, \(f\) equals the boundary morphism \(\partial_1 : C_1 \rightarrow C_0\) whose columns are edges and rows are vertices. For example, the red edge \(e_r\) whose grade is \((1, 1)\) has two boundary vertices \(v_b\), the blue vertex with grade \((0, 1)\) and \(v_r\), the red vertex with grade \((1, 0)\). To bring \(v_b\) to grade \((1, 1)\), we need to multiply by the polynomial \(t_1\). Similarly, to bring \(v_r\) to grade \((1, 1)\), we need to multiply by \(t_2\). The corresponding entries in the column of \(e_r\) are \(t_1\) and \(t_2\) respectively indicated by shaded boxes. Actual matrices are shown later in Example 1.

An important property of a persistence module \(H\) is that a decomposition of its presentation \(f\) corresponds to a decomposition of \(H\) itself. The decomposition of \(f\) can be computed by diagonalizing its presentation matrix \([f]\). Informally, a diagonalization of a matrix \(A\) is an equivalent
matrix $A'$ in the following form (see formal Definition 15 later):

$$
A' = \begin{bmatrix}
A_1 & 0 & \cdots & 0 \\
0 & A_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A_k
\end{bmatrix}
$$

All nonzero entries are in $A_i$'s and we write $A \simeq \bigoplus A_i$. It is not hard to see that for a map $f \simeq \bigoplus f_i$, there is a corresponding diagonalization $[f] \simeq \bigoplus [f_i]$. With these definitions and the fact that persistence modules are graded modules, we have the following theorem that motivates the decomposition algorithm (see proof in [10]).

**Theorem 1.** There are 1-1 correspondences between the following three structures arising from a minimal presentation map $f : F^1 \to F^0$ of a graded module $H$, and its presentation matrix $[f]$:

1. A decomposition of the persistence module $H \simeq \bigoplus H^i$;
2. A decomposition of the presentation map $f \simeq \bigoplus f_i$;
3. A diagonalization of the presentation matrix $[f] \simeq \bigoplus [f_i]$.

**Remark 2.** From Theorem 1, we can see that there exist an essentially unique total decomposition of a presentation map and an essentially unique total diagonalization of the presentation matrix of $H$ which correspond to an essentially unique total decomposition of $H$ (up to permutation, isomorphism, and trivial summands). In practice, we might be given a presentation which is not necessarily minimal. One way to handle this case is to compute the minimal presentation of the given presentation first. For 2-parameter modules, this can be done by an algorithm presented in [22]. The other choice is to compute the decomposition of the given presentation directly, which is sufficient to get the decomposition of the module thanks to the following proposition.

Figure 14.6: The presentation matrix of the module $H_0$ consisting of zero dimensional homology groups for the example in Figure 14.5. The boxes in the matrix containing non-zero entries are shaded.
Proposition 2. Let \( f \) be any presentation (not necessarily minimal) of a graded module \( H \). The following statements hold:

1. for any decomposition of \( H \simeq \bigoplus H^i \), there exists a decomposition of \( f \simeq \oplus f_i \) such that \( \operatorname{coker} f_i = H^i \), \( \forall i; \)

2. the total decomposition of \( H \) follows from the total decomposition of \( f \).

Remark 3. By Remark 1, any presentation \( f \) can be written as \( f \simeq f^* \oplus f' \) with \( f^* \) being the minimal presentation and \( \operatorname{coker} f' = 0 \). Furthermore, \( f' \) can be written as \( f' \simeq g \oplus h \) where \( g \) is an identity map and \( h \) is a zero map. The corresponding matrix form is \( [f] \simeq [f^*] \oplus [g] \oplus [h] \) with \([g] \) being an identity submatrix and \([h] \) being an empty matrix representing a collection of zero columns. Therefore, one can easily read these trivial parts from the result of matrix diagonalization. See the following diagram for an illustration.

\[
\begin{pmatrix}
    f^* & g & h \\
    [f^*] & 1 & 1 \\
    1 & 1 & 1
\end{pmatrix}
\]

14.3 Presentation matrix: diagonalization and simplification

Our aim is to compute a total diagonalization of a presentation matrix over \( \mathbb{Z}_2 \). Here we formally define some notations used in the diagonalization. All modules are assumed to be finitely presented and we take \( k = \mathbb{Z}_2 \) for simplicity though the method can be generalized for any finite field (Exercise 9). We have observed that a total decomposition of a module can be achieved by computing a total decomposition of its presentation \( f \). This in turn requires a total diagonalization of the presentation matrix \([f]\). Here we formally define some notations about the diagonalization.

Given an \( \ell \times m \) matrix \( A = [A_{i,j}] \), with row indices \( \text{Row}(A) = \{\ell\} := \{1, 2, \ldots, \ell\} \) and column indices \( \text{Col}(A) = \{m\} := \{1, 2, \ldots, m\} \), we define an index block \( B \) of \( A \) as a pair \( [\text{Row}(B), \text{Col}(B)] \) with \( \text{Row}(B) \subseteq \text{Row}(A), \text{Col}(B) \subseteq \text{Col}(A) \). We say an index pair \((i, j)\) is in \( B \) if \( i \in \text{Row}(B) \) and \( j \in \text{Col}(B) \), denoted as \((i, j) \in B \). We denote a block of \( A \) on \( B \) as the matrix restricted to the index block \( B \), i.e. \([A_{i,j}]_{(i,j)\in B} \), denoted as \( A_{B} \). We call \( B \) the index of the block \( A_{B} \). We abuse the notations \( \text{Row}(A_{B}) := \text{Row}(B) \) and \( \text{Col}(A_{B}) := \text{Col}(B) \). For example, the \( i \)th row \( r_i = A_{i,*} = A_{[i,\text{Col}(A)]} \) and the \( j \)th column \( c_j = A_{*,j} = A_{[\text{Row}(A),j]} \) are blocks with indices \([i,\text{Col}(A)]\) and \([\text{Row}(A),j]\) respectively. Specifically, \([\varnothing, \varnothing]\) represents an index block of a single column \( j \) and \([i, \varnothing]\) represents an index block of a single row \( i \). We call \([\varnothing, \varnothing]\) the empty index block.

A matrix can have multiple equivalent forms for the same morphism they represent. We use \( A' \sim A \) to denote the equivalence of matrices. One fact about equivalent matrices is that they can be obtained from one another by row and column operations introduced later (Chapter 5 in [8]).
Definition 15 (Diagonalization). A matrix $A'$ ~ $A$ is called a diagonalization of $A$ with a set of nonempty disjoint index blocks $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$ if rows and columns of $A$ are partitioned into these blocks, i.e., $\text{Row}(A) = \bigsqcup_i \text{Row}(B_i)$ and $\text{Col}(A) = \bigsqcup_i \text{Col}(B_i)$, and all the nonzero entries of $A'$ have indices in some $B_i$ ($\bigsqcup_i$ denotes disjoint union). We write $A' = \bigoplus_{B_i \in \mathcal{B}} A'_{|B_i}$. We say $A' = \bigoplus_{B_i \in \mathcal{B}} A'_{|B_i}$ is total if no block in this diagonalization can be diagonalized further into smaller nonempty blocks. That means, for each block $A'_{|B_i}$, there is no nontrivial diagonalization. Specifically, when $A$ is a null column matrix (the presentation matrix of a free module), we say $A$ is itself a total diagonalization with index blocks $\{([i], \emptyset) \mid i \in \text{Row}(A)\}$.

Note that each nonempty matrix $A$ has a trivial diagonalization with the set of index blocks being the singleton $\{((\text{Row}(A), \text{Col}(A)))\}$. Guaranteed by Krull-Schmidt theorem [2], all total diagonalizations are unique up to permutations of their rows and columns, and equivalent transformation within each block. The total diagonalization of $A$ is denoted generically as $A^\ast$. All total diagonalizations of $A$ have the same set of index blocks unique up to permutations of rows and columns. See Figure 14.7 for an illustration of a diagonalized matrix.

![Figure 14.7: (left) A nontrivial diagonalization where the locations of non-zero entries are colored and the color for all such entries in the same block are the same. (right) The same matrix with permutation of columns and rows to bring entries of a block in adjacent locations, the three index blocks are: $((1, 4, 6)(1, 3)), ((2, 3)(2, 4))$, and $((5)(5))$.](image)

14.3.1 Simplification

First we want to transform the diagonalization problem to an equivalent problem that involves matrices with a simpler form. The idea is to simplify the presentation matrix to have entries only in $k$ which is taken as $\mathbb{Z}_2$. There is a correspondence between diagonalizations of the original presentation matrix and certain constrained diagonalizations of the corresponding transformed matrix.

We first make some observations about the homogeneous property of presentation maps and presentation matrices. Equivalent matrices actually represent isomorphic presentations $f' \simeq f$ that admit commutative diagram,
where $h^1$ and $h^0$ are endomorphisms on $F^1$ and $F^0$ respectively. The endomorphisms are realized by basis changes between corresponding presentation matrices $[f] = [f']$. Since all morphisms between graded modules are required to be homogeneous (preserve grades) by definition, we can use homogeneous bases (all the basis elements chosen are homogeneous elements\(^2\)) for $F^0$ and $F^1$ to represent matrices. Let $F^0 = \langle g_1, \cdots, g_\ell \rangle$ and $F^1 = \langle s_1, \cdots, s_m \rangle$ where $g_i$ and $s_i$ are homogeneous elements for every $i$. With this choice, we can consider only equivalent presentation matrices under homogeneous basis changes. Each entry $[f]_{i,j}$ is also homogeneous. That means, $[f]_{i,j} = t_{i1}^u t_{i2}^u \cdots t_{id}^u$ where $(u_1, u_2, \cdots, u_d) = \text{gr}(s_j) - \text{gr}(g_i)$. Writing $u = (u_1, u_2, \cdots, u_d)$ and $t^u = t_{i1}^u t_{i2}^u \cdots t_{id}^u$, we get $[f]_{i,j} = t^u$ where $u = \text{gr}(s_j) - \text{gr}(g_i)$ called the grade of $[f]_{i,j}$. We call such presentation matrix homogeneous presentation matrix.

For example, given $F^0 = \langle g_1^{(1,1)}, g_2^{(2,2)} \rangle$, the basis change $g_2^{(2,2)} \leftarrow g_2^{(2,2)} + g_1^{(1,1)}$ is not homogeneous since $g_2^{(2,2)} + g_1^{(1,1)}$ is not a homogeneous element. However, $g_2^{(2,2)} \leftarrow g_2^{(2,2)} + t^{(1,1)} g_1^{(1,1)}$ is a homogeneous change with $\text{gr}(g_2^{(2,2)} + t^{(1,1)} g_1^{(1,1)}) = \text{gr}(g_2^{(2,2)}) = (2, 2)$, which results in a new homogeneous basis, $(g_1^{(1,1)}, g_2^{(2,2)} + t^{(1,1)} g_1^{(1,1)})$. Homogeneous basis changes always result in homogeneous bases.

Let $[f]$ be a homogeneous presentation matrix of $f : F^1 \to F^0$ with bases $F^0 = \langle g_1, \cdots, g_\ell \rangle$ and $F^1 = \langle s_1, \cdots, s_m \rangle$. We extend the notation of grading to every row $r_i$ and every column $c_j$ from the basis elements $g_i$ and $s_j$ they represent respectively, that is, $\text{gr}(r_i) := \text{gr}(g_i)$ and $\text{gr}(c_j) := \text{gr}(s_j)$. We define a strict partial order $<_{\text{gr}}$ on rows $[r_i]$ by asserting $r_i <_{\text{gr}} r_j$ if and only if $\text{gr}(r_i) < \text{gr}(r_j)$. Similarly, we define a strict partial order on columns $[c_j]$. For such a homogeneous presentation matrix $[f]$, we aim to diagonalize it totally by homogeneous change of basis while trying to zero out entries by column and row operations that include additions and scalar multiplication of columns and rows as done in well known Gaussian elimination. We have the following observations:

1. $\text{gr}([f]_{i,j}) = \text{gr}(c_j) - \text{gr}(r_i)$
2. a nonzero entry $[f]_{i,j}$ can only be zeroed out by column operations from columns $c_k <_{\text{gr}} c_j$ or by row operations from rows $r_\ell >_{\text{gr}} r_i$.

Observation (2) indicates which subset of column and row operations is sufficient to zero out the entry $[f]_{i,j}$. We restate the diagonalization problem as follows:

Given an $n \times m$ homogeneous presentation matrix $A = [f]$ consisting of entries in $k[t_1, \cdots, t_\ell]$ with grading on rows and columns, find a total diagonalization of $A$ under the following admissible row and column operations:

- multiply a row or column by nonzero $\alpha \in k$; (For $k = \mathbb{Z}_2$, we can ignore these operations),
- for two rows $r_i, r_j$ with $j \neq i, r_j <_{\text{gr}} r_i$, set $r_j \leftarrow r_j + t^u \cdot r_i$ where $u = \text{gr}(r_i) - \text{gr}(r_j)$,
- for two columns $c_i, c_j$ with $j \neq i, c_i <_{\text{gr}} c_j$, set $c_j \leftarrow c_j + t^v \cdot c_i$ where $v = \text{gr}(c_j) - \text{gr}(c_i)$.

The above operations realize all possible homogeneous basis changes. That means, any homogeneous presentation matrix can be realized by a combination of the above operations.\(^2\)

\(^2\)Recall that an element $m \in M$ is homogeneous with grade $\text{gr}(m) = u$ for some $u \in \mathbb{Z}^d$ if $m \in M_u$.\(^2\)
In fact, the values of nonzero entries in the matrix are redundant under the homogeneous property $\text{gr}(A_{i,j}) = \text{gr}(c_i) - \text{gr}(r_j)$ given by observation (1). So, we can further simplify the matrix by replacing all the nonzero entries with their $k$-coefficients. For example, we can replace $2t^6$ with $2$. What really matters are the partial orders defined by the grading of rows and columns. With our assumption of $k = \mathbb{Z}_2$, all nonzero entries are replaced with $1$. Based on above observations, we further simplify the diagonalization problem to be the one as follows.

Given a $k$-valued matrix $A$ with a partial order on rows and columns, find a total diagonalization $A^* \sim A$ with the following admissible operations:

- multiply a row or column by nonzero $\alpha \in k$; (For $k = \mathbb{Z}_2$, we can ignore these operations).
- Adding $c_i$ to $c_j$ only if $j \neq i$ and $\text{gr}(c_i) < \text{gr}(c_j)$; denoted as $c_i \rightarrow c_j$.
- Adding $r_k$ to $r_l$ only if $l \neq k$ and $\text{gr}(r_l) < \text{gr}(r_k)$; denoted as $r_k \rightarrow r_l$.

The assumption of $k = \mathbb{Z}_2$ allows us to ignore the first set of multiplication operations on the binary matrix obtained after transformation. We denote the set of all admissible column and row operations as

$$\text{Colop} = \{(i, j) \mid c_i \rightarrow c_j \text{ is an admissible column operation}\}$$
$$\text{Rowop} = \{(k, l) \mid r_k \rightarrow r_l \text{ is an admissible row operation}\}$$

Under the assumption that no two columns nor rows have same grades, $\text{Colop}$ and $\text{Rowop}$ are closed under transitive relation.

**Proposition 3.** If $(i, j), (j, k) \in \text{Colop (Rowop)}$ then $(i, k) \in \text{Colop (Rowop)}$.

Given a solution of the diagonalization problem in the simplified form, one can reconstruct a solution of the original problem on the presentation matrix by reversing the above process of simplification. We will illustrate it by running the algorithm on the working example in Figure 14.5 at the end of this section. The matrix reduction we employ for diagonalization may be viewed as a generalized matrix reduction because the matrix is reduced under constrained operations $\text{Colop}$ and $\text{Rowop}$ which might be a nontrivial subset of all basic operations.

**Remark 4.** There are two extreme but trivial cases: (i) there are no $\text{gr}$-comparable pair of rows and columns. In this case, $\text{Colop} = \text{Rowop} = \emptyset$ and the original matrix is a trivial solution. (ii) All pairs of rows and all pairs of columns are $\text{gr}$-comparable. Or equivalently, both $\text{Colop}$ and $\text{Rowop}$ are totally ordered. In this case, one can apply traditional matrix reduction algorithm to reduce the matrix to a diagonal matrix with all nonzero blocks being $1 \times 1$ minors. This is also the case for 1-parameter persistence module if one further applies row reduction after column reduction. Note that row reductions are not necessary for reading out persistence information because it essentially does not change the persistence information. However, in multiparameter cases, both column and row reductions are necessary to obtain a diagonalization from which the persistence information can be read. From this viewpoint, the algorithm we present can be thought of as a generalization of the traditional persistence algorithm.
Example 1. Consider our working example in Figure 14.5. One can see later in Section 14.5 (Case 1) that the presentation matrix of this example can be chosen to be the same as the matrix of the boundary morphism \( \partial_1 : C_1 \to C_0 \). With fixed bases \( C_0 = \langle v_b(0, 1), v_r(1, 0), v_g(1, 1) \rangle \) and \( C_1 = \langle e_r(1, 1), e_p(1, 2), e_s(2, 1) \rangle \), this presentation matrix \([\partial_1]\) and the corresponding binary matrix \(A\) can be written as follows (recall that superscripts indicate the grades):

\[
\begin{bmatrix}
\partial_{1} \\
v_b^{(0, 1)} \\
v_r^{(1, 0)} \\
v_g^{(1, 1)}
\end{bmatrix}
\begin{bmatrix}
e_r^{(1, 1)} & e_b^{(1, 2)} & e_g^{(2, 1)}
\end{bmatrix}
\begin{bmatrix}
t^{(1, 0)} & 0 & t^{(1, 1)}
0 & t^{(0, 1)} & t^{(0, 0)}
\end{bmatrix}
\rightarrow
\begin{bmatrix}
A \\
r_1^{(0, 1)} \\
r_2^{(1, 0)} \\
r_3^{(1, 1)}
\end{bmatrix}
\begin{bmatrix}
c_1^{(1, 1)} & c_2^{(1, 2)} & c_3^{(2, 1)}
1 & 1 & 0
1 & 0 & 1
0 & 1 & 1
\end{bmatrix}
\]

Four admissible operations are: \( r_3 \to r_1, r_3 \to r_2, c_1 \to c_2, c_1 \to c_3 \). Figure 14.8 shows the persistence module \( H_0 \) whose presentation matrix is \([\partial_1]\).

14.4 Total diagonalization algorithm

Assume that no two columns nor rows have the same grades. Without this assumption, the problem of total diagonalization becomes more complicated. At this point, we do not know how to extend the algorithm to overcome this limitation. However, the algorithm introduced below can still compute a correct diagonalization (not necessarily total) by applying the trick of adding small enough perturbations to tied grades (considering \( \mathbb{Z}^d \subseteq \mathbb{R}^d \)) to reduce the case to the one satisfying our assumption. Furthermore, this diagonalization in fact coincides with a total diagonalization of some persistence module which is arbitrarily close to the original persistence module under a well-known metric called interleaving distance which we discuss in the next chapter. In practice, the persistence module usually arises from a simplicial filtration as shown in our working example. The assumption of distinct grading of the columns and rows is automatically satisfied if at most one simplex is introduced at each grade in the filtration.

Let \( A \) be the presentation matrix whose total diagonalization we are seeking for. We order the rows and columns of the matrix \( A \) according to any order that extends the partial order on the grades to a total order, e.g., dictionary order. We fix the indices \( \text{Row}(A) = \{1, 2, \cdots, \ell\} \)
and \( \text{Col}(A) = \{1, 2, \cdots , m\} \) according to this order. With this ordering, observe that, for each admissible column operation \( c_i \rightarrow c_j \), we have \( i < j \), and for each admissible row operation \( r_l \rightarrow r_k \), we have \( l > k \).

For any column \( c_t \), let \( A_{\leq t} := A|_{C} \) denote the left submatrix on \( C = \{\text{Row}(A), \{j \in \text{Col}(A) | j \leq t\}\} \) and \( A_{< t} \) denote its stricter version obtained by excluding column \( c_t \) from \( A_{\leq t} \). Our algorithm starts with the finest decomposition that puts every free module given by each generator (rows) into a separate block and then combine them incrementally as we process the relations (columns). The main idea of our algorithm is presented in Algorithm 1: TorDiagonalize which runs as follows (see Figure 14.9 for an illustration):

1. **Initialization:** Initialize the collection of index blocks \( \mathcal{B}^{(0)} := \{B^{(0)}_i := [[i], \varnothing] | i \in \text{Row}(A)\} \), for the total diagonalization of the null column matrix \( A_{\leq 0} \).

2. **Main iteration:** Process \( A \) from left to right incrementally by introducing a column \( c_t \) and considering left submatrices \( A_{\leq t} \) for \( t = 1, 2, \cdots , m \). We update and maintain the collection of index blocks \( \mathcal{B}^{(i)} \leftarrow \{B^{(i)}_j\} \) for the current submatrix \( A_{\leq t} \) in each iteration by using column and block updates stated below. Here we use upper index \( (\cdot)^{(i)} \) to emphasize the iteration \( t \).

2(a). **Sub-column update:** Partition the column \( c_t \) into sub-columns

\[
\begin{align*}
c_t|_{\text{Row}(B^{(t-1)}_j)} & := A|_{\text{Row}(B^{(t-1)}_j), \{t\}}, \\
n & = B^{(t-1)}_j,
\end{align*}
\]

one for the set of rows \( \text{Row}(B^{(t-1)}_j) \) for each block from the previous iteration. We process each such sub-column \( c_t|_{\text{Row}(B^{(t-1)}_j)} \) one by one, checking whether there exists a sequence of admissible operations that are able to reduce the sub-column to zero while preserving the prior as defined below.
Definition 16. We say a prior with respect to a sub-column $c_i|_{\text{Row} B_j^{t-1}}$ is the left submatrix $A_{< t}$ and sub-columns $c_i|_{\text{Row} B_j^{t-1}}$ for all $j < i$.

Prior preservation means that the operations together change neither $A_{< t}$ nor other sub-columns $c_i|_{\text{Row} B_j^{t-1}}$ for every $j < i$. If such operations exist, we apply them on the current $A$ to get an equivalent matrix with the sub-column $c_i|_{\text{Row} B_j^{t-1}}$ being zeroed out and we set $B_i^{(t)} \leftarrow B_i^{(t-1)}$. Otherwise, we leave the matrix $A$ unchanged and add the column index $t$ to those of $B_i^{(t-1)}$, i.e., we set $B_i^{(t)} \leftarrow [\text{Row}(B_i^{(t-1)}), \text{Col}(B_i^{(t-1)}) \cup \{t\}]$. After processing every sub-column $c_i|_{\text{Row} B_j^{t-1}}$ one by one, all index blocks $B_i^{(t)}$ containing column index $t$ are merged into one single index block. At the end of iteration $t$, we get an equivalent matrix $A$ with $A_{< t}$ being totally diagonalized with index blocks $\mathcal{B}^{(t)}$.

2(b). Block reduce: To update the entries of each sub-column of $c_i$ described in 2(a), we propose a block reduction algorithm Algorithm 3:BlockReduce to compute the correct entries. Given $T := [\text{Row}(B_i^{(t-1)}), (\text{Col}(A_{< t}) \setminus \text{Col}(B_i^{(t-1)}))]$, this routine checks whether the block $T$ can be zeroed out by some collection of admissible operations. If so, $c_i$ does not join the block $B_i^{(t)}$ and $A$ is updated with these operations.

For two index blocks $B_1, B_2$, we denote the merging $B_1 \oplus B_2$ of these two index blocks as an index block $[\text{Row}(B_1) \cup \text{Row}(B_2), \text{Col}(B_1) \cup \text{Col}(B_2)]$. In the following algorithm, we treat the given matrix $A$ to be a global variable which can be visited and modified anywhere by every subroutines called. Consequently, every time we update values on $A$ by some operations, these operations are applied to the latest $A$.

The outer loop is the incremental step for main iteration introducing a new column $c_i$ which updates the diagonalization of $A_{< t}$ from the last iteration. The inner loop corresponds to block updates which checks the intersection of the current column and the rows of each previous block one by one.

Remark 5. The algorithm TôtDiagonalize does not require the input presentation matrix to be minimal. As indicated in Remark 3, the trivial parts result in either identity blocks or single column blocks like $[\emptyset, [j]]$. Such a single column block corresponds to a zero morphism and is not merged with any other blocks. Therefore, $c_j$ is a zero column. For a single row block $[[i], \emptyset]$ which is not merged with any other blocks, $r_i$ is a zero row vector. It represents a free indecomposable submodule in the total decomposition of the input persistence module.

We first prove the correctness of TôtDiagonalize assuming that BlockReduce routine works as claimed, namely, it checks if a sub-column of the current column $c_i$ can be zeroed out while preserving the prior, that is, without changing the left submatrix from the previous iteration and also the other sub-columns of $c_i$ that have already been zeroed out.

Proposition 4. At the end of each iteration $t$, $A_{< t}$ is a total diagonalization.

Proof. We prove it by induction on $t$. For the base case $t = 0$, it follows trivially by definition. Now assume $A^{(t-1)}$ is the matrix we get at the end of iteration $(t - 1)$ with $A^{(t-1)}_{< t-1}$ being totally diagonalized. That means, $A^{(t-1)}_{< t-1} = A^{*}_{< t-1}$ where $A = A^{(0)}$ is the original given matrix. For
Algorithm 1 \textsc{TotDiagonalize}(A)

\textbf{Input:} \\
$A$ = input matrix treated as a global variable whose columns and rows are totally ordered respecting some fixed partial order given by the grading.

\textbf{Output:} \\
A total diagonalization $A^*$ with index blocks $B^*$

1: $B^{(0)} \leftarrow \{B_i^{(0)} := [\{i\}, \emptyset] \mid i \in \text{Row}(A)\}$
2: for $t \leftarrow 1$ to $m := |\text{Col}(A)|$ do
3: 
4: for each $B_i^{(t-1)} \in B^{(t-1)}$ do
5: 
6: \textbf{if } \text{BlockReduce} $(T) == \text{false}$ \textbf{then}
7: 
8: \textbf{else}
9: 
10: \# $A$ and $c_t$ are updated in BlockReduce when it return true \\n11: 
12: \end if \\n13: 
14: \end for \\n15: \textbf{end for} \\n16: $B^{(t)} \leftarrow \{B_i^{(t)}\}$ with all $B_i^{(t)}$ containing $t$ merged as one block.
17: 
18: \textbf{end for} \\n19: \textbf{end for} \\n20: \textbf{Return} $(A, B^{(m)})$

contradiction, assume at the end of iteration $t$, the matrix we get, $A^{(t)}$, has left submatrix $A_{st}^{(t)}$ which is not a totally diagonalized. That means, some index block $B \in B^{(t)}$ can be decomposed further. Observe that such $B$ must contain $t$ because all other index blocks (not containing $t$) in $B^{(t)}$ are also in $B^{(t-1)}$ which cannot be decomposed further by our inductive assumption. We denote this index block containing $t$ as $B_t$. Let $A'$ be the equivalent matrix of $A^{(t)}$ such that $A'_{st}$ is a total diagonalization with index blocks $B'$. Let $F$ be an equivalent transformation from $A^{(t)}$ to $A'$, which decomposes $B_t$ into at least two distinct index blocks of $B'$, say $B_0, B_1, \cdots$. Only one of them contains $t$, say $B_0$. Then $B_1$ consists of only indices that are from $A_{st-1}$, which means $B_1$ equals some index block $B_1 \in B^{(t-1)}$. Therefore, the transformation $F$ gives a sequence of admissible operations which can reduce the sub-column $c_t|\text{Row}(B_1)$ to zero in $A^{(t)}$. Starting with this sequence of admissible operations, we construct another sequence of admissible operations which further keeps $A_{st-1}^{(t)}$ unchanged to reach the contradiction. Note that $A_{st-1}^{(t)} = A_{st-1}^{(t-1)}$.

Observe that all index blocks of $B'$ other than $B_0$ are also index blocks in $B^{(t-1)}$, i.e. $B' \cap \{B_{0}\} \subseteq B^{(t-1)}$. For $B_{0}$, it can be written as $B_{0} = \bigoplus_{B_j \in B\cap\{B_{0}\}} B_j \oplus [\{i\}, \emptyset]$. Let $B_{0}$ be the merge of index blocks that are in $A^{(t-1)}$ and also in $A'$ and $B_{0}$ be the merge of the rest of the index blocks of $A^{(t-1)}$, i.e., $B_{a} = \bigoplus_{B_j \in B'\cap\{B_{0}\}} B_j$ and $B_{b} = \bigoplus_{B_j \in B\cap\{B_{0}\}} B_j$. Then $B_a$ and $B_b$ can be viewed as a coarser decomposition on $A_{st-1}^{(t)}$ and also on $A'_{st-1}$. By taking restrictions, we have $A'|_{B_{0}} \sim A_{st-1}|_{B_{0}}$ with equivalent transformation $F_a$ and $A'|_{B_{0}} \sim A_{st-1}|_{B_{0}}$ with equivalent transformation $F_b$. Then $F_a$ gives a sequence of admissible operations with indices in $B_{a}$ and $F_b$ gives a sequence of admissible operations with indices in $B_{b}$. By applying these operations on $A'$, we can transform
A'_{s,t-1} \text{ to } A_{s,t-1}^{(t-1)} \text{ with sub-column } [\text{Row}(A) \setminus \text{Row}(B_0), \{t\}] \text{ unchanged, which consists of the sub-columns that have already been reduced to zero. Combining all admissible operations from the three transformations } F, F_a \text{ and } F_b \text{ together, we get a sequence of admissible operations that reduce sub-column } [\text{Row}(B_i), \{t\}] \text{ to zero without changing } A_{s,t}^{(t)} \text{ and also those sub-columns which have already been reduced. But, then BlockReduce would have returned ‘true’ signaling that } B_i \text{ should not be merged with any other block required to form the block } B_i \text{ reaching a contradiction. □}

Now we design the BlockReduce subroutine as required. With the requirement of prior preservation, observe that reducing the sub-column } c_{l|\text{Row}B} \text{ for some } B \in \mathcal{B}^{(t-1)} \text{ is the same as reducing } T = [\text{Row}(B), (\text{Col}(A) \setminus \text{Col}(B))] \text{ called the target block (see Figure 14.9 on right). The main idea of BlockReduce is to consider a specific subset of admissible operations called independent operations. Within } A_{s,t} \text{, these operations only change entries in } T \text{ and this change is independent of their order of application. The BlockReduce subroutine is designed to search for a sequence of admissible operations within this subset and reduce } T \text{ with it, if it exists. Clearly, the prior is preserved with these operations. The only thing we need to ensure is that searching within the set of independent operations is sufficient. That means, if there exists a sequence of admissible operations that can reduce } T \text{ to } 0 \text{ and meanwhile preserves the prior, then we can always find one such sequence with only independent operations. This is what we show next.}

Consider the following matrices for each admissible operation. For each admissible column operation } c_i \rightarrow c_j \text{, let}

\[ Y^{i,j} := A \cdot [\delta_{i,j}] \]

where } [\delta_{i,j}] \text{ is the } m \times m \text{ square matrix with only one non-zero entry at } (i, j). \text{ Observe that } A \cdot [\delta_{i,j}] \text{ is a matrix with the only nonzero column at } j \text{ with entries copied from } c_i \text{ in } A. \text{ Similarly, for each admissible row operation } r_l \rightarrow r_k \text{, let } [\delta_{k,l}] \text{ be the } \ell \times \ell \text{ matrix with only non-zero entry at } (k, l), \text{ let}

\[ X^{k,l} := [\delta_{k,l}] \cdot A \]

Application of a column operation } c_i \rightarrow c_j \text{ can be viewed as updating } A \text{ to } A \cdot (I + [\delta_{i,j}]) = A + Y^{i,j}. \text{ Similar observation holds for row operations as well. For a target block } T = [\text{Row}(B), (\text{Col}(A) \setminus \text{Col}(B))] \text{ defined on some } B \in \mathcal{B}^{(t-1)}, \text{ we say an admissible column (row) operation, } c_i \rightarrow c_j \text{ (} r_l \rightarrow r_k \text{ resp.) is independent on } T \text{ if } i \notin \text{Col}(T), j \in \text{Col}(T) \text{ (} l \notin \text{Row}(T), k \in \text{Row}(T) \text{ resp.). Briefly, we just call such operations independent operations if } T \text{ is clear from the context.}

We have two observations about independent operations that are important. The first one follows from the definition that } T = [\text{Row}(B), (\text{Col}(A) \setminus \text{Col}(B))].

**Observation 1.** Within } A_{s,t} \text{, an independent column or row operation only changes entries on } T.

**Observation 2.** For any independent column operation } c_i \rightarrow c_j \text{ and row operation } r_l \rightarrow r_k \text{, we have } [\delta_{k,l}] \cdot A \cdot [\delta_{i,j}] = 0. \text{ Or, equivalently}

\[ (I + [\delta_{k,l}]) \cdot A \cdot (I + [\delta_{i,j}]) = A + [\delta_{k,l}]A + A[\delta_{i,j}] = A + X^{k,l} + Y^{i,j} \quad (14.1) \]

**Proof.** } [\delta_{k,l}] \cdot A \cdot [\delta_{i,j}] = A_{t,l} [\delta_{k,j}] \text{ (see Fig 14.10 for an illustration). By definitions of independence and } T, \text{ we have } l \notin \text{Row}(B), i \in \text{Col}(B). \text{ That means they are row index and column index from...}
different blocks. Therefore, $A_{l,i} = 0$. □

The following proposition reveals why we are after the independent operations.

**Proposition 5.** The target block $A|_T$ can be reduced to 0 while preserving the prior if and only if $A|_T$ can be written as a linear combination of independent operations. That is,

$$A|_T = \sum_{l \in \text{Row}(T)} \alpha_{k,l} X_{k,l}|_T + \sum_{j \in \text{Col}(T)} \beta_{i,j} Y_{i,j}|_T$$

where $\alpha_{k,l}'s$ and $\beta_{i,j}'s$ are coefficient in $k = \mathbb{Z}_2$.

The full proof can be seen in [10]. Here, we give some intuitive explanation. Reducing the target block $A|_T$ to 0 is equivalent to finding matrices $P$ and $Q$ encoding sequences of admissible row operations and admissible column operations respectively so that $PAQ|_T = 0$. For $\Leftarrow$ direction, we can build $P = I + \sum_{l \in \text{Row}(T)} \alpha_{k,l}[\delta_{k,l}]$ and $Q = I + \sum_{j \in \text{Col}(T)} \beta_{i,j}[\delta_{i,j}]$ with binary coefficients $\alpha_{k,l}'s$ and $\beta_{i,j}'s$ given in Eqn. (5). Then using Observations 1 and 2, one can show $PAQ$ indeed reduces $A|_T$ to 0 with the prior being preserved. This provides the proof for the ‘if’ direction.

For ‘only if’ direction, as long as we show that the existence of a transformation reducing $A|_T$ to 0 implies the existence of a transformation reducing $A|_T$ to 0 by independent operations, we are done. This is formally proved in [10].

We can view $A|_T, Y_{i,j}|_T, X_{i,j}|_T$ as binary vectors in the same $|T|$-dimensional space. Proposition 5 tells us that it is sufficient to check if $A|_T$ can be a linear combination of the vectors corresponding to a set of independent operations. So, we first linearize each of the matrices $Y_{i,j}|_T's$, $X_{i,j}|_T's$, and $A|_T$ to a column vector as described later (see Figure 14.11). Then, we check if $A|_T$ is in the span of $Y_{i,j}|_T's$ and $X_{i,j}|_T's$. This is done by collecting all vectors $X_{i,j}|_T's$ and $Y_{i,j}|_T's$ into a matrix $S$ called the source matrix (Figure 14.11(right)) and then reducing the vector $c := A|_T$ with $S$ by some standard matrix reduction algorithm with left-to-right column additions, which we have seen before for computing persistence. This routine is presented in Algorithm 2:ColReduce $(S, c)$ which reduces the column $c$ w.r.t. the input matrix $S$ by reducing the matrix $[S|c]$ altogether by MatPersistence that we described previously (Topic 5).

If $c = A|_T$ can be reduced to 0, we apply the corresponding independent operations to update $A$. Observe that all column operations used in reducing $A|_T$ together only change the sub-column
computational topology for data analysis: notes from book by T. K. Dey and Y. Wang

c.t row \( (B) \) while row operations may change \( A \) to the right of the column \( t \). We say this procedure reduces \( c \) with \( S \).

**Algorithm 2** \( \text{ColReduce}(S, c) \)

**Input:**
- Source matrix \( S \) and target column \( c \) to reduce

**Output:**
- Reduced column \( c \) with \( S \)

1. \( S' \leftarrow [S|c] \)
2. Call \( \text{MatPersistence}(S') \);
3. \textbf{return} \( c \) along with indices of columns in \( S \) used for reduction of \( c \)

**Fact 1.** There exists a set of column operations adding a column only to its right such that the matrix \( [S|c] \) is reduced to \( [S'|0] \) if and only if \( \text{ColReduce}(S, c) \) returns a zero vector.

**Proposition 6.** The target block on \( T \) can be reduced to zero in \( A \) while preserving the prior if and only if \( \text{BlockReduce}(T) \) returns true.

**Time complexity.** First we analyze the time complexity of \( \text{TotDiagonalize} \) assuming that the input matrix has size \( \ell \times m \). Clearly, \( \max(\ell, m) = O(N) \) where \( N \) is the total number of generators.
Algorithm 3 BLOCKREDUCE(T)

Input:
Index of target block T to be reduced; Given matrix A is assumed to be a global variable

Output:
A boolean to indicate whether A|\_T can be reduced and reduced block A|\_T if possible.

1: Compute c := Lin(A|\_T) and initialize empty matrix S
2: for each admissible column operation c_i \rightarrow c_j with i \notin Col(T), j \in Col(T) do
   3: compute Y^i:j|\_T := (A \cdot [\delta_{i,j}])|\_T and y^{i,j} := Lin(Y^i:j|\_T); update S ← [S|y^{i,j}]
4: end for
5: for each admissible row operation r_l \rightarrow r_k with l \notin Row(T), k \in Row(T) do
   6: compute X^{k,l}|\_T := ([\delta_{k,l}] \cdot A)|\_T and x^{k,l} := Lin(X^{k,l}|\_T); update S ← [S|x^{k,l}]
7: end for
8: ColReduce(S, c) returns indices of y^{i,j}'s and x^{k,l}'s used to reduce c (if possible)
9: For every returned index of y^{i,j} or x^{k,l} apply c_i \rightarrow c_j or r_l \rightarrow r_k to transform A
10: return A|\_T == 0

and relations. For each of O(N) columns, we attempt to zero out every sub-column with row indices coinciding with each block B of the previously determined O(N) blocks. Let B has N_B rows. Then, the block T has N_B rows and O(N) columns.

To zero out a sub-column, we create a source matrix out of T which has size O(N(N_B) × O(N^2)) because each of O(\binom{N}{2}) possible operations is converted to a column of size O(NN_B) in the source matrix. The source matrix S with the target vector c can be reduced with an efficient algorithm \[4, 17\] in O(a + N^2(NN_B)^{\omega-1}) time where a is the total number of nonzero elements in [S|c] and \omega \in (2, 2.373) is the exponent for matrix multiplication. We have a = O(NN_B \cdot N^2) = O(N^3N_B). Therefore, for each block B we spend O(N^3N_B + N^2(NN_B)^{\omega-1}) time. Then, observing \sum_{B \in \mathcal{B}} N_B = O(N), for each column we spend a total time of
\[
\sum_{B \in \mathcal{B}} O(N^3N_B + N^2(NN_B)^{\omega-1}) = O(N^4 + N^{\omega+1} \sum_{B \in \mathcal{B}} N_B^{\omega-1}) = O(N^4 + N^{2\omega}) = O(N^{2\omega}).
\]

Therefore, counting for all of the O(N) columns, the total time for decomposition takes O(N^{2\omega+1}) time.

We finish this analysis by commenting that one can build the presentation matrix from a given simplicial filtration consisting of n simplices leading to the following cases: (i) For 0-th homology, the boundary matrix \( \partial_1 \) can be taken as the presentation matrix giving \( N = O(n) \) and a total time complexity of \( O(n^{2\omega+1}) \); (ii) for 2-parameter case, \( N = O(n) \) and presentations can be computed in \( O(n^3) \) time giving a total time complexity of \( O(n^{2\omega+1}) \); (iii) for d-parameter case, \( N = O(n^d) \) and a presentation matrix can be computed in \( O(n^{2d+1}) \) time giving a total time complexity of \( O(n^{(2d+1)d}) \). We discuss the details in Section 14.5.
14.4.1 Running $\text{TotDiagonalize}$ on the working example in Figure 14.5

**Example 2.** Consider the binary matrix after simplification as illustrated in Example 1.

$$\begin{pmatrix}
\{1,1\} & c_1^{(1,1)} \\
\{1,0\} & r_1^{(0,1)} \\
\{1,2\} & r_2^{(1,0)} \\
\{0,1\} & r_1^{(1,1)} \\
\{1,2\} & r_3^{(2,1)} \\
\end{pmatrix}$$

with 4 admissible operations: $r_3 \rightarrow r_1, r_3 \rightarrow r_2, c_1 \rightarrow c_2, c_1 \rightarrow c_3$. The matrix remains the same after the first column $c_1$ is processed in $\text{TotDiagonalize}$.

Figure 14.12: Diagonalizing the binary matrix given in Example 1. It can be viewed as multiplying the original matrix $\partial$ with a left matrix $U$ that represents the row operation and a right matrix $V$ that represents the column operations.

Before the first iteration, $B$ is initialized to be $B = \{B_1 = \{(1), \emptyset\}, B_2 = \{(2), \emptyset\}, B_3 = \{(3), \emptyset\}\}$. In the first iteration when $t = 1$, we have block $B_0 = (\emptyset, \{1\})$ for column $c_1$. For $B_1 = \{(1), \emptyset\}$, the target block we hope to zero out is $T = \{(1), \{1\}\}$. So we call $\text{BlockReduce}(T)$ to check if $A|_T$ can be zeroed out and update the entries on $T$ according to the results of $\text{BlockReduce}(T)$. There is only one admissible operation from outside of $T$ into it, namely, $r_3 \rightarrow r_1$. The target vector $c = \text{Lin}(A|_T)$ and the source matrix $S = \{\text{Lin}((\delta_{1,3}A)|_T)\}$ are:

$$S = \begin{bmatrix}
\text{Lin}((\delta_{1,3}A)|_T) \\
0 \\
c = \text{Lin}(A|_T)
\end{bmatrix}$$

The result of $\text{ColReduce}(S, c)$ stays the same as its input. That means we cannot reduce $c$ at all. Therefore, $\text{BlockReduce}(T, t)$ returns false and nothing is updated in the original matrix.

It is not surprising that the matrix remains the same because the only admissible operation that can affect $T$ does not change any entries in $T$ at all. So there is nothing one can do to reduce it, which results in merging $B_1 \oplus B_0 = \{(1), \{1\}\}$. Similarly, for $B_2$ with $T = \{(2), \{1\}\}$, the only admissible operation $r_3 \rightarrow r_2$ does not change anything in $T$. Therefore, the matrix does not change and $B_2$ is merged with $B_1 \oplus B_0$, which results in the block $\{(1,2), \{1\}\}$. For $B_3$ with $T = \{(3), \{1\}\}$, there is no admissible operation. So the matrix does not change. But $A|_T = A|_{\{(3),\{1\}\}} = 0$. That means $\text{BlockReduce}$ returns true. Therefore, we do not merge $B_3$. In summary, $B_0, B_1, B_2$ are merged to be one block $\{(1,2), \{1\}\}$ in the first iteration. So after the first iteration, there are two index blocks in $B^{(1)}$: $\{(1,2), \{1\}\}$ and $\{(3), \emptyset\}$.
In the second iteration \( t = 2 \), we process the second column \( c_2 \). Now \( B_1 = ([1, 2], [1]) \), \( B_2 = ([3], \emptyset) \) and \( B_0 = (\emptyset, [2]) \). For the block \( B_1 = ([1, 2], [1]) \), the target block we hope to zero out is \( T = ([1, 2], [2]) \). There are three admissible operations from outside of \( T \) into \( T \), \( r_3 \rightarrow r_1, r_3 \rightarrow r_2, c_1 \rightarrow c_2 \). \text{BlockReduce}(T) constructs the target vector \( c = \text{Lin}(A_T) \) and the source matrix \( S = [\text{Lin}([\delta_{1,3}]A)_T], \text{Lin}([\delta_{2,3}]A)_T], \text{Lin}(A[\delta_{1,2}])_T] \) illustrated as follows:

\[
S = \begin{bmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0
\end{bmatrix}
\]

The result of \text{ColReduce}(S, c) is

\[
S = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\]

So the \text{BlockReduce} updates \( A|_T \) to get the following updated matrix:

\[
A' = \begin{pmatrix}
c_1^{(1,1)} & c_2^{(1,2)} + c_1^{(1,1)} & c_3^{(2,1)} \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 1
\end{pmatrix}
\]

and returns \text{true} since \( A'|_T = 0 \). Therefore, we do not merge \( B_1 \). We continue to check for the block \( B_2 = ([3], \emptyset) \) and \( T = ([3], [1, 2]) \), whether \( A'|_T \) can be reduced to zero. There is no admissible operation for this block at all. Therefore, the matrix stays the same and \text{BlockReduce} returns \text{false}. We merge \( B_2 \oplus B_0 = ([3], [2]) \). Continuing for the third iteration \( t = 3 \), we can see that nothing changes and finally the algorithm returns the matrix \( A' \) shown above as the final result. It is the correct total diagonalization with two index blocks in \( \mathbb{B}^3 \): \( B_1 = ([1, 2], [1]) \) and \( B_2 = ([3], [2, 3]) \).

By examining the process of \text{ColReduce}(S, c) for \( T = ([1, 2], [2, 3]) \) shown above, it is not hard to see that, the target column \( c \) is reduced by \( c \leftarrow c + \text{Lin}([\delta_{2,3}]A)_T] + \text{Lin}(A[\delta_{1,2}])_T] \), which reveals that the original matrix is updated by operations \( r_3 \rightarrow r_2 \) and \( c_1 \rightarrow c_2 \).

We can further transform it back to the original form of the presentation matrix \([\partial_1]\). Observe that a row addition \( r_i \leftarrow r_i + r_j \) reverts to a basis change in the opposite direction.

\[
[\partial_1] = \begin{bmatrix}
\partial_1^b \\
\partial_1^r \\
\partial_1^g
\end{bmatrix}
= \begin{bmatrix}
e_r^{(1,1)} & e_b^{(1,2)} & e_g^{(2,1)} \\
\partial_1^b \\
\partial_1^r \\
\partial_1^g
\end{bmatrix}
\]

\[
\Rightarrow[\partial_1]^* = \begin{bmatrix}
e_r^{(1,1)} & e_b^{(1,2)} + t^{(0,1)} e_r^{(1,1)} & e_g^{(2,1)} \\
\partial_1^b \\
\partial_1^r \\
\partial_1^g
\end{bmatrix}
\]
14.5 Computing presentations

Now that we know how to decompose a presentation by diagonalizing its matrix form, we describe how to construct and compute these matrices in this section. For a persistence module $H_p$ with $p$-th homology groups, we consider a presentation $C_{p+1} \to Z_p \to H_p \to 0$ where $C_{p+1}$ is a graded module of $(p+1)$-chains and $Z_p$ is a graded module of $p$-cycles which we describe now. Recall that a $(d$-parameter) simplicial filtration is a family of simplicial complexes $\{X_u\}_{u \in \mathbb{Z}^d}$ such that for each grade $u \in \mathbb{Z}^d$ and each $i = 1, \cdots, d$, $X_u \subseteq X_{u+i}$.

14.5.1 Graded chain, cycle, and boundary modules

We obtain a simplicial chain complex $(\cdots, C_p(X_u), \partial_p, C_{p+1}(X_u), \partial_{p+1}, \cdots)$ induced by the canonical inclusion $u$. For each grade $u \leq v \in \mathbb{Z}^d$, a family of inclusion maps $C_*(X_u) \hookrightarrow C_*(X_v)$ is induced by the canonical inclusion $X_u \hookrightarrow X_v$ giving rise to the following diagram:

\[
\begin{array}{cccccccc}
C_*(X_u) & : & \cdots & \xrightarrow{\partial_{p+2}} & C_{p+1}(X_u) & \xrightarrow{\partial_{p+1}} & C_p(X_u) & \xrightarrow{\partial_p} & C_{p-1}(X_u) & \xrightarrow{\partial_{p-1}} & \cdots \\
\downarrow & & & & \downarrow & & \downarrow & & \downarrow & & \\
C_*(X_v) & : & \cdots & \xrightarrow{\partial_{p+2}} & C_{p+1}(X_v) & \xrightarrow{\partial_{p+1}} & C_p(X_v) & \xrightarrow{\partial_p} & C_{p-1}(X_v) & \xrightarrow{\partial_{p-1}} & \cdots
\end{array}
\]

For each chain complex $C_*(X_u)$, we have the cycle spaces $Z_p(X_u)$'s and boundary spaces $B_p(X_u)$'s as kernels and images of boundary maps $\partial_p$'s respectively, and the homology group $H_p(X_u) = Z_p(X_u)/B_p(X_u)$ as the cokernel of the inclusion maps $B_p(X_u) \hookrightarrow Z_p(X_u)$. In line with category theory we use the notations $\text{im}$, $\text{ker}$, $\text{coker}$ for indicating both the modules of kernel, image, cokernel and the corresponding morphisms uniquely determined by their constructions$^3$.

We obtain the following commutative diagram:

\[
\begin{array}{ccccccccccccccc}
\text{im}\partial_{p+1} & \hookrightarrow & Z_p(X_u) & \xrightarrow{\coker} & H_p(X_u) & \xrightarrow{\text{ker}\partial_p} & \text{ker}\partial_{p+1} & \hookrightarrow & B_p(X_u) & \xrightarrow{\text{coker}\partial_{p+1}} & C_p(X_u) & \xrightarrow{\text{ker}\partial_{p+1}} & \text{coker}\partial_{p+1} & \hookrightarrow & \cdots
\end{array}
\]

In the language of graded modules, for each $p$, the family of vector spaces and linear maps (inclusions) $\{C_p(X_u)\}_{u \in \mathbb{Z}^d}$ can be summarized as a $\mathbb{Z}^d$-graded $R$-module:

\[
C_p(X) := \bigoplus_{u \in \mathbb{Z}^d} C_p(X_u), \text{ with the ring action } t_i : C_p(X_u) : C_p(X_u) \hookrightarrow C_p(X_{u+i}) \forall i, \forall u.
\]

That is, the ring $R$ acts as the linear maps (inclusions) between pairs of vector spaces in $C_p(X)$ with comparable grades. It is not too hard to check that this $C_p(X)$ is indeed a graded module. Each $p$-chain in a chain space $C_p(X_u)$ is a homogeneous element with grade $u$.

Then we have a chain complex of graded modules $(C_*(X), \partial_*)$ where $\partial_* : C_*(X) \to C_{*-1}(X)$ is the boundary morphism given by $\partial_* \triangleq \bigoplus_{u \in \mathbb{Z}^d} \partial_{*,u}$ with $\partial_{*,u} : C_*(X_u) \to C_{*-1}(X_u)$ being the boundary map on $C_*(X_u)$.

$^3$e.g. $\text{ker}\partial_p$ denotes the inclusion of $Z_p$ into $C_p$.
The kernel and image of a graded module morphism are also graded modules as submodules of domain and codomain respectively whereas the cokernel is a quotient module of the codomain. They can also be defined grade-wise in the expected way:

For \( f : M \rightarrow N \), \((\ker f)_u = \ker f_u\), \((\text{im} f)_u = \text{im} f_u\), \((\text{coker} f)_u = \text{coker} f_u\).

All the linear maps are naturally induced from the original linear maps in \( M \) and \( N \). In our chain complex cases, the kernel and image of the boundary morphism \( \partial_p : C_p(X) \rightarrow C_{p-1}(X) \) is the family of cycle spaces \( Z_p(X) \) and family of boundary spaces \( B_{p-1}(X) \) respectively with linear maps induced by inclusions. Also, from the inclusion induced morphism \( B_p(X) \hookrightarrow Z_p(X) \), we have the cokernel module \( H_p(X) \), consisting of homology groups \( \bigoplus_{u \in Z^d} H_p(X_u) \) and linear maps induced from inclusion maps \( X_u \hookrightarrow X_v \) for each comparable pairs \( u \leq v \). This \( H_p(X) \) is the persistence module \( M \) which we decompose. Classical persistence modules arising from a filtration of a simplicial complex over \( \mathbb{Z} \) is an example of a 1-parameter persistence module where the action \( t_1 \cdot M_u \subseteq M_{u+e_1} \) signifies the linear map \( M_u \rightarrow M_v \) between homology groups induced by the inclusion of the complex at \( u \) into the complex at \( v = u + e_1 \).

In our case, we have chain complex of graded modules and induced homology groups which can be succinctly described by the following diagram:

\[
\begin{array}{cccccccc}
\cdots & C_{p+1}(X) & \xrightarrow{\partial_{p+1}} & C_p(X) & \xrightarrow{\partial_p} & C_{p-1}(X) & \xrightarrow{\partial_{p-1}} & \cdots \\
& \text{ker}(\partial_p) & \downarrow & \text{ker}(\partial_p) & \downarrow & \text{ker}(\partial_p) & \downarrow & \\
& \text{im}(\partial_{p+1}) & \downarrow & \text{im}(\partial_p) & \downarrow & \text{im}(\partial_{p-1}) & \downarrow & \\
& \ker(\partial_{p+1}) & \downarrow & \ker(\partial_p) & \downarrow & \ker(\partial_{p-1}) & \downarrow & \\
& \cdots & \downarrow & \cdots & \downarrow & \cdots & \downarrow & \\
B_p(X) & \hookrightarrow & Z_p(X) & \rightarrow & H_p(X) & \hookrightarrow & Z_{p-1}(X) & \rightarrow & H_{p-1}(X) & \rightarrow & \cdots
\end{array}
\]

**An assumption.** We always assume that the simplicial filtration is 1-critical, which means that each simplex has a unique earliest birth time. For the case which is not 1-critical, called multi-critical, one may utilize the mapping telescope, a standard algebraic construction [13], which transforms a multi-critical filtration to a 1-critical one. However, notice that this transformation increases the input size depending on the multiplicity of the incomparable birth times of the simplices. For 1-critical filtrations, each module \( C_p \) is free. With a fixed basis for each free module \( C_p \), a concrete matrix \( [\partial_p] \) for each boundary morphism \( \partial_p \) based on the chosen bases can be constructed.

With this input, we discuss our strategies for different cases that depend on two parameters, \( d \), the number of parameters of filtration function, and \( p \), the dimension of the homology groups in the persistence modules.

Note that a presentation gives an exact sequence \( F^1 \rightarrow F^0 \rightarrow H \rightarrow 0 \). To reveal further details of a presentation of \( H \), we recognize that it respects the following commutative diagram,

\[
\begin{array}{cccccccc}
F^1 & \xrightarrow{f^1} & F^0 & \xrightarrow{f^0 = \text{coker} f^1} & H \\
\text{im}(f^1) & \downarrow & \text{ker} f^0 & \downarrow & \text{ker} f^0 & \downarrow & \text{ker} f^0 & \downarrow & \text{ker} f^0 & \downarrow & \text{ker} f^0
\end{array}
\]
where $Y^1 \hookrightarrow F^0$ is the kernel of $f^0$. With this diagram being commutative, all maps in this diagram are essentially determined by the presentation map $f^1$. We call the surjective map $f^0 : F^0 \to H$ generating map, and $Y^1 = \ker f^0$ the 1$st$ syzygy module of $H$.

### 14.5.2 Multiparameter filtration, zero-dimensional homology

In this case $p = 0$ and $d > 0$. In this case, we obtain a presentation matrix straightforwardly with the observation that the module $Z_0$ of cycle spaces coincides with the module $C_0$ of chain spaces.

- **Presentation:** $C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\text{coker} \partial_0} H_0$

- **Presentation matrix** $= [\partial_1]$ is given as part of the input.

**Justification.** For $p = 0$, the cycle module $Z_0 = C_0$ is a free module. So we have the presentation of $H_0$ as claimed. It is easy to check that $\partial_1 : C_1 \to C_0$ is a presentation of $H_0$ since both $C_1$ and $C_0$ are free modules. With standard basis of chain modules $C_p$'s, we have a presentation matrix $[\partial_1]$ as the valid input to our decomposition algorithm.

The 0-th homology in our working example (Figure 14.5) corresponds to this case. The presentation matrix is the same as the matrix of boundary morphism $\partial_1$.

### 14.5.3 2-parameter filtration, multi-dimensional homology

In this case, $d = 2$ and $p \geq 0$. Lesnick and Wright [22] present an algorithm to compute a presentation, in fact a minimal presentation, for this case. When $d = 2$, by Hilbert Syzygy Theorem [14], the kernel of a morphism between two free graded modules is always free. This implies that the canonical surjective map $Z_p \twoheadrightarrow H_p$ from free module $Z_p$ can be naturally chosen as a generating map in the presentation of $H_p$. In this case we have:

- **Presentation:** $C_{p+1} \xrightarrow{\tilde{\partial}_{p+1}} Z_p \xrightarrow{\text{coker} \tilde{\partial}_{p+1}} H_p$ where $\tilde{\partial}_{p+1}$ is the induced map from the diagram:

\[
\begin{array}{ccc}
C_{p+1} & \xrightarrow{\partial_{p+1}} & Z_p \\
\downarrow{\text{im} \partial_{p+1}} & \searrow{\text{im} \partial_{p+1}} & \downarrow{\ker \partial_{p+1}} \\
B_p & \xleftarrow{\partial_{p+1}} & Z_p \\
\end{array}
\]

- **Presentation matrix** $= [\tilde{\partial}_{p+1}]$ is constructed as follows:

1. Compute a basis $G(Z_p)$ for the free module $Z_p$ where $G(Z_p)$ is presented as a set of generators in the basis of $C_p$. This can be done by an algorithm in [22]. Take $G(Z_p)$ as the row basis of the presentation matrix $[\tilde{\partial}_{p+1}]$.

2. Present im $\partial_{p+1}$ in the basis of $G(Z_p)$ to get the presentation matrix $[\tilde{\partial}_{p+1}]$ of the induced map as follows. Originally, im $\partial_{p+1}$ is presented in the basis of $C_p$ through the given matrix $[\partial_{p+1}]$. One needs to rewrite each column of $[\partial_{p+1}]$ in the basis $G(Z_p)$ computed in the previous step. This can be done as follows. Let $[G(Z_p)]$ denote the
matrix presenting basis elements in \( G(\mathbb{Z}_p) \) in the basis of \( C_p \). Let \( c \) be any column vector in \([\partial_{p+1}]\). We reduce \( c \) to zero vector by the matrix \([G(\mathbb{Z}_p)]\) and note the columns that are added to \( c \). These columns provide the necessary presentation of \( c \) in the basis \( G(\mathbb{Z}_p) \). This reduction can be done by the persistent algorithm described earlier.

**Justification.** Unlike \( p = 0 \) case, for \( p > 0 \), we just know \( \mathbb{Z}_p \) is a (proper) submodule of \( C_p \), which means that \( \mathbb{Z}_p \) is not necessarily equal to the free module \( C_p \). However, fortunately for \( d = 2 \), the module \( \mathbb{Z}_p \) is free, and we have an efficient algorithm to compute a basis of \( \mathbb{Z}_p \) as the kernel of the boundary map \( \partial_p : C_p \rightarrow C_{p-1} \). Then, we can construct the following presentation of \( H_p \):

\[
\begin{array}{c}
\mathbb{B}_p \\
\uparrow \text{im} \partial_{p+1} \\
\rightarrow \mathbb{C}_{p+1} \\
\downarrow \partial_{p+1} \\
\rightarrow \mathbb{Z}_p \\
\downarrow \text{coker} \partial_{p+1} \\
\rightarrow H_p \\
\rightarrow \mathbf{0}
\end{array}
\]

Here the \( \partial_{p+1} \) is an induced map from \( \partial_{p+1} \). With a fixed basis on \( \mathbb{Z}_p \) and standard basis of \( \mathbb{C}_{p+1} \), we rewrite the presentation matrix \([\partial_{p+1}]\) to get \([\tilde{\partial}_{p+1}]\), which constitutes a valid input to our decomposition algorithm.

### 14.5.4 \( d > 2 \)-parameter filtration, multi-dimensional homology

The above construction of presentation matrix cannot be extended straightforwardly to \( d \)-parameter persistence modules \( d > 2 \). Unlike the case in \( d \leq 2 \), the cycle module \( \mathbb{Z} \) is not necessarily free when \( d > 2 \). The issue caused by non-free \( \mathbb{Z} \) is that, if we use the same presentation matrix as we did in the previous case with free \( \mathbb{Z} \), we may lose some relations coming from the inner relations of a generating set of \( \mathbb{Z} \). One can fix this problem by adding these inner relations into the presentation matrix as detailed in [10]. It is more complicated and we skip it here.

Figure 14.13 shows a simple example of a filtration of simplicial complex whose persistence module \( H_p \) for \( p = 1 \) is a quotient module of non-free module \( \mathbb{Z} \). The module \( H_1 \) is generated by three 1-cycles presented as \( g_1^{(0,1,1)}, g_2^{(1,0,1)}, g_3^{(1,1,0)} \). But when they appear together in \( (1,1,1) \), there is a relation between these three: \( t^{(1,0,0)} g_1^{(0,1,1)} + t^{(0,1,0)} g_2^{(1,0,1)} + t^{(0,0,1)} g_3^{(1,1,0)} = 0 \). Although \( \text{im} \partial_1 = 0 \), we still have a nontrivial relation from \( \mathbb{Z} \). So, we have \( H_1 = \text{ker} [g_1^{(0,1,1)}, g_2^{(1,0,1)}, g_3^{(1,1,0)} : s^{(1,1,1)} = t^{(1,0,0)} g_1^{(0,1,1)} + t^{(0,1,0)} g_2^{(1,0,1)} + t^{(0,0,1)} g_3^{(1,1,0)}] \). The presentation matrix turns out to be the following:

\[
\begin{pmatrix}
g_1^{(0,1,1)} \\
g_2^{(1,0,1)} \\
g_3^{(1,1,0)}
\end{pmatrix}
\begin{pmatrix}
t^{(1,0,0)} \\
t^{(0,1,0)} \\
t^{(0,0,1)}
\end{pmatrix}
\]
Figure 14.13: An example of a filtration of simplicial complex for \( d = 3 \) with non-free \( \mathbb{Z} \) when \( p = 1 \). The three red cycles are three generators in \( \mathbb{Z}_1 \). However, at grading \((1, 1, 1)\), the earliest time these three red cycles exist simultaneously, there is a relation among these three generators.

### 14.5.5 Time complexity

Now we consider the time complexity for computing presentation and decomposition together. Let \( n \) be the size of the input filtration, that is, total number of simplices obtained by counting at most one new simplex at a grid point of \( \mathbb{Z}^d \). We consider three different cases as before:

**Multi-parameter, 0-th homology**: In this case, the presentation matrix \([\partial_1]\) where \( \partial_1 : \mathbb{C}_1 \to \mathbb{C}_0 \) has size \( O(n) \times O(n) \), that is, \( N = O(n) \). Therefore, the total time complexity for this case is \( O(n^{2\omega+1}) \).

**2-parameter, multi-dimensional homology**: In this case, as described in section 14.5.3, first we compute a basis \( \mathcal{G}(\mathbb{Z}_p) \) that is presented in the basis of \( \mathbb{C}_p \). This is done by the algorithm of Lesnick and Wright [22] which runs in \( O(n^3) \) time. Using \( \mathcal{G}(\mathbb{Z}_p) \), we compute the presentation matrix \( [\tilde{\partial}_{p+1}] \) as described in section 14.5.3. This can be done in \( O(n^3) \) time assuming that \( \mathcal{G}(\mathbb{Z}_p) \) has at most \( O(n) \) elements. The presentation matrix is decomposed with \text{TotDiagonalize} as in the previous case. However, to claim that it runs in \( O(n^{2\omega+1}) \) time, one needs to ensure that the basis \( \mathcal{G}(\mathbb{Z}_p) \) has \( O(n) \) elements. This follows from the following proposition. Its proof implicitly follows from the algorithm of Lesnick and Wright [22] as it computes a basis of size at most \( n \). For an explicit proof, see [10].

**Proposition 7.** Assume that \( \mathbb{C}_p \) is an \( R \)-module given by \( n \) \( p \)-simplices. Then, there are at most \( n \) elements in \( \mathcal{G}(\mathbb{Z}_p) \).

In summary, the total time complexity in this case becomes \( O(n^3) + O(n^{2\omega+1}) = O(n^{2\omega+1}) \).

**d-parameter, \( d > 2 \), multi-dimensional homology**: For \( d \)-parameter persistence modules where \( d > 2 \), currently efficient computation of the presentation matrix remains elusive. A rather crude
algorithm running in time $O(n^{2d+1})$ that produces a presentation matrix of dimensions $O(n^d) \times O(n^d)$ is given in [10]. Plugging $N = O(n^d)$ and taking the computation of presentation matrix into consideration, we get a time complexity bound of $O(n^{2d+1}) + O(n^{(2\omega+1)d}) = O(n^{(2\omega+1)d})$.

### 14.6 Invariants

For a given persistence module, it is useful to compute invariants that in some sense summarize the information contained in them. Ideally, these invariants should characterize the input module completely, meaning that the two invariants should be equal if and only if the modules are isomorphic. Persistence diagrams for 1-parameter tame persistence modules are such invariants. For multiparameter persistence modules, no such complete invariants exist that are finite and hence computable. However, we can still aim for invariants that are computable and characterize the modules in some limited sense, meaning that these invariants remain equal for isomorphic modules though may not differentiate non-isomorphic modules. Of course, their effectiveness in practice is determined by their discriminative power. We present two such invariants below: the first one rank invariant was suggested in [6] whereas the second one graded Betti number was brought to TDA by [19] and studied further in [21].

#### 14.6.1 Rank invariants

Assume that the input module $M$ is finitely generated as before and additionally finitely supported, that is, if $M = \bigoplus_u M_u$ then $\text{supp}(M) = \{u \mid M_u \neq 0\} \subset \mathbb{Z}^d$ is finite. Furthermore, we have $u \leq v$ in $\text{supp}(M)$ if and only if there exist $u = s_0, s_1, \ldots, s_k = v$ in $\text{supp}(M)$ so that for every $i \in [0, k - 1]$, $s_i \leq s_{i+1}$ in the poset $\mathbb{Z}^d$. This last condition is imposed to prohibit two comparable grades in $\mathbb{Z}^d$ becoming incomparable in $\text{supp}(M)$ because of the omission of the chain connecting the two grades in $\mathbb{Z}^d$.

For a finitely generated and supported module $M$, we can compute a finite number of ranks of linear maps which collectively form the rank invariant of $M$. For two grades $u \preceq v$, the linear maps between $M_u$ and $M_v$ are not defined. In the following definitions, we take them as zero maps.

**Definition 17 (Rank invariant).** Let $r_{uv} = \text{rank}(M_u \rightarrow M_v)$ for every pair $u, v \in \text{supp}(M)$ with $u \leq v$. The collection $\{r_{uv}\}$ is called the rank invariant of $M$.

**Fact 2.** Rank invariant of a 1-parameter module is a complete invariant. For a 1-parameter persistence module $H_p$, it is given by persistent Betti numbers $\beta_{p,j}$ as defined earlier.

Although in 1-parameter case, the rank invariant provides complete information about the module, it does not do so for multiparameter persistence modules. For example, it cannot provide information about ‘birth’ and ‘death’ of the generators. This information can be deduced from a wider collection of rank invariant data called multirank invariant where we compute ranks of the linear maps between vector spaces at multiple grades. Multirank invariant is still not a complete invariant.

**Definition 18 (Multirank invariant).** The collection $\{r_{UV}\}$ for every pair $U \subseteq \text{supp}(M)$ and $V \subseteq \text{supp}(M)$ where $r_{UV} = \text{rank}(\bigoplus_{u \in U} M_u \rightarrow \bigoplus_{v \in V} M_v)$ is called the multirank invariant of $M$. 
We can retrieve the information about birth and death of generators from the multirank. For a grade \( u \), define its immediate predecessor \( P_u \) and immediate successors \( S_u \) as:

\[
P_u = \{ u' \in \text{supp}(M) \mid u' \leq u \text{ and } \not\exists u'' \text{ with } u < u'' < u \} \\
S_u = \{ u' \in \text{supp}(M) \mid u' \geq u \text{ and } \not\exists u'' \text{ with } u < u'' < u' \}.
\]

**Fact 3.**

1. We have that \( m \) generators get born at grade \( u \) if and only if \( \text{coker} \left( \bigoplus_{u' \in P_u} M_{u'} \to M_u \right) \) has dimension \( m \).

2. We have that \( m \) generators die at grade \( u \) if and only if \( \text{ker} \left( M_u \to \bigoplus_{u' \in S_u} M_{u'} \right) \) has dimension \( m \).

Although rank invariants cannot characterize a multiparameter persistence module completely in general, they do so for the special case of interval decomposable modules. We will describe these modules in details in the next chapter. Here we introduce them briefly.

We call \( I \subseteq \text{supp}(M) \) an interval if for every \( u, v \in I \), (i) if \( u < w < v \), then \( w \in I \), (ii) there is an ordered sequence \( u = u_1, \ldots, u_k = v \) in \( I \) so that either \( u_i \leq u_{i+1} \) or \( u_{i+1} \leq u_i \). We call a persistence module with support on an interval an interval module. A persistence module \( M \) is called interval decomposable if there is a decomposition \( M = \bigoplus M^i \) where each \( M^i \) is an interval module.

**Fact 4.** Two interval decomposable modules are isomorphic if and only if they have the same rank invariants.

### 14.6.2 Graded Betti numbers and blockcodes

For 1-parameter persistence modules, the barcodes provide a complete invariant. For multiparameter persistence, we first introduce an invariant called graded Betti number, which we refine further to define persistent graded Betti numbers as a generalization of persistence diagrams. The decomposition of a module also allows us to define blockcodes as a generalization of barcodes. Both of them depend on the ideas of free resolution and graded Betti numbers which are well studied in commutative algebra and are first introduced in topological data analysis by Knudson [19].

**Definition 19** (Free resolution). For a graded module \( M \), a free resolution \( \mathcal{F} \to M \) is an exact sequence:

\[
\cdots \to F^2 \xrightarrow{f^2} F^1 \xrightarrow{f^1} F^0 \xrightarrow{f^0} M \to 0
\]

where each \( F^i \) is a free graded \( R \)-module.

Now we observe that a free resolution can be obtained as an extension of a free presentation. Consider a free presentation of \( M \) as depicted below.
If the presentation map $f^1$ has nontrivial kernel, we can find a nontrivial map $f^2 : F^2 \to F^1$ with $\text{im} f^2 = \ker f^1$, which implies $\text{coker} f^2 \cong \text{im} f^1 = \ker f^0 = Y^1$. Therefore, $f^2$ is in fact a presentation map of the module $Y^1$ which is so called the first syzygy module of $M$ (named after Hilbert’s famous syzygy theorem [14]). We can keep doing this to get $f^3, f^4, \ldots$ by constructing presentation maps on higher order syzygy modules $Y^2, Y^3, \ldots$ of $M$, which results in a diagram depicted below, which is gives a free resolution of $M$.

\[
\begin{array}{ccccccc}
\vdots & \xrightarrow{f^3} & F^3 & \xrightarrow{f^2} & F^2 & \xrightarrow{f^1} & F^1 & \xrightarrow{f^0} & F^0 & \xrightarrow{f^0=\text{coker} f^1} & M \\
\end{array}
\]

Free resolution is not unique. However, there exists an essentially unique minimal free resolution in the sense that any free resolution can be obtained by summing the minimal free resolution with a free resolution of a trivial module. Below we give a construction to build a minimal free resolution from a minimal free presentation. The proof that it indeed creates a minimal free resolution can be found in [3, 26].

**Construction of minimal free resolution.** Choose a minimal set of homogeneous generators $g_1, \cdots, g_n$ of $M$. Let $F^0 = \bigoplus_{i=1}^n R_{\text{gr}(g_i)}$ with standard basis $e^1_{\text{gr}(g_1)}, \cdots, e^n_{\text{gr}(g_n)}$ of $F^0$. The homogeneous $R$-map $f^0 : F^0 \to M$ is determined by $f^0(e_i) = g_i$. Now the 1st syzygy module of $M$, $Y_1 \xrightarrow{\ker f^0} F^0$, is again a finitely generated graded $R$-module. We choose a minimal set of homogeneous generators $y_1, \cdots, y_m$ of $Y_1$ and let $F^1 = \bigoplus_{j=1}^m R_{\text{gr}(y_j)}$ with standard basis $e^1_{\text{gr}(y_1)}, \cdots, e^m_{\text{gr}(y_m)}$ of $F^1$. The homogeneous $R$-map $f^1 : F^1 \to F^0$ is determined by $f^1(\cdot') = y_j$. By repeating this procedure for $Y_2 = \ker f^1$ and moving backward further, one gets a graded free resolution of $M$.

**Definition 20** (Graded Betti numbers). Let $F^j$ be a free module in the minimal free resolution of a graded module $M$. Let $\beta^M_{j,u}$ be the multiplicity of each grade $u \in \mathbb{Z}^d$ in the multiset consisting of the grades of homogeneous basis elements for $F^j$. Then, the mapping $\beta^M_{(-,-)} : \mathbb{Z}_{\geq 0} \times \mathbb{Z}^d \to \mathbb{Z}_{\geq 0}$ is an invariant called the graded Betti numbers of $M$.

For example, the graded Betti number of the persistence module for our working example in Figure 14.5 is listed as Table 14.1.

<table>
<thead>
<tr>
<th>$\beta^M$</th>
<th>(1,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
<th>(2,1)</th>
<th>(1,2)</th>
<th>(2,2)</th>
<th>$\cdots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta_0$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\beta_1$</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\beta_2$</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\beta_{\geq 3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14.1: All the nonzero graded Betti numbers $\beta_{j,u}$ are listed in the table. Empty items are all zeros.

**Definition 21** (Persistent graded Betti numbers). Let $M = \bigoplus M^i$ be a total decomposition of a graded module $M$. We have for each indecomposable $M^i$, the refined graded Betti numbers
\( \beta^M = \{ \beta^M_{j,u} | j \in \mathbb{N}, u \in \mathbb{Z}^d \} \). We call the set \( \mathcal{PB}(M) := \{ \beta^M \} \) the persistent graded Betti numbers of \( M \).

For the working example in Figure 14.5, the persistent graded Betti numbers are given in two tables listed in Table 14.2.

<table>
<thead>
<tr>
<th>( \beta^M )</th>
<th>(1,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
<th>(2,1)</th>
<th>(1,2)</th>
<th>(2,2)</th>
<th>( \cdots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_0 )</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \beta_1 )</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \beta_{\geq 2} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \beta^M )</th>
<th>(1,0)</th>
<th>(0,1)</th>
<th>(1,1)</th>
<th>(2,1)</th>
<th>(1,2)</th>
<th>(2,2)</th>
<th>( \cdots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_0 )</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \beta_1 )</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \beta_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \beta_{\geq 3} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14.2: Persistence grades \( \mathcal{PB}(M) = \{ \beta^M, \beta^M_{1,u} \} \). All nonzero entries are listed in this table. Blank boxes indicate 0 entries.

One way to summarize the information of graded Betti numbers is to use the Hilbert function, which is also called dimension function [9] defined as:

\[
dm^M : \mathbb{Z}^d \to \mathbb{Z}_{\geq 0} \quad dm^M(u) = \dim(M_u)
\]

**Fact 5.** There is a relation between the graded Betti numbers and dimension function of a persistence module as follows:

\[
\forall u \in \mathbb{Z}^d, \; dm^M(u) = \sum_{v \leq u} \sum_j (-1)^j \beta_{j,v}
\]

Then for each indecomposable \( M^i \), we have the dimension function \( dm^M^i \) related to persistent graded Betti numbers restricted to \( M^i \).

**Definition 22 (Blockcode).** The set of dimension functions \( \mathcal{B}_{dm}^M(M) := \{ dm^M^i \} \) is called the blockcode of \( M \).

For our working example, the dimension functions of indecomposable summands \( M^1 \) and \( M^2 \) are (see Figure 14.14 for the visualization):

\[
dm^{M^1}(u) = \begin{cases} 
1 & \text{if } u \geq (1,0) \text{ or } u \geq (0,1) \\
0 & \text{otherwise}
\end{cases}, \quad dm^{M^2}(u) = \begin{cases} 
1 & \text{if } u = (1,1) \\
0 & \text{otherwise}
\end{cases}
\]  

(14.2)

We can read out some useful information from dimension functions on each indecomposable. We take the dimension functions of our working example as an example. For \( dm^{M^1} \), two connected components are born at the two left-bottom corners of the purple region. They are merged together immediately when they meet at grade \( (1,1) \). After that, they persist forever as
one connected component. For $dmM^2$, one connected component born at the left-bottom corner of the square green region. Later at the grades of left-top corner and right-bottom corner of the green region, it is merged with some other connected component with smaller grades of birth. Therefore, it only persists within this green region.

In general, both persistent graded Betti numbers and blockcodes are not sufficient to classify multiparameter persistence modules, which means they are not complete invariants. As indicated in [5], there is no complete discrete invariant for multiparameter persistence modules. However, interestingly, these two invariants are indeed complete invariants for interval decomposable modules like this example, which we will study in the next chapter.

### 14.7 Notes and Exercises

In one of the first extensions of the persistence algorithm for 1-parameter, the authors in [1] presented a matrix reduction based algorithm which applies to a very special case of commutative ladder $C_n$ for $n \leq 4$ defined on a subgrid of $\mathbb{Z}^2$. This matrix construction and the algorithm are...
very different from the one presented here. This algorithm may not terminate if the input does not satisfy the stated assumption.

We have already mentioned that the Meataxe algorithm [24] known in the computational algebra community can be used for more general modules and hence for persistence modules. The main advantage of this algorithm is that it applies to general persistence modules, but a major disadvantage is that it runs very slow. Even allowing approximation, the algorithm [16] runs in $O(N^{3(d+1)} \log q)$ time (or $O(N^{4(d+1)} \log q)$ as conjectured in [15] because of some special cases mentioned in [16]) where $N$ is the number of generators and relations in the input module that is defined with polynomial ring $\mathbb{Z}_q[t_1t_2 \ldots t_d]$.

Under suitable finiteness condition, the fact that persistence modules are indeed finitely presented graded modules over multivariate polynomials was first recognized by Carlsson et al. [5, 6] and Knudson [19] and further studied by Lesnick et al. [20, 22]. The graded module structure studied in algebraic topology and commutative algebra [12, 23] encodes a lot of information and thus can be leveraged for designing efficient algorithms. Lesnick and Wright [22] leveraged this fact to design an efficient algorithm for computing minimal presentations for 2-parameter persistence modules from an input filtration on a 2D grid. Recognizing the power of expressing persistence modules in terms of presentations, Dey and Xin [10] proposed the decomposition algorithm using matrix equivalents of presentations and their direct sums. The materials in this chapter are mostly taken from this paper. This decomposition algorithm can be viewed as a generalization of the classical persistence algorithm for 1-parameter though the matrix reduction technique is more involved because it has to accommodate constraints on grades. The algorithm in [10] handled these constraints using the technique of matrix linearization as described in Section 14.4.

As a generalization of the 1-parameter persistence algorithm, it is expected that our algorithm can be interpreted as computing invariants such as persistence diagrams or barcodes. A roadblock to this goal is that $d$-parameter persistence modules do not have complete discrete invariants for $d \geq 2$ [6, 20]. Consequently, one needs to invent other invariants suitable for multiparameter persistence modules. The rank invariants and multirank invariants described in Section 14.6.1 serve this purpose. There is a related notion of generalized persistence diagram introduced by Patel [25] and further studied in [18].

One natural approach taking advantage of a decomposition algorithm would be to consider the decomposition and take the discrete invariants in each indecomposable component. This gives invariants which may not be complete but still contain rich information. We mentioned two interpretations of the output of the algorithm presented in this chapter as two different invariants: persistent graded Betti numbers as a generalization of persistence diagrams and blockcodes as a generalization of barcodes. The persistent graded Betti numbers are linked to the graded Betti numbers studied in commutative algebra brought to TDA by [19]. The bigraded Betti numbers are further studied in [22]. By constructing the free resolution of a persistence module, we can compute its graded Betti numbers and then decompose them according to each indecomposable module, which results into the presistent graded Betti numbers. For each indecomposable, we apply dimension function [9], which is also known as the Hilbert function in commutative algebra to summarize the graded Betti numbers for each indecomposable module. This constitutes a blockcode for indecomposable module of the persistence module. The blockcode is a good vehicle for visualizing lower dimensional persistence modules such as 2- or 3-parameter persistence modules. For details on these invariants, see [10].
Exercises

1. Using the matrix diagonalization algorithm as described in this chapter, devise an algorithm to compute a minimal presentation of a 2-parameter persistence module given by a simplicial filtration over $\mathbb{Z}^2$.

2. Give an example of a 2-parameter simplicial filtration over $\mathbb{Z}^2$ at least one of whose decomposables is not free.

3. Give an example of a 2-parameter simplicial filtration over $\mathbb{Z}^2$ at least one of whose decomposables does not have non-trivial vector spaces of same dimensions over all grades.

4. Give an example of a 2-parameter persistence module $M$ with three generators and relations that have the following properties: (i) $M$ is indecomposable, (ii) $M$ has two indecomposables, (iii) $M$ has three indecomposables.

5. Prove that the cycle module $Z_p$ arising from a 2-parameter simplicial filtration is always free.

6. Design a polynomial time algorithm for computing decomposition of the persistence module induced by a given simplicial filtration over $\mathbb{Z}^2$ when a simplex can be a generator at different grades.

7. Let $A$ be a presentation matrix with $n$ generators and relations whose grades are distinct and totally ordered. Design an $O(n^3)$ time algorithm to decompose such a presentation. Interpret types of each indecomposable in such a case.


9. The algorithm \textsc{TorDiagonalize} has been written assuming that the field of the polynomial ring is $\mathbb{Z}_2$. Write it for a general finite field.

10. Design an efficient algorithm to compute the rank invariant of a module from the simplicial filtration inducing it.

11. Prove that a 2-parameter persistence module $M$ is an interval (see Section 14.6.1) if and only if $\text{supp}(M)$ is connected and each $M_u$ for $u \in \text{supp}(M)$ has dimension 1.

12. Suppose that a 2-parameter persistence module is given by a presentation matrix. Design an algorithm to determine if $M$ is interval or not without decomposing the input matrix (hint: consider computing graded Betti numbers from the grades of the rows and columns of the matrix).

13. Write a pseudocode for the construction of a minimal free resolution given in Section 14.6.2. Analyze its complexity.
Bibliography


