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.

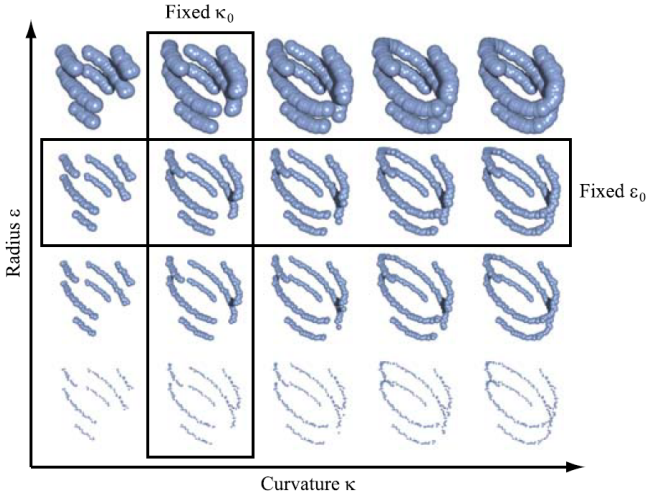


Figure 14.1: A bi-filtration parameterized over curvature and radius; picture taken from [6].

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-

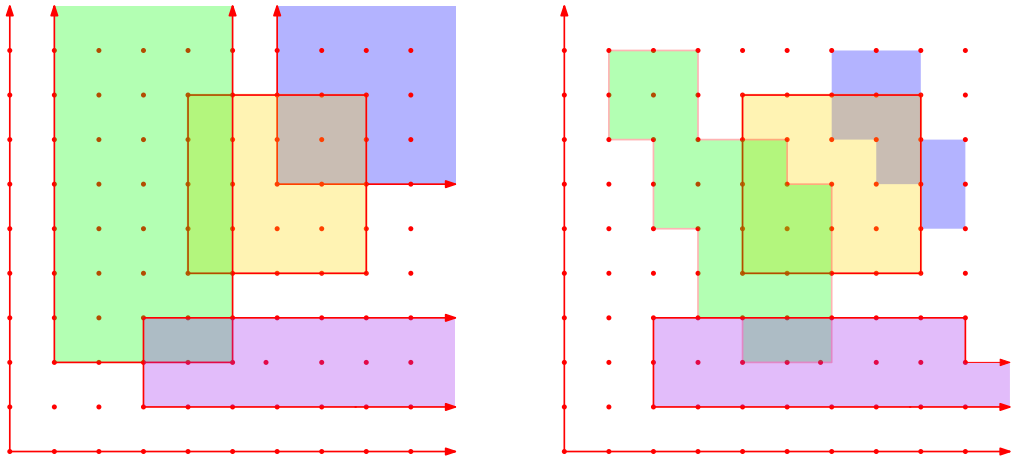


Figure 14.2: Decomposition of a finitely generated 2-parameter persistence module: (left) rectangle decomposable module: each indecomposable is supported by either bounded (A) or unbounded rectangle (B and C), D is a free module; (right) interval decomposable module: each indecomposable is supported over a 2D interval (defined in next chapter).

sented. 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 $M_i \to M_{i+1}$ 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 $\Bbbk[t]$ and presenting it with the so-called *generators* $\{g_i\}$ of the module and *relations* $\{\sum_i \alpha_i g_i = 0 \mid \alpha_i \in \Bbbk[t]\}$ 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 $\mathbf{k}^{D\times D}$ with entries in a fixed field $\mathbf{k} = \mathbb{Z}_q$, 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.

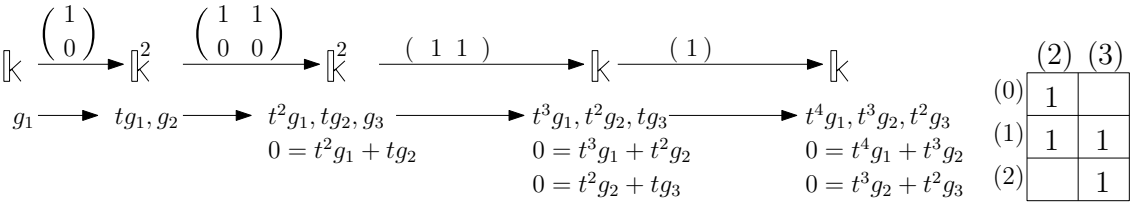


Figure 14.3: Costly presentation (top) vs. graded presentation (bottom,right). The top chain can be summarized by three generators g_1, g_2, g_3 at grades $(0), (1), (2)$ respectively, and two relations $0 = t^2g_1 + tg_2$, $0 = t_2g_2 + tg_3$ at grades $(2), (3)$ respectively (Definition 5). The grades of the generators and relations are given by the first times they appear in the chain. Finally, these information can be summarized succinctly by the presentation matrix on the right.

the most efficient way to transform an input that specifies generators and relations (or a simplicial filtration) to a representation matrix required by the Meataxe 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 $O(\binom{N}{d}) = O(N^d)$ 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 $O(N^{6(d+1)} \log q)$. Even allowing approximation, the algorithm runs in $O(N^{3(d+1)} \log q)$ 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 $R = \mathbf{k}[t_1, \cdots, t_d]$ [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 indecompsable 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 $O(N^{2\omega+1})$ 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 $O(n^{d-1}) \times O(n^{d-1})$ can be computed in $O(n^{d+1})$ time by adapting an algorithm of Skryzalin [27] as described 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, 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 $N = O(n)$.

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 R is taken as the polynomial ring.

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

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

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

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

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 $M_{\mathbf{u}}$ indexed by $\mathbf{u} = (u_1, u_2, \ldots, u_d) \in \mathbb{Z}^d$, i.e. $M = \bigoplus_{\mathbf{u}} M_{\mathbf{u}}$, such that the ring action satisfies that $\forall i, \forall \mathbf{u} \in \mathbb{Z}^d$, $t_i \cdot M_{\mathbf{u}} \subseteq M_{\mathbf{u}+e_i}$, where $\{e_i\}_{i=1}^d$ is the standard basis in $\mathbb{Z}^d$. The indices $\mathbf{u} \in \mathbb{Z}^d$ are called *grades*.

Another interpretation of graded module is that, for each $\mathbf{u} \in \mathbb{Z}^d$, the action of t_i on $M_{\mathbf{u}}$ determines a linear map $t_i\bullet : M_{\mathbf{u}} \to M_{\mathbf{u}+e_i}$ by $(t_i\bullet)(m) = t_i \cdot m$. So, we can also describe a graded module equivalently as a collection of vectors spaces $\{M_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{Z}^d}$ with a collection of linear maps $\{t_i\bullet : M_{\mathbf{u}} \to M_{\mathbf{u}+e_i}, \forall i, \forall \mathbf{u}\}$ where the commutative property $(t_j\bullet) \circ (t_i\bullet) = (t_i\bullet) \circ (t_j\bullet)$ 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 $M_{\mathbf{u}}$ is the $\mathbf{k}$-vector space generated by $\mathbf{t}^{\mathbf{u}} = t_1^{u_1} t_2^{u_2} \cdots t_d^{u_d}$ 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.

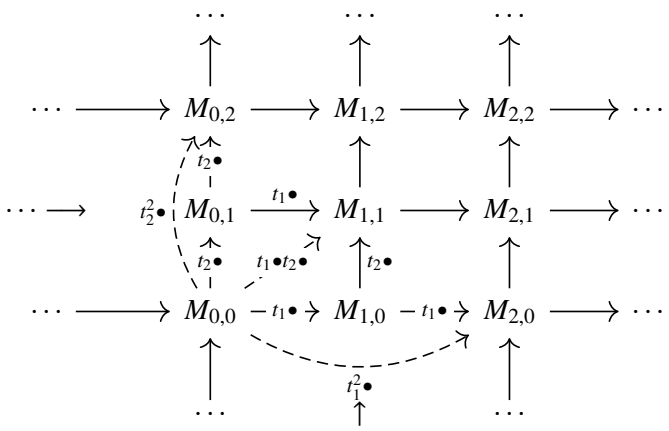


Figure 14.4: A graded 2-parameter module. All sub-diagrams of maps and compositions of maps are commutative.

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

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 R-module M a d-parameter persistence module when $M_{\mathbf{u}}$ for each $\mathbf{u} \in \mathbb{Z}^{\mathbf{d}}$ 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 $\{g_1, \cdots, g_n\} \subseteq M$ such that each element $m \in M$ can be written as an R-linear combination of these elements, i.e. $m = \sum_{i=1}^{n} \alpha_i g_i$ with $\alpha_i \in R$. 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 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 $\{v_b, v_r, v_g\}$ 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 R-linear map $f : M \to N$ preserving grades: $f(M_{\mathbf{u}}) \subseteq N_{\mathbf{u}}, \forall \mathbf{u} \in \mathbb{Z}^d$. Equivalently, it can also be described as a collection of linear maps $\{f_{\mathbf{u}} : M_{\mathbf{u}} \to N_{\mathbf{u}}\}$ which gives the following commutative diagram for each $\mathbf{u}$ and i:

$$\begin{array}{ccc} M_{\mathbf{u}} & \xrightarrow{t_i} & M_{\mathbf{u}+e_i} \\ {\scriptstyle f_{\mathbf{u}}}\downarrow & & \downarrow{\scriptstyle f_{\mathbf{u}+e_i}} \\ N_{\mathbf{u}} & \xrightarrow{t_i} & N_{\mathbf{u}+e_i} \end{array}$$

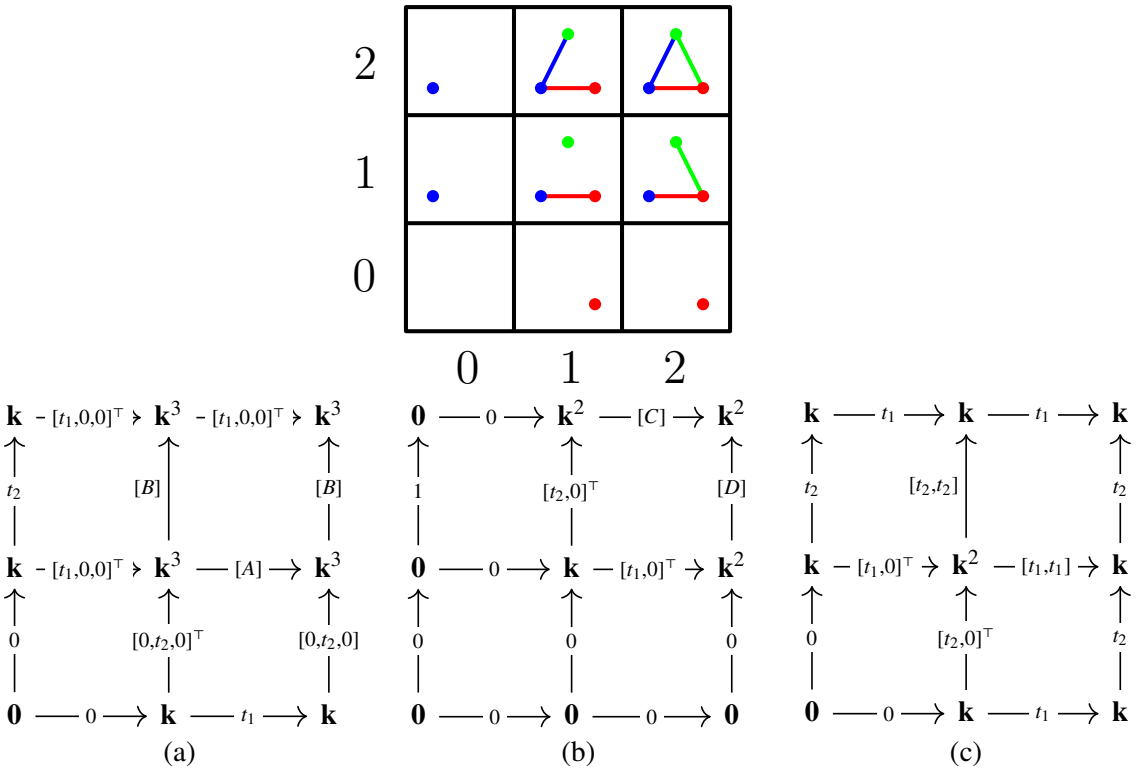


Figure 14.5: (top) An example of a 2-parameter simplicial filtration. Each square box indicates what is the current (filtered) simplical complex at the bottom left grid point of the box. (bottom) We show different modules considering different abelian groups arising out of the complexes with the ring 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 $f : M \to N$ and $g : N \to M$ such that $g \circ f$ and $f \circ g$ are identity maps. We denote it as $M \simeq N$.

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

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

Definition 9 (Homogeneous element)**.** We say an element $m \in M$ is *homogeneous* if $m \in M_{\mathbf{u}}$ for some $\mathbf{u} \in \mathbb{Z}^d$. We denote $\mathrm{gr}(m) = \mathbf{u}$ as the *grade* of such homogeneous element. To emphasize

the grade of a homogeneous element, we also write $m^{\mathrm{gr}(m)} := m$.

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 $F \simeq \bigoplus_j R^j$ 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 $\bigoplus_j R^j =< e_j : j = 1, 2, \cdots >$.

14.2 Presentations of persistence modules

Definition 10 (Decomposition)**.** For a finitely generated module M, we call $M \simeq \bigoplus M^i$ a *decomposition* of M for some collection of modules $\{M^i\}$. We say M is *indecomposable* if $M \simeq M^1 \oplus M^2 \implies M^1 = \mathbf{0}$ or $M^2 = \mathbf{0}$ where $\mathbf{0}$ denotes a trivial module. By the Krull-Schmidt theorem [2], there exists an essentially unique (up to permutation and isomorphism) decomposition $M \simeq \bigoplus M^i$ 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 $< e_1^{(0,0)} >$ and the free module $R_{\to(0,1)} \oplus R_{\to(1,0)}$ is generated by $< e_1^{(0,1)}, e_2^{(1,0)} >$. A free module M generated by $< e_j^{\mathbf{u}_j} : j = 1, 2, \cdots >$ has a (total) decomposition $M \simeq \bigoplus_j R_{\to \mathbf{u}_j}$.

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

$$\begin{array}{ccc} M & \xrightarrow{f} & N \\ \overset{g}{\underset{\simeq}{\downarrow}} & & \overset{h}{\underset{\simeq}{\downarrow}} \\ M' & \xrightarrow{f'} & N' \end{array}$$

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 $\mathbb{0} : \mathbf{0} \to \mathbf{0}$. 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 $f \simeq f_1 \oplus f_2 \implies$ f_1 or f_2 is the trivial morphism $\mathbb{0} : \mathbf{0} \to \mathbf{0}$. We call $f \simeq \bigoplus f_i$ 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.

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

$$F^1 \xrightarrow{f} F^0 \xrightarrow{g} H \longrightarrow \mathbf{0} \quad \text{where } F^1, F^0 \text{ are free.}$$

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 $\operatorname{im} f = \ker g$ and $\operatorname{im} g = H$. The double arrows on the second map in the sequence signifies the surjection of g. It follows that $\operatorname{coker} f \simeq H$ 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 $f' \simeq f \oplus f''$ with* $\operatorname{coker} f'' = \mathbf{0}$*. 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 $F^1 \xrightarrow{f} F^0 \to H$, 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 $\partial_1 : \mathsf{C}_1 \to \mathsf{C}_0$ 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 $\mathbf{A}$ is an equivalent

matrix $\mathbf{A}'$ in the following form (see formal Definition 15 later):

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{A}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{A}_k \end{bmatrix}$$

All nonzero entries are in $\mathbf{A}_i$'s and we write $\mathbf{A} \simeq \bigoplus \mathbf{A}_i$. 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]).

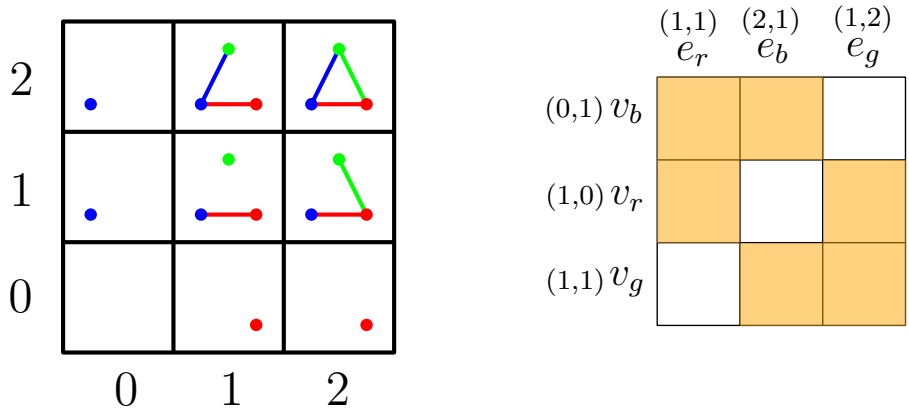


Figure 14.6: The presentation matrix of the module H_0 consisting of zero dimensional homology groups for the example in Figure 14.5. The boxes in the matrix containing non-zero entries are shaded.

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.*

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* $\mathrm{coker} f_i = H^i, \forall i$;
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* $\mathrm{coker} f' = \mathbf{0}$. *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.*

$$[f] = \begin{array}{c|c|c} f^* & g & h \\ \hline [f^*] & & \\ \hline & \begin{matrix} 1 & & \\ & 1 & \\ & & 1 \end{matrix} & \end{array}$$

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 $\mathbf{k} = \mathbb{Z}_2$ for simplicity though the method can be generalized for any finite field (Exercise 8). 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 $\mathbf{A} = [\mathbf{A}_{i,j}]$, with row indices $\mathsf{Row}(\mathbf{A}) = [\ell] := \{1, 2, \cdots, \ell\}$ and column indices $\mathsf{Col}(\mathbf{A}) = [m] := \{1, 2, \cdots, m\}$, we define an *index block* B of $\mathbf{A}$ as a pair $[\mathsf{Row}(B), \mathsf{Col}(B)]$ with $\mathsf{Row}(B) \subseteq \mathsf{Row}(\mathbf{A}), \mathsf{Col}(B) \subseteq \mathsf{Col}(\mathbf{A})$. We say an index pair (i, j) is in B if $i \in \mathsf{Row}(B)$ and $j \in \mathsf{Col}(B)$, denoted as $(i, j) \in B$. We denote a *block* of $\mathbf{A}$ on B as the matrix restricted to the index block B, i.e. $[\mathbf{A}_{i,j}]_{(i,j)\in B}$, denoted as $\mathbf{A}|_B$. We call B the index of the block $\mathbf{A}|_B$. We abuse the notations $\mathsf{Row}(\mathbf{A}|_B) := \mathsf{Row}(B)$ and $\mathsf{Col}(\mathbf{A}|_B) := \mathsf{Col}(B)$. For example, the ith row $r_i = \mathbf{A}_{i,*} = \mathbf{A}|_{[\{i\},\mathsf{Col}(\mathbf{A})]}$ and the jth column $c_j = \mathbf{A}_{*,j} = \mathbf{A}|_{[\mathsf{Row}(\mathbf{A}),\{j\}]}$ are blocks with indices $[\{i\}, \mathsf{Col}(\mathbf{A})]$ and $[\mathsf{Row}(\mathbf{A}), \{j\}]$ respectively. Specifically, $[\emptyset, \{j\}]$ represents an index block of a single column j and $[\{i\}, \emptyset]$ represents an index block of a single row i. We call $[\emptyset, \emptyset]$ the empty index block.

A matrix can have multiple equivalent forms for the same morphism they represent. We use $\mathbf{A}' \sim \mathbf{A}$ 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 $\mathbf{A}' \sim \mathbf{A}$ is called a *diagonalization of* $\mathbf{A}$ with a set of nonempty disjoint index blocks $\mathcal{B} = \{B_1, B_2, \cdots, B_k\}$ if rows and columns of $\mathbf{A}$ are partitioned into these blocks, i.e., $\mathsf{Row}(\mathbf{A}) = \coprod_i \mathsf{Row}(B_i)$ and $\mathsf{Col}(\mathbf{A}) = \coprod_i \mathsf{Col}(B_i)$, and all the nonzero entries of $\mathbf{A}'$ have indices in some B_i ($\coprod_i$ denotes disjoint union). We write $\mathbf{A}' = \bigoplus_{B_i \in \mathcal{B}} \mathbf{A}'|_{B_i}$. We say $\mathbf{A}' = \bigoplus_{B_i \in \mathcal{B}} \mathbf{A}'|_{B_i}$ is *total* if no block in this diagonalization can be diagonalized further into smaller nonempty blocks. That means, for each block $\mathbf{A}'|_{B_i}$, there is no nontrivial diagonalization. Specifically, when $\mathbf{A}$ is a null column matrix (the presentation matrix of a free module), we say $\mathbf{A}$ is itself a total diagonalization with index blocks $\{[\{i\}, \varnothing] \mid i \in \mathsf{Row}(\mathbf{A})\}$.

Note that each nonempty matrix $\mathbf{A}$ has a trivial diagonalization with the set of index blocks being the singleton $\{(\mathsf{Row}(\mathbf{A}), \mathsf{Col}(\mathbf{A}))\}$. 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 $\mathbf{A}$ is denoted generically as $\mathbf{A}^*$. All total diagonalizaitons of $\mathbf{A}$ 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.

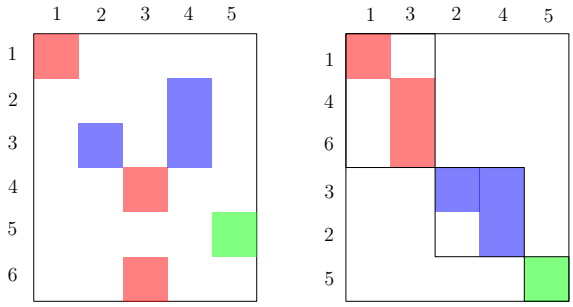


Figure 14.7: (left) A nontrivial diagonalization where the locations of non-zero entries are colored and the color for all such entries in the same block are the same. (right) The same matrix with permutation of columns and rows to bring entries of a block in adjacent locations, the three index blocks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

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 $\mathbf{k}$ 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,

$$\begin{array}{ccc} F^1 & \xrightarrow{f} & F^0 \\ {\scriptstyle h^1}\downarrow{\scriptstyle \simeq} & & \downarrow{\scriptstyle h^0}{\scriptstyle \simeq} \\ F^1 & \xrightarrow{f'} & F^0 \end{array}$$

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 $[f] \simeq [f']$. 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 $F^0 =< g_1, \cdots, g_\ell >$ and $F^1 =< s_1, \cdots, s_m >$ 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, $[f]_{i,j} = t_1^{u_1} t_2^{u_2} \cdots t_d^{u_d}$ where $(u_1, u_2, \cdots, u_d) = \mathrm{gr}(s_j) - \mathrm{gr}(g_i)$. Writing $\mathbf{u} = (u_1, u_2, \cdots, u_d)$ and $\mathbf{t}^{\mathbf{u}} = t_1^{u_1} t_2^{u_2} \cdots t_d^{u_d}$, we get $[f]_{i,j} = \mathbf{t}^{\mathbf{u}}$ where $\mathbf{u} = \mathrm{gr}(s_j) - \mathrm{gr}(g_i)$ called the grade of $[f]_{i,j}$. We call such presentation matrix *homogeneous presentation matrix*.

For example, given $F^0 =< g_1^{(1,1)}, g_2^{(2,2)} >$, 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, $g_2^{(2,2)} \leftarrow g_2^{(2,2)} + \mathbf{t}^{(1,1)} g_1^{(1,1)}$ is a homogeneous change with $\mathrm{gr}(g_2^{(2,2)} + \mathbf{t}^{(1,1)} g_1^{(1,1)}) = \mathrm{gr}(g_2^{(2,2)}) = (2,2)$, which results in a new homogeneous basis, $\{g_1^{(1,1)}, g_2^{(2,2)} + \mathbf{t}^{(1,1)} g_1^{(1,1)}\}$. Homogeneous basis changes always result in homogeneous bases.

Let $[f]$ be a homogeneous presentation matrix of $f : F^1 \to F^0$ with bases $F^0 =< g_1, \cdots, g_\ell >$ and $F^1 =< s_1, \cdots, s_m >$. 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, $\mathrm{gr}(r_i) := \mathrm{gr}(g_i)$ and $\mathrm{gr}(c_j) := \mathrm{gr}(s_j)$. We define a strict partial order $<_{\mathrm{gr}}$ on rows $\{r_i\}$ by asserting $r_i <_{\mathrm{gr}} r_j$ if and only if $\mathrm{gr}(r_i) < \mathrm{gr}(r_j)$. 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. $\mathrm{gr}([f]_{i,j}) = \mathrm{gr}(c_j) - \mathrm{gr}(r_i)$
2. a nonzero entry $[f]_{i,j}$ can only be zeroed out by column operations from columns $c_k <_{\mathrm{gr}} c_j$ or by row operations from rows $r_\ell >_{\mathrm{gr}} r_i$.

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 $\mathbf{A} = [f]$ consisting of entries in $\mathbf{k}[t_1, \cdots, t_d]$ with grading on rows and columns, find a total diagonalization of $\mathbf{A}$ under the following *admissible* row and column operations:

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

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]Recall that an element $m \in M$ is homogeneous with grade $\mathrm{gr}(m) = \mathbf{u}$ for some $\mathbf{u} \in \mathbb{Z}^d$ if $m \in M_{\mathbf{u}}$.

In fact, the values of nonzero entries in the matrix are redundant under the homogeneous property $\mathrm{gr}(\mathbf{A}_{i,j}) = \mathrm{gr}(c_j) - \mathrm{gr}(r_i)$ given by observation (1). So, we can further simplify the matrix by replacing all the nonzero entries with their $\mathbf{k}$-coefficients. For example, we can replace $2{\cdot}\mathbf{t}^{\mathbf{u}}$ with 2. What really matters are the partial orders defined by the grading of rows and columns. With our assumption of $\mathbf{k} = \mathbb{Z}_2$, 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 $\mathbf{k}$-valued matrix $\mathbf{A}$ with a partial order on rows and columns, find a total diagonalization $\mathbf{A}^* \sim \mathbf{A}$ with the following *admissible operations*:

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

The assumption of $\mathbf{k} = \mathbb{Z}_2$ 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

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

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

Proposition 3. *If* $(i,j),(j,k) \in$ Colop (Rowop) *then* $(i,k) \in$ 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 Colop and Rowop which might be a nontrivial subset of all basic operations.

Remark 4. *There are two extreme but trivial cases: (i) there are no* $<_{\mathrm{gr}}$*-comparable pair of rows and columns. In this case,* Colop = Rowop = $\emptyset$ *and the original matrix is a trivial solution. (ii) All pairs of rows and all pairs of columns are* $<_{\mathrm{gr}}$*-comparable. Or equivalently, both* Colop *and* 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* 1×1 *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 view-point, the algorithm we present can be thought of as a generalization of the traditional persistence algorithm.*

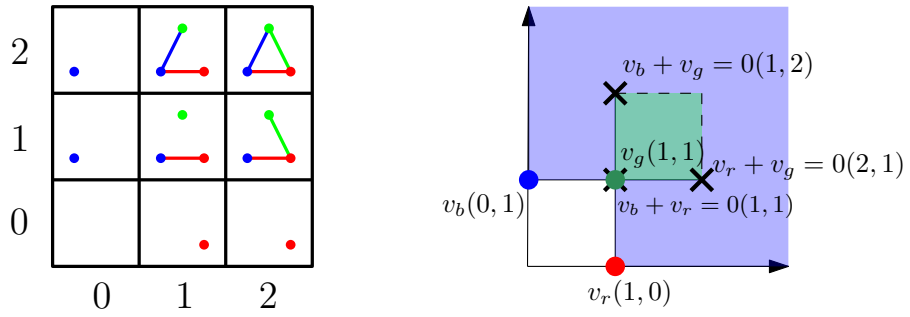


Figure 14.8: The persistence module corresponding to the presentation matrix $[\partial_1]$ shown in Example 1. The generators are given by the three vertices with grades $(0,1)$, $(1,0)$, $(1,1)$ and the relations are given by the edges with grades $(1,1)$, $(1,2)$, $(2,1)$.

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* $\partial_1 : \mathsf{C}_1 \to \mathsf{C}_0$. *With fixed bases* $\mathsf{C}_0 =< v_b^{(0,1)}, v_r^{(1,0)}, v_g^{(1,1)} >$ *and* $\mathsf{C}_1 =< e_r^{(1,1)}, e_b^{(1,2)}, e_g^{(2,1)} >$, *this presentation matrix* $[\partial_1]$ *and the corresponding binary matrix* $\mathbf{A}$ *can be written as follows (recall that superscripts indicate the grades) :*

$$
\begin{array}{c} [\partial_1] \\ v_b^{(0,1)} \\ v_r^{(1,0)} \\ v_g^{(1,1)} \end{array}
\begin{array}{c} \begin{array}{ccc} e_r^{(1,1)} & e_b^{(1,2)} & e_g^{(2,1)} \end{array} \\ \begin{pmatrix} \mathbf{t}^{(1,0)} & \mathbf{t}^{(1,1)} & 0 \\ \mathbf{t}^{(0,1)} & 0 & \mathbf{t}^{(1,1)} \\ 0 & \mathbf{t}^{(0,1)} & \mathbf{t}^{(1,0)} \end{pmatrix} \end{array}
\longrightarrow
\begin{array}{c} \mathbf{A} \\ r_1^{(0,1)} \\ r_2^{(1,0)} \\ r_3^{(1,1)} \end{array}
\begin{array}{c} \begin{array}{ccc} c_1^{(1,1)} & c_2^{(1,2)} & c_3^{(2,1)} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}
$$

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 $\mathbf{A}$ be the presentation matrix whose total diagonalization we are seeking for. We order the rows and columns of the matrix $\mathbf{A}$ according to any order that extends the partial order on the grades to a total order, e.g., dictionary order. We fix the indices $\mathsf{Row}(\mathbf{A}) = \{1, 2, \cdots, \ell\}$

and $\mathsf{Col}(\mathbf{A}) = \{1, 2, \cdots, m\}$ according to this order. With this ordering, observe that, for each admissible column operation $c_i \to c_j$, we have $i < j$, and for each admissible row operation $r_l \to r_k$, we have $l > k$.

For any column c_t, let $\mathbf{A}_{\leq t} := \mathbf{A}|_C$ denote the left submatrix on $C = [\mathsf{Row}(\mathbf{A}), \{j \in \mathsf{Col}(\mathbf{A}) \mid j \leq t\}]$ and $\mathbf{A}_{<t}$ denote its stricter version obtained by excluding column c_t from $\mathbf{A}_{\leq t}$. 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:TotDiagonalize which runs as follows (see Figure 14.9 for an illustration):

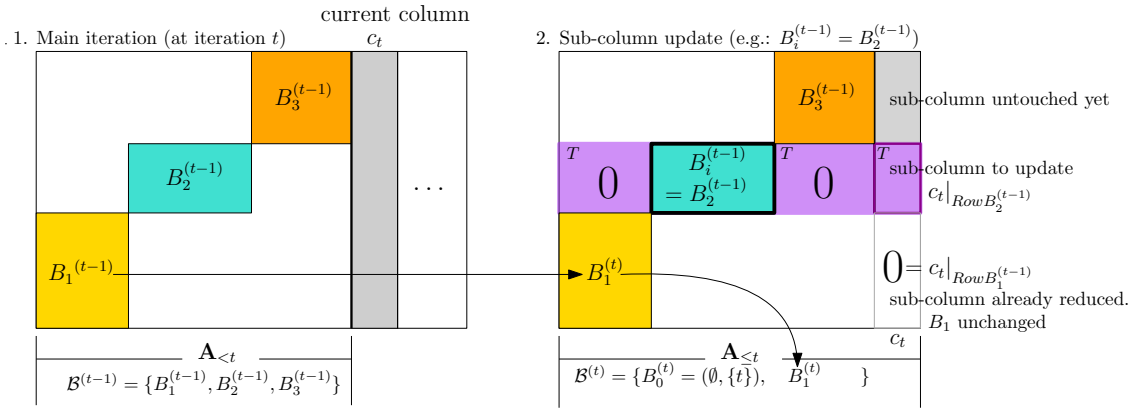


Figure 14.9: (left) $\mathbf{A}$ at the beginning of iteration t with $\mathbf{A}_{<t}$ being totally diagonalized with three index blocks $\mathcal{B}^{(t-1)} = \{B_1^{(t-1)}, B_2^{(t-1)}, B_3^{(t-1)}\}$. (right) A sub-column update step: $c_t|_{\mathsf{Row}B_1^{(t-1)}}$ has already been reduced to zero. So, $B_1^{(t)} = B_1^{(t-1)}$ is added into $\mathcal{B}^{(t)}$. White regions including $c_t|_{\mathsf{Row}B_1^{(t-1)}}$ must be preserved afterward. Now for $i = 2$, we attempt to reduce purple sub-column $c_t|_{\mathsf{Row}B_2^{(t-1)}}$. We extend it to block on $T := [\mathsf{Row}(B_2^{(t-1)}), (\mathsf{Col}(A_{\leq t}) \setminus \mathsf{Col}(B_2^{(t-1)}))]$ (colored purple) and try to reduce it in BlockReduce.

1. **Initialization**: Initialize the collection of index blocks $\mathcal{B}^{(0)} := \{B_i^{(0)} := [\{i\}, \varnothing] \mid i \in \mathsf{Row}(\mathbf{A})\}$, for the total diagonalization of the null column matrix $\mathbf{A}_{\leq 0}$.

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

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

   $$c_t|_{\mathsf{Row}B_i^{(t-1)}} := \mathbf{A}_{[\mathsf{Row}(B_i^{(t-1)}), \{t\}]},$$

   one for the set of rows $\mathsf{Row}(B_i^{(t-1)})$ for each block from the previous iteration. We process each such sub-column $c_t|_{\mathsf{Row}B_i^{(t-1)}}$ 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 $c_t|_{\mathsf{Row}B_i^{(t-1)}}$ is the left submatrix $A_{<t}$ and sub-columns $c_t|_{\mathsf{Row}B_j^{(t-1)}}$ for all $j < i$.

Prior preservation means that the operations together *change neither* $\mathbf{A}_{<t}$ *nor other sub-columns* $c_t|_{\mathsf{Row}B_j^{(t-1)}}$ *for every* $j < i$. If such operations exist, we apply them on the current $\mathbf{A}$ to get an equivalent matrix with the sub-column $c_t|_{\mathsf{Row}B_i^{(t-1)}}$ being zeroed out and we set $B_i^{(t)} \leftarrow B_i^{(t-1)}$. Otherwise, we leave the matrix $\mathbf{A}$ unchanged and add the column index t to those of $B_i^{(t-1)}$, i.e., we set $B_i^{(t)} \leftarrow [\mathsf{Row}(B_i^{(t-1)}), \mathsf{Col}(B_i^{(t-1)}) \cup \{t\}]$. After processing every sub-column $c_t|_{\mathsf{Row}B_i^{(t-1)}}$ 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 $\mathbf{A}$ with $\mathbf{A}_{\leq t}$ being totally diagonalized with index blocks $\mathcal{B}^{(t)}$.

2(b). **Block reduce**: To update the entries of each sub-column of c_t described in 2(a), we propose a block reduction algorithm ALgorithm 3:BlockReduce to compute the correct entries. Given $T := [\mathsf{Row}(B_i^{(t-1)}), (\mathsf{Col}(\mathbf{A}_{\leq t}) \setminus \mathsf{Col}(B_i^{(t-1)}))]$, this routine checks whether the block T can be zeroed out by some collection of admissible operations. If so, c_t does not join the block $B_i^{(t)}$ and $\mathbf{A}$ 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 $[\mathsf{Row}(B_1) \cup \mathsf{Row}(B_2), \mathsf{Col}(B_1) \cup \mathsf{Col}(B_2)]$. In the following algorithm, we treat the given matrix $\mathbf{A}$ to be a global variable which can be visited and modified anywhere by every subroutines called. Consequently, every time we update values on $\mathbf{A}$ by some operations, these operations are applied to the latest $\mathbf{A}$.

The outer loop is the incremental step for main iteration introducing a new column c_t which updates the diagonalization of $\mathbf{A}_{\leq t}$ 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* TotDiagonalize *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 TotDiagonalize assuming that BlockReduce routine works as claimed, namely, it checks if a sub-column of the current column c_t 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_t that have already been zeroed out.

Proposition 4. *At the end of each iteration t,* $\mathbf{A}_{\leq t}$ *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 $\mathbf{A}^{(t-1)}$ is the matrix we get at the end of iteration $(t-1)$ with $\mathbf{A}^{(t-1)}_{\leq t-1}$ being totally diagonalized. That means, $\mathbf{A}^{(t-1)}_{\leq t-1} = \mathbf{A}^{*}_{\leq t-1}$ where $\mathbf{A} = \mathbf{A}^{(0)}$ is the original given matrix. For

Algorithm 1 TotDiagonalize($\mathbf{A}$)

Input:
$\mathbf{A}$ = input matrix treated as a global variable whose columns and rows are totally ordered respecting some fixed partial order given by the grading.
Output:
A total diagonalization $\mathbf{A}^*$ with index blocks $\mathcal{B}^*$

$\mathcal{B}^{(0)} \leftarrow \{B_i^{(0)} := [\{i\}, \varnothing] \mid i \in \mathsf{Row}(\mathbf{A})\}$
for $t \leftarrow 1$ to $m := |\mathsf{Col}(\mathbf{A})|$ **do**
  $B_0^{(t)} \leftarrow [\varnothing, \{t\}]$
  for each $B_i^{(t-1)} \in \mathcal{B}^{(t-1)}$ **do**
    $T := [\mathsf{Row}(B_i^{(t-1)}), \mathsf{Col}(\mathbf{A}_{\le t}) \setminus \mathsf{Col}(B_i^{(t-1)})]$
    if BlockReduce (T)== `false` **then**
      $B_i^{(t)} \leftarrow B_i^{(t-1)} \oplus B_0^{(t)}$; *update B_i by appending t*\
    else
      $B_i^{(t)} \leftarrow B_i^{(t-1)}$; *$B_i$ remains unchanged*\
      * $\mathbf{A}$ and c_t are updated in BlockReduce when it return `true`*\
    end if
  end for
  $\mathcal{B}^{(t)} \leftarrow \{B_i^{(t)}\}$ with all $B_i^{(t)}$ containing t merged as one block.
end for
Return $(\mathbf{A}, \mathcal{B}^{(m)})$

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

Observe that all index blocks of $\mathcal{B}'$ other than B_0 are also index blocks in $\mathcal{B}^{(t-1)}$, i.e. $\mathcal{B}' \setminus \{B_0\} \subseteq \mathcal{B}^{(t-1)}$. For B_0, it can be written as $B_0 = \bigoplus_{B_j \in \mathcal{B}^{(t-1)} \setminus \mathcal{B}'} B_j \oplus [\varnothing, \{t\}]$. Let B_a be the merge of index blocks that are in $\mathbf{A}^{(t-1)}$ and also in $\mathbf{A}'$ and B_b be the merge of the rest of the index blocks of $\mathbf{A}^{(t-1)}$, i.e., $B_a = \bigoplus_{B_j \in \mathcal{B}' \cap \mathcal{B}^{(t-1)}} B_j$ and $B_b = \bigoplus_{B_j \in \mathcal{B}^{(t-1)} \setminus \mathcal{B}'} B_j$. Then B_a and B_b can be viewed as a coarser decomposition on $\mathbf{A}^{(t-1)}_{\le t-1}$ and also on $\mathbf{A}'_{\le t-1}$. By taking restrictions, we have $\mathbf{A}'|_{B_a} \sim \mathbf{A}^{(t-1)}|_{B_a}$ with equivalent transformation F_a and $\mathbf{A}'|_{B_b} \sim \mathbf{A}^{(t-1)}|_{B_b}$ 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 $\mathbf{A}'$, we can transform

$\mathbf{A}'_{\leq t-1}$ to $\mathbf{A}^{(t-1)}_{\leq t-1}$ with sub-column $[\mathsf{Row}(\mathbf{A}) \setminus \mathsf{Row}(B_0), \{t\}]$ 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 and F_b together, we get a sequence of admissible operations that reduce sub-column $[\mathsf{Row}(B_i), \{t\}]$ to zero without changing $\mathbf{A}^{(t)}_{<t}$ and also those sub-columns which have already been reduced. But, then BlockReduce would have returned 'true' signaling that B_i should not be merged with any other block required to form the block B_t reaching a contradiction. □

Now we design the BlockReduce subroutine as required. With the requirement of prior preservation, observe that reducing the sub-column $c_t|_{\mathsf{Row}B}$ for some $B \in \mathcal{B}^{(t-1)}$ is the same as reducing $T = [\mathsf{Row}(B), (\mathsf{Col}(\mathbf{A}_{\leq t}) \setminus \mathsf{Col}(B))]$ 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 $\mathbf{A}_{\leq t}$, these operations only change entries in T 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 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 to 0 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 $c_i \to c_j$, let

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

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

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

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

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

Observation 1. *Within $\mathbf{A}_{\leq t}$, an independent column or row operation only changes entries on T.*

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

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

Proof. $[\delta_{k,l}]\cdot\mathbf{A}\cdot[\delta_{i,j}] = \mathbf{A}_{l,i}[\delta_{k,j}]$ (see Fig 14.10 for an illustration). By definitions of independence and T, we have $l \notin \mathsf{Row}(B), i \in \mathsf{Col}(B)$. That means they are row index and column index from

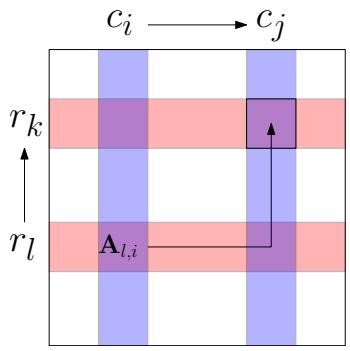


Figure 14.10: $[\delta_{k,l}]\mathbf{A}[\delta_{i,j}]$ is a matrix with the only nonzero entry at (k, j) being a copy of $\mathbf{A}_{l,i}$.

different blocks. Therefore, $\mathbf{A}_{l,i} = 0$. □

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

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

$$\mathbf{A}|_T = \sum_{\substack{l \notin \mathsf{Row}(T) \\ k \in \mathsf{Row}(T)}} \alpha_{k,l}\mathbf{X}^{k,l}|_T + \sum_{\substack{i \notin \mathsf{Col}(T) \\ j \in \mathsf{Col}(T)}} \beta_{i,j}\mathbf{Y}^{i,j}|_T$$

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

The full proof can be seen in [10]. Here, we give some intuitive explanation. Reducing the target block $\mathbf{A}|_T$ to 0 is equivalent to finding matrices $\mathbf{P}$ and $\mathbf{Q}$ encoding sequences of admissible row operations and admissible column operations respectively so that $\mathbf{PAQ}|_T = 0$. For $\Leftarrow$ direction, we can build $\mathbf{P} = \mathbf{I} + \sum \alpha_{k,l}[\delta_{k,l}]$ and $\mathbf{Q} = \mathbf{I} + \sum \beta_{i,j}[\delta_{i,j}]$ 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 $\mathbf{PAQ}$ indeed reduces $\mathbf{A}|_T$ 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 $\mathbf{A}|_T$ to 0 implies the existence of a transformation reducing $\mathbf{A}|_T$ to 0 by independent operations, we are done. This is formally proved in [10].

We can view $\mathbf{A}|_T, \mathbf{Y}^{i,j}|_T, \mathbf{X}^{k,l}|_T$ as binary vectors in the same $|T|$-dimensional space. Proposition 5 tells us that it is sufficient to check if $\mathbf{A}|_T$ can be a linear combination of the vectors corresponding to a set of independent operations. So, we first *linearize* each of the matrices $\mathbf{Y}^{i,j}|_T$'s, $\mathbf{X}^{k,l}|_T$'s, and $\mathbf{A}|_T$ to a column vector as described later (see Figure 14.11). Then, we check if $\mathbf{A}|_T$ is in the span of $\mathbf{Y}^{i,j}|_T$'s and $\mathbf{X}^{k,l}|_T$'s. This is done by collecting all vectors $\mathbf{X}^{i,j}|_T$'s and $\mathbf{Y}^{k,l}|_T$'s into a matrix S called the *source matrix* (Figure 14.11(right)) and then reducing the vector $c := \mathbf{A}|_T$ 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 $(\mathbf{S}, c)$ which reduces the column c w.r.t. the input matrix $\mathbf{S}$ by reducing the matrix $[\mathbf{S}|c]$ altogether by MatPersistence that we described previously (Topic 5).

If $c = \mathbf{A}|_T$ can be reduced to 0, we apply the corresponding independent operations to update $\mathbf{A}$. Observe that all column operations used in reducing $\mathbf{A}|_T$ together only change the sub-column

$c_t|_{\mathsf{Row}(B)}$ while row operations may change $\mathbf{A}$ to the right of the column t. We say this procedure *reduces c with $\mathbf{S}$.*

Algorithm 2 ColReduce($\mathbf{S}, c$)

Input:
Source matrix $\mathbf{S}$ and target column c to reduce
Output:
Reduced column c with $\mathbf{S}$

1: $\mathbf{S}' \leftarrow [\mathbf{S}|c]$
2: Call MatPersistence($\mathbf{S}'$);
3: **return** c along with indices of columns in $\mathbf{S}$ 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* $[\mathbf{S}|c]$ *is reduced to* $[\mathbf{S}'|0]$ *if and only if* ColReduce($\mathbf{S}, c$) *returns a zero vector.*

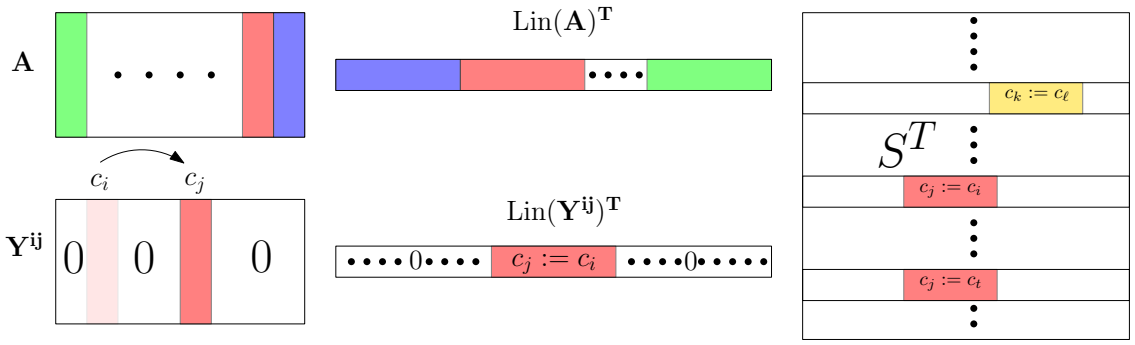


Figure 14.11: (top) Matrix $\mathbf{A}$ is linearized to the vector $\mathsf{Lin}(\mathbf{A})$ in middle; (bottom) the column operation $c_i \to c_j$ is captured by $\mathbf{Y}^{ij}$ whose linearization is illustrated in middle; (right) source matrix S combining all operations (row operations not shown). In the picture, $(\cdot)^T$ denotes transposed matrices.

Now we describe the linearization used in in Algorithm 3:BlockReduce. We fix a linear order $\leq_{\mathsf{Lin}}$ on the set of matrix indices, $\mathsf{Row}(\mathbf{A}) \times \mathsf{Col}(\mathbf{A})$, as follows: $(i, j) \leq_{\mathsf{Lin}} (i', j')$ if $j > j'$ or $j = j', i < i'$. Explicitly, we linearly order the indices as:

$$((1, m), (2, m), \ldots, (\ell, n), (1, m-1), (2, m-1), \ldots).$$

For any index block B, let $\mathsf{Lin}(\mathbf{A}|_B)$ be the vector of dimension $|\mathsf{Col}(B)| \cdot |\mathsf{Row}(B)|$ obtained by linearizing $\mathbf{A}|_B$ to a vector in the above linear order on the indices.

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

Time complexity. First we analyze the time complexity of 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 $\mathbf{A}$ is assumed to be a global variable
Output:
A boolean to indicate whether $\mathbf{A}|_T$ can be reduced and reduced block $\mathbf{A}|_T$ if possible.

Compute $c := \mathsf{Lin}(\mathbf{A}|_T)$ and initialize empty matrix $\mathbf{S}$
for each admissible column operation $c_i \to c_j$ with $i \notin \mathsf{Col}(T)$, $j \in \mathsf{Col}(T)$ **do**
    compute $\mathbf{Y}^{i,j}|_T := (\mathbf{A}\cdot[\delta_{i,j}])|_T$ and $y^{j,j} = \mathsf{Lin}(\mathbf{Y}^{i,j}|_T)$; update $\mathbf{S} \leftarrow [\mathbf{S}|y^{i,j}]$
end for
for each admissible row operation $r_l \to r_k$ with $l \notin \mathsf{Row}(T), k \in \mathsf{Row}(T)$ **do**
    compute $\mathbf{X}^{k,l}|_T := ([\delta_{k,l}]\cdot\mathbf{A})|_T$ and $x^{k,l} := \mathsf{Lin}(\mathbf{X}^{k,l}|_T)$; update $\mathbf{S} \leftarrow [\mathbf{S}|x^{k,l}]$
end for
COLREDUCE $(\mathbf{S}, c)$ returns indices of $y^{i,j}$'s and $x^{k,l}$'s used to reduce c (if possible)
For every returned index of $y^{i,j}$ or $x^{k,l}$ apply $c_i \to c_j$ or $r_l \to r_k$ to transform $\mathbf{A}$
return $\mathbf{A}|_T == 0$

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 $O(NN_B) \times O(N^2)$ 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 $\mathbf{S}$ 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 $[\mathbf{S}|c]$ 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 ∂_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, $N = O(n^{d-1})$ and a presentation matrix can be computed in $O(n^{d+1})$ time giving a total time complexity of $O(n^{(2\omega+1)(d-1)})$. We discuss the details in Section 14.5.

14.4.1 Running TotDiagonalize on the working example in Figure 14.5

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

$$\begin{array}{c} \mathbf{A} \\ r_1^{(0,1)} \\ r_2^{(1,0)} \\ r_3^{(1,1)} \end{array} \begin{array}{c} \begin{array}{ccc} c_1^{(1,1)} & c_2^{(1,2)} & c_3^{(2,1)} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

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

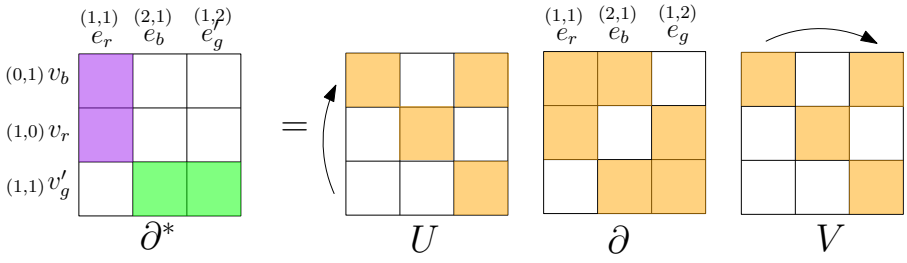


Figure 14.12: Diagonalizing the binary matrix given in Example 1. It can be viewed as multiplying the original matrix ∂ with a left matrix U that represents the row operation and a right matrix V that represents the column operations.

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

$$\begin{array}{c} \mathbf{S} \\ \end{array} \begin{array}{c} \begin{array}{c|c} \mathsf{Lin}(([\delta_{1,3}]\mathbf{A})|_T) & c{=}\mathsf{Lin}(\mathbf{A}|_T) \end{array} \\ \left[\begin{array}{c|c} 0 & 1 \end{array}\right] \end{array}$$

The result of ColReduce$(\mathbf{S}, c)$ *stays the same as its input. That means we cannot reduce c at all. Therefore,* 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 $r_3 \to r_2$ 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 $\mathbf{A}|_T = \mathbf{A}|_{(\{3\},\{1\})} = 0$. That means 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 $\mathcal{B}^{(1)}$: $(\{1, 2\}, \{1\})$ and $(\{3\}, \varnothing)$.*

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, $r_3 \to r_1, r_3 \to r_2, c_1 \to c_2$. BlockReduce$(T)$ *constructs the target vector $c = \mathsf{Lin}(\mathbf{A}|_T)$ and the source matrix $\mathbf{S} = \{\mathsf{Lin}(([\delta_{1,3}]\mathbf{A})|_T), \mathsf{Lin}(([\delta_{2,3}]\mathbf{A})|_T), \mathsf{Lin}((\mathbf{A}[\delta_{1,2}])|_T)\}$ illustrated as follows:*

$$\begin{array}{c|ccc|c} \mathbf{S} & \mathsf{Lin}(([\delta_{1,3}]\mathbf{A})|_T) & \mathsf{Lin}([(\delta_{2,3}]\mathbf{A})|_T) & \mathsf{Lin}((\mathbf{A}[\delta_{1,2}])|_T) & c{=}\mathsf{Lin}(\mathbf{A}|_T) \\ \hline \begin{matrix}\\ \\ \end{matrix} & \begin{matrix}1\\0\end{matrix} & \begin{matrix}0\\1\end{matrix} & \begin{matrix}1\\1\end{matrix} & \begin{matrix}1\\0\end{matrix} \end{array}$$

The result of ColReduce$(\mathbf{S}, c)$ *is*

$$\begin{array}{ccc|c} \multicolumn{3}{c|}{\mathbf{S}} & c \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}$$

So the BlockReduce *updates $\mathbf{A}|_T$ to get the following updated matrix:*

$$\begin{array}{ll} \mathbf{A}' & \begin{array}{ccc} c_1^{(1,1)} & c_2^{(1,2)} & c_3^{(2,1)} \end{array} \\ \begin{array}{l} r_1^{(0,1)} + r_3^{(1,1)} \\ r_2^{(1,0)} \\ r_3^{(1,1)} \end{array} & \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

and returns TRUE *since $\mathbf{A}'|_T == 0$. Therefore, we do not merge B_1. We continue to check for the block $B_2 = (\{3\},\emptyset)$ and $T = (\{3\},\{1,2\})$, whether $\mathbf{A}'|_T$ can be reduced to zero. There is no admissible operation for this block at all. Therefore, the matrix stays the same and* BlockReduce *returns* FALSE. *We merge $B_2 \oplus B_0 = (\{3\},\{2\})$.*

Continuing the process for the last column c_3 in the third iteration $t = 3$, we see that $B_1 = (\{1,2\},\{1\}), B_2 = (\{3\},\{2\})$ and $B_0 = (\emptyset,\{3\})$. For the block $B_1 = (\{1,2\},\{1\})$, the target block we hope to zero out is $T = (\{1,2\},\{2,3\})$. There are four admissible operations from outside of T into T, $r_3 \to r_1, r_3 \to r_2, c_1 \to c_2, c_1 \to c_3$. BlockReduce$(T)$ *constructs the target vector $c = \mathsf{Lin}(\mathbf{A}|_T)$ and the source matrix $\mathbf{S} = \{\mathsf{Lin}(([\delta_{1,3}]\mathbf{A})|_T), \mathsf{Lin}(([\delta_{2,3}]\mathbf{A})|_T), \mathsf{Lin}((\mathbf{A}[\delta_{1,2}])|_T)\}, \mathsf{Lin}((\mathbf{A}[\delta_{1,3}])|_T)\}$ illustrated as follows:*

$$\begin{array}{c|cccc|c} \mathbf{S} & \mathsf{Lin}(([\delta_{1,3}]\mathbf{A})|_T) & \mathsf{Lin}([(\delta_{2,3}]\mathbf{A})|_T) & \mathsf{Lin}((\mathbf{A}[\delta_{1,2}])|_T) & \mathsf{Lin}((\mathbf{A}[\delta_{1,3}])|_T) & c{=}\mathsf{Lin}(\mathbf{A}|_T) \\ \hline \begin{matrix}\\ \\ \\ \\ \end{matrix} & \begin{matrix}1\\0\\1\\0\end{matrix} & \begin{matrix}0\\1\\0\\1\end{matrix} & \begin{matrix}0\\0\\1\\1\end{matrix} & \begin{matrix}1\\1\\0\\0\end{matrix} & \begin{matrix}1\\1\\0\\0\end{matrix} \end{array}$$

The result of ColReduce$(\mathbf{S}, c)$ *is*

$$\begin{array}{cccc|c} \multicolumn{4}{c|}{\mathbf{S}} & c \\ \hline 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{array}$$

So the BlockReduce *updates* $\mathbf{A}|_T$ *to get the following updated matrix:*

$$\begin{array}{l} \mathbf{A}' \\ r_1^{(0,1)} \\ r_2^{(1,0)} + r_3^{(1,1)} \\ r_3^{(1,1)} \end{array} \begin{array}{c} \begin{array}{ccc} c_1^{(1,1)} & c_2^{(1,2)} + c_1^{(1,1)} & c_3^{(2,1)} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \end{array}$$

and returns true *since* $\mathbf{A}'|_T == 0$. *Therefore, we do not merge* B_1 *with any other block. We continue to check for the block* $B_2 = (\{3\}, \{2\})$ *and* $T = (\{3\}, \{1, 3\})$, *whether* $\mathbf{A}'|_T$ *can be reduced to zero. There is no admissible operation for this block at all. Therefore, the matrix stays the same and* BlockReduce *returns* false. *We merge* $B_2 \oplus B_0 = (\{3\}, \{2, 3\})$.

Finally the algorithm returns the matrix $\mathbf{A}'$ *shown above as the final result. It is the correct total diagonalization with two index blocks in* $\mathcal{B}^{\mathbf{A}^*}$*:* $B_1 = (\{1, 2\}, \{1\})$ *and* $B_2 = (\{3\}, \{2, 3\})$. *An examination of* ColReduce$(\mathbf{S}, c)$ *in all three iterations over columns reveals that the entire matrix* $\mathbf{A}$ *is updated by operations* $r_3 \to r_2$ *and* $c_1 \to c_2$. *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.*

$$\begin{array}{l} [\partial_1] \\ v_b^{(0,1)} \\ v_r^{(1,0)} \\ v_g^{(1,1)} \end{array} \begin{array}{c} \begin{array}{ccc} e_r^{(1,1)} & e_b^{(1,2)} & e_g^{(2,1)} \end{array} \\ \begin{pmatrix} \mathbf{t}^{(1,0)} & \mathbf{t}^{(1,1)} & 0 \\ \mathbf{t}^{(0,1)} & 0 & \mathbf{t}^{(1,1)} \\ 0 & \mathbf{t}^{(0,1)} & \mathbf{t}^{(1,0)} \end{pmatrix} \end{array}$$

$\Longrightarrow$

$$\begin{array}{l} [\partial_1]^* \\ v_b^{(0,1)} \\ v_r^{(1,0)} \\ v_g^{(1,1)} + \mathbf{t}^{(0,1)} v_r^{(1,0)} \end{array} \begin{array}{c} \begin{array}{ccc} e_r^{(1,1)} & e_b^{(1,2)} + \mathbf{t}^{(0,1)} e_r^{(1,1)} & e_g^{(2,1)} \end{array} \\ \begin{pmatrix} \mathbf{t}^{(1,0)} & 0 & 0 \\ \mathbf{t}^{(0,1)} & 0 & 0 \\ 0 & \mathbf{t}^{(0,1)} & \mathbf{t}^{(1,0)} \end{pmatrix} \end{array}$$

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 $\mathsf{C}_{p+1} \to \mathsf{Z}_p \twoheadrightarrow \mathsf{H}_p \to \mathbf{0}$ 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 $\{X_\mathbf{u}\}_{\mathbf{u} \in \mathbb{Z}^d}$ such that for each grade $\mathbf{u} \in \mathbb{Z}^d$ and each $i = 1, \cdots, d$, $X_\mathbf{u} \subseteq X_{\mathbf{u}+e_i}$.

14.5.1 Graded chain, cycle, and boundary modules

We obtain a simplicial chain complex $(\mathsf{C}_\bullet(X_\mathbf{u}), \partial_\bullet)$ for each $X_\mathbf{u}$ in the given simplicial filtration. For each comparable pairs in the grading $\mathbf{u} \le \mathbf{v} \in \mathbb{Z}^d$, a family of inclusion maps $\mathsf{C}_\bullet(X_\mathbf{u}) \hookrightarrow \mathsf{C}_\bullet(X_\mathbf{v})$ is induced by the canonical inclusion $X_\mathbf{u} \hookrightarrow X_\mathbf{v}$ giving rise to the following diagram:

$$\begin{array}{ccccccccccc}
\mathsf{C}_\bullet(X_{\mathbf{u}}): & \cdots & \xrightarrow{\partial_{p+2}} & \mathsf{C}_{p+1}(X_{\mathbf{u}}) & \xrightarrow{\partial_{p+1}} & \mathsf{C}_p(X_{\mathbf{u}}) & \xrightarrow{\partial_p} & \mathsf{C}_{p-1}(X_{\mathbf{u}}) & \xrightarrow{\partial_{p-1}} & \cdots \\
\downarrow & & & \downarrow & & \downarrow & & \downarrow & & \\
\mathsf{C}_\bullet(X_{\mathbf{v}}): & \cdots & \xrightarrow{\partial_{p+2}} & \mathsf{C}_{p+1}(X_{\mathbf{v}}) & \xrightarrow{\partial_{p+1}} & \mathsf{C}_p(X_{\mathbf{v}}) & \xrightarrow{\partial_p} & \mathsf{C}_{p-1}(X_{\mathbf{v}}) & \xrightarrow{\partial_{p-1}} & \cdots
\end{array}$$

For each chain complex $\mathsf{C}_\bullet(X_{\mathbf{u}})$, we have the cycle spaces $\mathsf{Z}_p(X_{\mathbf{u}})$'s and boundary spaces $\mathsf{B}_p(X_{\mathbf{u}})$'s as kernels and images of boundary maps ∂_p's respectively, and the homology group $\mathsf{H}_p(X_{\mathbf{u}}) = \mathsf{Z}_p(X_{\mathbf{u}})/\mathsf{B}_p(X_{\mathbf{u}})$ as the cokernel of the inclusion maps $\mathsf{B}_p(X_{\mathbf{u}}) \hookrightarrow \mathsf{Z}_p(X_{\mathbf{u}})$. In line with category theory we use the notations im , ker, 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:

$$\begin{array}{ccccccc}
 & & & & \mathsf{B}_p(X_{\mathbf{u}}) & \hookrightarrow & \mathsf{Z}_p(X_{\mathbf{u}}) \xrightarrow{-\operatorname{coker}} \mathsf{H}_p(X_{\mathbf{u}}) \\
 & & & \overset{\operatorname{im}\partial_{p+1}}{\nearrow} & & & \searrow \\
\cdots & \mathsf{C}_{p+1}(X_{\mathbf{u}}) & & \xrightarrow{\qquad\partial_{p+1}\qquad} & & & \mathsf{C}_p(X_{\mathbf{u}}) \cdots \\
 & & & \underset{\ker\partial_p}{\searrow} & & & \nearrow
\end{array}$$

In the language of graded modules, for each p, the family of vector spaces and linear maps (inclusions) $(\{\mathsf{C}_p(X_{\mathbf{u}})\}_{\mathbf{u}\in\mathbb{Z}^d}, \{\mathsf{C}_p(X_{\mathbf{u}}) \hookrightarrow \mathsf{C}_p(X_{\mathbf{v}})\}_{\mathbf{u}\le\mathbf{v}})$ can be summarized as a $\mathbb{Z}^d$-graded R-module:

$$\mathsf{C}_p(X) := \bigoplus_{\mathbf{u}\in\mathbb{Z}^d} \mathsf{C}_p(X_{\mathbf{u}}), \text{ with the ring action } t_i \cdot \mathsf{C}_p(X_{\mathbf{u}}) : \mathsf{C}_p(X_{\mathbf{u}}) \hookrightarrow \mathsf{C}_p(X_{\mathbf{u}+e_i})\ \forall i,\ \forall \mathbf{u}.$$

That is, the ring R acts as the linear maps (inclusions) between pairs of vector spaces in $\mathsf{C}_p(X_\bullet)$ with comparable grades. It is not too hard to check that this $\mathsf{C}_p(X_\bullet)$ is indeed a graded module. Each p-chain in a chain space $\mathsf{C}_p(X_{\mathbf{u}})$ is a homogeneous element with grade $\mathbf{u}$.

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

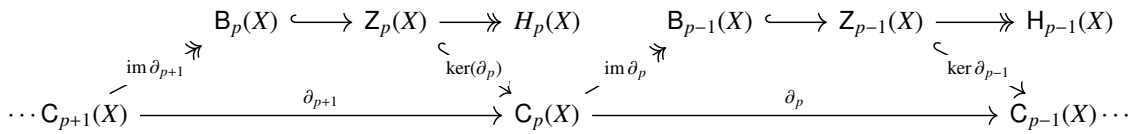
The kernel and image of a graded module morphism are also graded modules as submodules of domain and codomain respectively whereas the cokernel is a quotient module of the codomain. They can also be defined grade-wise in the expected way:

$$\text{For } f : M \to N, (\ker f)_{\mathbf{u}} = \ker f_{\mathbf{u}}, (\operatorname{im} f)_{\mathbf{u}} = \operatorname{im} f_{\mathbf{u}}, (\operatorname{coker} f)_{\mathbf{u}} = \operatorname{coker} f_{\mathbf{u}}.$$

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 $\partial_p : \mathsf{C}_p(X) \to \mathsf{C}_{p-1}(X)$ is the family of cycle spaces $\mathsf{Z}_p(X)$ and family of boundary spaces $\mathsf{B}_{p-1}(X)$ respectively with linear maps induced by inclusions. Also, from the inclusion induced morphism $\mathsf{B}_p(X) \hookrightarrow \mathsf{Z}_p(X)$, we have the cokernel module $\mathsf{H}_p(X)$, consisting of homology groups $\bigoplus_{\mathbf{u}\in\mathbb{Z}^d} \mathsf{H}_p(X_{\mathbf{u}})$ and linear maps induced from inclusion maps $X_{\mathbf{u}} \hookrightarrow X_{\mathbf{v}}$ for each comparable pairs $\mathbf{u} \le \mathbf{v}$. This $\mathsf{H}_p(X)$ 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 $t_1 \cdot M_{\mathbf{u}} \subseteq M_{\mathbf{u}+e_1}$ signifies the linear map $M_{\mathbf{u}} \to M_{\mathbf{v}}$ between homology groups induced by the inclusion of the complex at $\mathbf{u}$ into the complex at $\mathbf{v} = \mathbf{u} + e_1$.

[3] e.g. $\ker\partial_p$ denotes the inclusion of Z_p into C_p

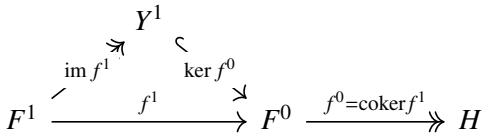
In our case, we have chain complex of graded modules and induced homology groups which can be succinctly described by the following diagram:



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 ∂_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 $F^1 \to F^0 \twoheadrightarrow H \to \mathbf{0}$. To reveal further details of a presentation of H, we recognize that it respects the following commutative diagram,



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: $\mathsf{C}_1 \xrightarrow{\partial_1} \mathsf{C}_0 \xrightarrow{\operatorname{coker}\partial_1} \mathsf{H}_0$
- Presentation matrix = $[\partial_1]$ is given as part of the input.

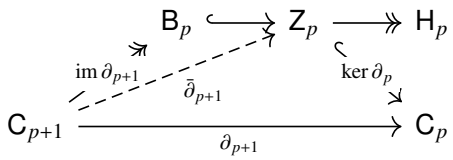
Justification. For $p = 0$, the cycle module $\mathsf{Z}_0 = \mathsf{C}_0$ is a free module. So we have the presentation of H_0 as claimed. It is easy to check that $\partial_1 : \mathsf{C}_1 \to \mathsf{C}_0$ is a presentation of H_0 since both C_1 and C_0 are free modules. With standard basis of chain 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 ∂_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 $\mathsf{Z}_p \twoheadrightarrow \mathsf{H}_p$ from free module Z_p can be naturally chosen as a generating map in the presentation of H_p. In this case we have:

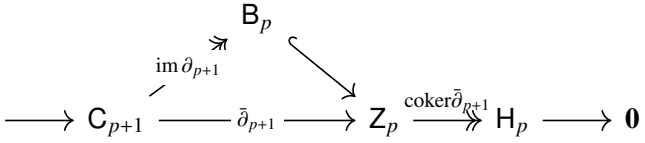
- Presentation: $\mathsf{C}_{p+1} \xrightarrow{\bar{\partial}_{p+1}} \mathsf{Z}_p \xrightarrow{\text{coker}\bar{\partial}_{p+1}} \mathsf{H}_p$ where $\bar{\partial}_{p+1}$ is the induced map from the diagram:



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

  1. Compute a basis $G(\mathsf{Z}_p)$ for the free module Z_p where $G(\mathsf{Z}_p)$ is presented as a set of generators in the basis of C_p. This can be done by an algorithm in [22]. Take $G(\mathsf{Z}_p)$ as the row basis of the presentation matrix $[\bar{\partial}_{p+1}]$.
  2. Present $\text{im}\,\partial_{p+1}$ in the basis of $G(\mathsf{Z}_p)$ to get the presentation matrix $[\bar{\partial}_{p+1}]$ of the induced map as follows. Originally, $\text{im}\,\partial_{p+1}$ 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 $G(\mathsf{Z}_p)$ computed in the previous step. This can be done as follows. Let $[G(\mathsf{Z}_p)]$ denote the matrix presenting basis elements in $G(\mathsf{Z}_p)$ 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 $[G(\mathsf{Z}_p)]$ and note the columns that are added to c. These columns provide the necessary presentation of c in the basis $G(\mathsf{Z}_p)$. This reduction can be done by the persistent algorithm described earlier.

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



Here the $\bar{\partial}_{p+1}$ is an induced map from ∂_{p+1}. With a fixed basis on Z_p and standard basis of C_{p+1}, we rewrite the presentation matrix $[\partial_{p+1}]$ to get $[\bar{\partial}_{p+1}]$, which constitutes a valid input to our decomposition algorithm.

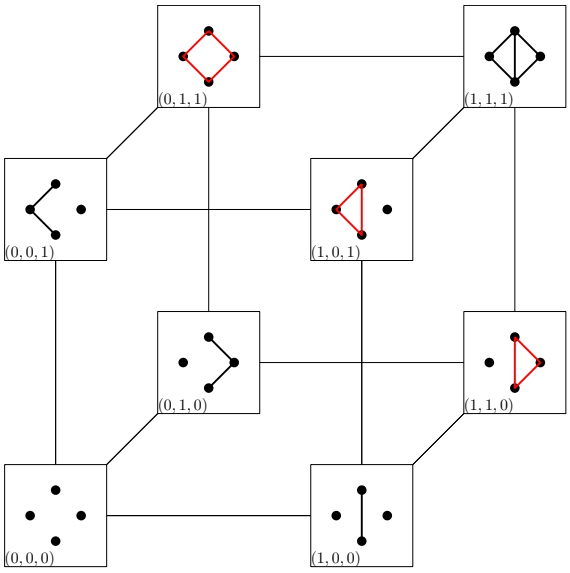


Figure 14.13: An example of a filtration of simplicial complex for $d = 3$ with non-free Z when $p = 1$. The three red cycles are three generators in Z_1. However, at grading $(1, 1, 1)$, the earliest time these three red cycles exist simultaneously, there is a relation among these three generators.

14.5.4 $d > 2$-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 Z is not necessarily free when $d > 2$. The issue caused by non-free Z is that, if we use the same presentation matrix as we did in the previous case with free Z, we may lose some relations coming from the inner relations of a generating set of 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 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: $\mathbf{t}^{(1,0,0)} g_1^{(0,1,1)} + \mathbf{t}^{(0,1,0)} g_2^{(1,0,1)} + \mathbf{t}^{(0,0,1)} g_3^{(1,1,0)} = 0$. Although $\operatorname{im} \partial_1 = 0$, we still have a nontrivial relation from Z. So, we have $\mathsf{H}_1 =< g_1^{(0,1,1)}, g_2^{(1,0,1)}, g_3^{(1,1,0)} : s^{(1,1,1)} = \mathbf{t}^{(1,0,0)} g_1^{(0,1,1)} + \mathbf{t}^{(0,1,0)} g_2^{(1,0,1)} + \mathbf{t}^{(0,0,1)} g_3^{(1,1,0)} >$. The presentation matrix turns out to be the following:

$$\begin{matrix} & s^{(1,1,1)} \\ g_1^{(0,1,1)} & \mathbf{t}^{(1,0,0)} \\ g_2^{(1,0,1)} & \mathbf{t}^{(0,1,0)} \\ g_3^{(1,1,0)} & \mathbf{t}^{(0,0,1)} \end{matrix}$$

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 $\partial_1 : \mathsf{C}_1 \to \mathsf{C}_0$ 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 $G(\mathsf{Z}_p)$ that is presented in the basis of C_p. This is done by the algorithm of Lesnick and Wright [22] which runs in $O(n^3)$ time. Using $[G(\mathsf{Z}_p)]$, we compute the presentation matrix $[\bar{\partial}_{p+1}]$ as described in section 14.5.3. This can be done in $O(n^3)$ time assuming that $G(\mathsf{Z}_p)$ has at most $O(n)$ elements. The presentation matrix is decomposed with 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 $G(\mathsf{Z}_p)$ has $O(n)$ elements. This follows from the fact that Z_p being a free submodule of C_p cannot have a rank larger than that of C_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, $d \geq 2$, multi-dimensional homology: For d-parameter persistence modules where $d \geq 2$ (this subsumes the previous case), an algorithm that runs in time $O(n^{d+1})$ and produces a presentation matrix of dimensions $O(n^{d-1}) \times O(n^{d-1})$ can be designed using a result of Skryzalin [27]; see [10]. Plugging $N = O(n^{d-1})$ and taking the computation of presentation matrix into consideration, we get a time complexity bound of $O(n^{d+1}) + O(n^{(2\omega+1)(d-1)}) = O(n^{(2\omega+1)(d-1)})$.

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 isomophic. Persistence diagrams for 1-parameter tame persistence modules are such invaraints. 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 graded module M is finitely generated as before and additionally finitely supported. For this we need to define the support of M.

Definition 17 (Support)**.** Let M be a $\mathbb{Z}^d$-graded module. Its support is defined as the graph $\text{supp}(M) = (V, E \subseteq V \times V)$ where a node $\mathbf{v} \in V$ if and only if $M_{\mathbf{v}} \neq \mathbf{0}$ and an edge $(\mathbf{u}, \mathbf{v}) \in E$ if

and only if (i) $\mathbf{u} < \mathbf{v}$ and there is no $\mathbf{s} \in \mathbb{Z}^d$ satisfying $\mathbf{u} < \mathbf{s} < \mathbf{v}$, and (ii) $\text{rank}(M_\mathbf{u} \to M_\mathbf{v}) \neq 0$. We say M is finitely supported if $\text{supp}(M)$ is finite.

Fact 2. $\text{supp}(M)$ *is disconnected if there exist two grades* $\mathbf{u} < \mathbf{v}$ *in* $\text{supp}(M)$ *so that* $\text{rank}(M_\mathbf{u} \to M_\mathbf{v}) = 0$.

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 $\mathbf{u} \not\leq \mathbf{v}$, the linear maps between $M_\mathbf{u}$ and $M_\mathbf{v}$ are not defined. In the following definitions, we take them as zero maps.

Definition 18 (Rank invariant)**.** Let $r_{\mathbf{uv}} = \text{rank}(M_\mathbf{u} \to M_\mathbf{v})$ for every pair $\mathbf{u}, \mathbf{v} \in \text{supp}(M)$ with $\mathbf{u} \leq \mathbf{v}$. The collection $\{r_{\mathbf{uv}}\}$ is called the *rank invariant* of M.

Fact 3. *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* $\beta_p^{i,j}$ *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 19 (Multirank invariant)**.** The collection $\{r_{\mathbf{UV}}\}$ for every pair $\mathbf{U} \subseteq \text{supp}(M)$ and $\mathbf{V} \subseteq \text{supp}(M)$ where $r_{\mathbf{UV}} = \text{rank}\left(\bigoplus_{\mathbf{u}\in\mathbf{U}} M_\mathbf{u} \to \bigoplus_{\mathbf{v}\in\mathbf{V}} M_\mathbf{v}\right)$ is called the *multirank invariant* of M.

We can retrieve the information about birth and death of generators from the multirank. For a grade $\mathbf{u}$, define its immediate predecessors $P_\mathbf{u}$ and immediate successors $S_\mathbf{u}$ as:

$$P_\mathbf{u} = \{\mathbf{u}' \in \text{supp}(M) \,|\, \mathbf{u}' < \mathbf{u} \text{ and } \nexists \mathbf{u}'' \text{ with } \mathbf{u}' < \mathbf{u}'' < \mathbf{u}\}$$
$$S_\mathbf{u} = \{\mathbf{u}' \in \text{supp}(M) \,|\, \mathbf{u}' > \mathbf{u} \text{ and } \nexists \mathbf{u}'' \text{ with } \mathbf{u} < \mathbf{u}'' < \mathbf{u}'\}.$$

Fact 4.

1. *We have that m generators get born at grade* $\mathbf{u}$ *if and only if* $\text{coker}\left(\bigoplus_{\mathbf{u}'\in P_\mathbf{u}} M_{\mathbf{u}'} \to M_\mathbf{u}\right)$ *has dimension m.*
2. *We have that m generators die leaving grade* $\mathbf{u}$ *if and only if* $\ker\left(M_\mathbf{u} \to \bigoplus_{\mathbf{u}'\in S_\mathbf{u}} M_{\mathbf{u}'}\right)$ *has dimension m.*

Although multirank 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 I is connected and for every $\mathbf{u}, \mathbf{v} \in I$, if $\mathbf{u} < \mathbf{w} < \mathbf{v}$, then $\mathbf{w} \in I$. We call a persistence module with support on an interval an *interval module* if $M_\mathbf{u}$ is unit dimensional for each vertex $\mathbf{u} \in \text{supp}(M)$. 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 5. *Two interval decomposable modules are isomorphic if and only if they have the same multirank invariants.*

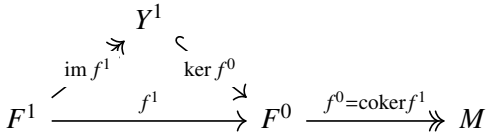
14.6.2 Graded Betti numbers and blockcodes

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].

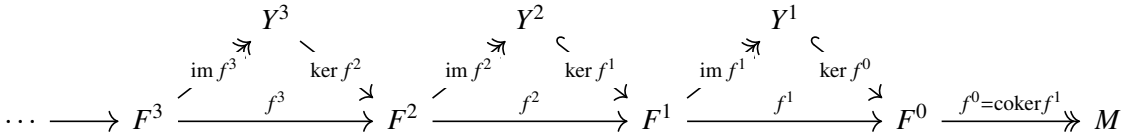
Definition 20 (Free resolution)**.** For a graded module M, a free resolution $\mathcal{F} \to M$ is an exact sequence:

$\cdots \longrightarrow F^2 \xrightarrow{f^2} F^1 \xrightarrow{f^1} F^0 \xrightarrow{f^0} M \longrightarrow \mathbf{0}$ 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 $\operatorname{im} f^2 = \ker f^1$, which implies $\operatorname{coker} f^2 \cong \operatorname{im} f^1 = \ker f^0 = Y^1$. 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.



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 $F^0 = \bigoplus_{i=1}^{n} R_{\to \operatorname{gr}(g_i)}$ with standard basis $e_1^{\operatorname{gr}(g_1)}, \cdots, e_n^{\operatorname{gr}(g_n)}$ 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, $Y_1 \xhookrightarrow{\ker f^0} F^0$, 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 $F^1 = \bigoplus_{j=1}^{m} R_{\to \operatorname{gr}(y_j)}$ with standard basis $e_1'^{\operatorname{gr}(y_1)}, \cdots, e_m'^{\operatorname{gr}(y_m)}$ of F^1. The homogeneous R-map $f^1 : F^1 \to F^0$ is determined by $f^1(e_j') = y_j$.

By repeating this procedure for $Y_2 = \ker f^1$ and moving backward further, one gets a graded free resolution of M.

Definition 21 (Graded Betti numbers)**.** Let F^j be a free module in the minimal free resolution of a graded module M. Let $\beta^M_{j,\mathbf{u}}$ be the multiplicity of each grade $\mathbf{u} \in \mathbb{Z}^d$ 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.

β^M	(1,0)	(0,1)	(1,1)	(2,1)	(1,2)	(2,2)	$\cdots$
β_0	1	1	1				
β_1			1	1	1		
β_2						1	
$\beta_{\geq 3}$							

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

Definition 22 (Persistent graded Betti numbers)**.** Let $M \simeq \bigoplus M^i$ be a total decomposition of a graded module M. We have for each indecomposable M^i, the refined graded Betti numbers $\beta^{M^i} = \{\beta^{M^i}_{j,\mathbf{u}} \mid j \in \mathbb{N}, \mathbf{u} \in \mathbb{Z}^d\}$. We call the set $\mathcal{PB}(M) := \{\beta^{M^i}\}$ 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.

β^{M^1}	(1,0)	(0,1)	(1,1)	(2,1)	(1,2)	(2,2)	$\cdots$
β_0	1	1					
β_1			1				
$\beta_{\geq 2}$							
β^{M^2}	(1,0)	(0,1)	(1,1)	(2,1)	(1,2)	(2,2)	$\cdots$
β_0			1				
β_1				1	1		
β_2						1	
$\beta_{\geq 3}$							

Table 14.2: Persistence grades $\mathcal{PB}(M) = \{\beta^{M^1}, \beta^{M^2}\}$. 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:

$$\mathrm{dm}M : \mathbb{Z}^d \to \mathbb{Z}_{\geq 0} \quad \mathrm{dm}M(\mathbf{u}) = \dim(M_{\mathbf{u}})$$

Fact 6. *There is a relation between the graded Betti numbers and dimension function of a persistence module as follows:*

$$\forall \mathbf{u} \in \mathbb{Z}^d,\ \mathrm{dm}M(\mathbf{u}) = \sum_{\mathbf{v} \le \mathbf{u}} \sum_{j} (-1)^j \beta_{j,\mathbf{v}}$$

Then for each indecomposable M^i, we have the dimension function $\mathrm{dm}M^i$ related to persistent graded Betti numbers restricted to M^i.

Definition 23 (Blockcode)**.** The set of dimension functions $\mathcal{B}_{\mathrm{dm}}(M) := \{\mathrm{dm}M^i\}$ is called the *blockcode* of M.

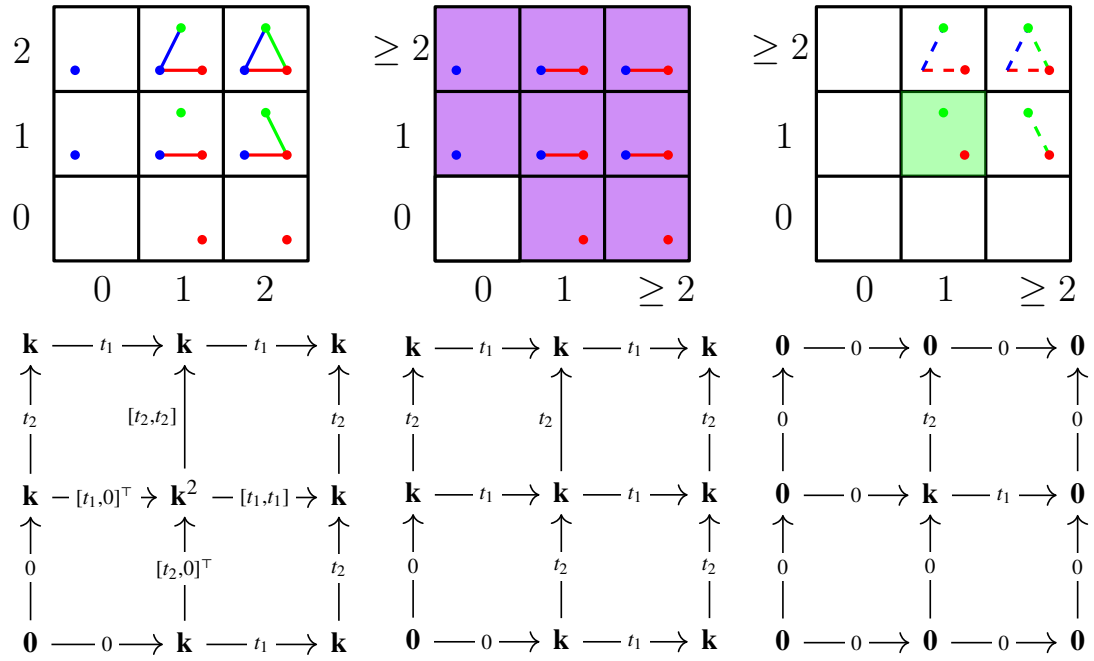


Figure 14.14: (top) 2-parameter simplicial filtration for our working example in Figure 14.5. $\mathrm{dm}M^1$ and $\mathrm{dm}M^2$: each colored square represents an 1-dimensional vector space $\mathbf{k}$ and each white square represents a 0-dimensional vector space. In the middle picture M^1 is generated by $v_b^{0,1}, v_r^{1,0}$ which are drawn as a blue dot and a red dot respectively. They are merged at $(1,1)$ by the red edge e_r. In the right picture, M^2 is generated by $v_g^{(1,1)} + \mathbf{t}^{(0,1)} v_r^{1,0}$ which is represented by the combination of the green circle and the red circle together at $(1,1)$. After this point $(1,1)$, the generator is mod out to be zero by relation of e_g starting at $(2,1)$, represented by the green dashed line segment, and by relation of $e_b + \mathbf{t}^{(0,1)} e_r$ starting at $(1,2)$, represented by the blue dashed line segment connected with the red dashed line segment.

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

$$\mathrm{dm}M^1(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \ge (1,0) \text{ or } \mathbf{u} \ge (0,1) \\ 0 & \text{otherwise} \end{cases} \qquad \mathrm{dm}M^2(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} = (1,1) \\ 0 & \text{otherwise} \end{cases} \tag{14.2}$$

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 $\mathrm{dm}M^1$, 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 $\mathrm{dm}M^2$, 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 $\mathbb{Z}_q[t_1 t_2 \dots t_d]$.

Under suitable finiteness condition, the fact that persistence modules are indeed finitely presented graded modules over multivariate polynoimials 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 geometry 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 indecomposalbe 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 $\mathbf{A}$ 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.

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

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

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

11. 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).

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

Bibliography

[1] Hideto Asashiba, Emerson G. Escolar, Yasuaki Hiraoka, and Hiroshi Takeuchi. Matrix method for persistence modules on commutative ladders of finite type. *Journal of Industrial and Applied Mathematic*, 36(1):97–130, 2019.

[2] Michael Atiyah. On the krull-schmidt theorem with application to sheaves. *Bulletin de la Société Mathématique de France*, 84:307–317, 1956.

[3] Winfried Bruns and H Jürgen Herzog. *Cohen-macaulay rings*. Cambridge university press, 1998.

[4] James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

[5] Gunnar Carlsson, Gurjeet Singh, and Afra Zomorodian. Computing multidimensional persistence. In *International Symposium on Algorithms and Computation*, pages 730–739. Springer, 2009.

[6] Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, Jul 2009.

[7] René Corbet and Michael Kerber. The representation theorem of persistence revisited and generalized. *Journal of Applied and Computational Topology*, 2(1):1–31, Oct 2018.

[8] David A Cox, John Little, and Donal O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[9] Tamal K. Dey and Cheng Xin. Computing bottleneck distance for 2-d interval decomposable modules. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 32:1–32:15, 2018.

[10] Tamal K. Dey and Cheng Xin. Generalized persistence algorithm for decomposing multi-parameter persistence modules, 2019. arXiv:1904.03766.

[11] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

[12] David Eisenbud. *The geometry of syzygies: a second course in algebraic geometry and commutative algebra*, volume 229. Springer Science & Business Media, 2005.

[13] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, 2002.

[14] David Hilbert. Über die theorie der algebraischen formen. *Mathematische Annalen*, 36:473–530, 1890.

[15] Derek F Holt. The meataxe as a tool in computational group theory. *London Mathematical Society Lecture Note Series*, pages 74–81, 1998.

[16] Derek F Holt and Sarah Rees. Testing modules for irreducibility. *Journal of the Australian Mathematical Society*, 57(1):1–16, 1994.

[17] Oscar H Ibarra, Shlomo Moran, and Roger Hui. A generalization of the fast lup matrix decomposition algorithm and applications. *Journal of Algorithms*, 3(1):45 – 56, 1982.

[18] Woojin Kim and Facundo Mémoli. Generalized persistence diagrams for persistence modules over posets. *CoRR*, arXiv:1810.11517, 2018.

[19] Kevin P Knudson. A refinement of multi-dimensional persistence. *arXiv preprint arXiv:0706.2608*, 2007.

[20] Michael Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015.

[21] Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *CoRR*, abs/1512.00180, 2015.

[22] Michael Lesnick and Matthew Wright. Computing Minimal Presentations and Betti Numbers of 2-Parameter Persistent Homology. *arXiv e-prints*, page arXiv:1902.05708, Feb 2019.

[23] Ezra Miller and Bernd Sturmfels. *Combinatorial Commutative Algebra*. 2004.

[24] Richard A Parker. The computer calculation of modular characters (the meat-axe). *Computational group theory*, pages 267–274, 1984.

[25] Amit Patel. Generalized persistence diagrams. *J. Appl. Comput. Topology*, 1:397–419, 2018.

[26] Tim Römer. On minimal graded free resolutions. *Illinois J. Math*, 45(2):1361–1376, 2001.

[27] Jacek Skryzalin. Numeric invariants from multidimensional persistence, 2016. PhD thesis, Stanford University.