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

COMPUTATIONAL DECOMPOSITION OF FILTERED CHAIN COMPLEXES

AXEL HASENKAMP

THESIS FOR THE ATTAINMENT OF THE ACADEMIC DEGREE

BACHELOR OF SCIENCE

AT THE TECHNICAL UNIVERSITY OF MUNICH

SUPERVISOR:

PROF. DR. RER. NAT. ULRICH BAUER

SUBMITTED:

MUNICH, 15 JUNE 2024

I hereby declare that this thesis is entirely the result of my own work except when otherwise indicated. I have only used the resources given in the list of references.

Axel Hasenkamp
Munich, 15 June 2024

ZUSAMMENFASSUNG

Die persistente Homologie eines gefilterten Kettenkomplexes zerfällt in eine direkte Summe von Intervall-Persistenzmodulen. Einerseits können wir uns für den Barcode der persistenten Homologie interessieren. Andererseits können wir uns für eine Zerlegung des gefilterten Kettenkomplexes interessieren, d.h. eine Basis mit Erzeugern von Homologie und deren berandenden Ketten als Basisvektoren. Wir stellen einen Algorithmus vor, welcher beides berechnet. Persistente relative Kohomologie weist strukturelle Ähnlichkeiten zur persistenten Homologie auf; vor allem berechnet der eben erwähnte Algorithmus, angewandt in persistenter relativer Kohomologie, einen Barcode und eine Basis, die in den Barcode der persistenten Homologie und eine Zerlegung des gefilterten Kettenkomplexes übersetzt werden können. Die letztgenannte Übersetzung ist jedoch nicht möglich, wenn der Algorithmus mit dem Ziel, nur den Barcode zurückzugeben, optimiert wird.

Wir führen den Begriff der U-Match-Zerlegung einer Randmatrix ein. Aus dieser kann eine Zerlegung des gefilterten Kettenkomplexes abgelesen werden. Wir führen zwei Algorithmen ein, die große strukturelle Ähnlichkeiten mit Algorithmen zur Berechnung persistenter relativer Kohomologie aufweisen, aus deren Output wir eine U-Match-Zerlegung rekonstruieren können. So können wir auch bei den obigen optimierten Ansätzen eine Zerlegung des gefilterten Kettenkomplexes berechnen.

Wir führen Computerexperimente durch, die die zusätzlichen Ressourcen veranschaulichen, welche für die Durchführung dieser Rekonstruktion erforderlich sind.

ABSTRACT

Persistent homology of a filtered chain complex decomposes into a direct sum of interval persistence modules. On one hand, we may be interested in the barcode of persistent homology. On the other hand, we may be interested in a basis which we refer to as a decomposition of the filtered chain complex, i.e., basis vectors being generators of homology and their bounding chains. We discuss an algorithm that computes both. Persistent relative cohomology shares structural similarities with persistent homology; most importantly, said algorithm applied in persistent relative cohomology computes a barcode and a basis, which can be translated into the barcode of persistent homology and a decomposition of the filtered chain complex. The latter translation not being feasible when optimizing the algorithm with the aim to only return the barcode, however.

We introduce the notion of a U-match decomposition of a boundary matrix, from which a decomposition of the filtered chain complex may be read off. We discuss two algorithms that share great structural similarities with algorithms computing persistent relative cohomology, but from whose output we may reconstruct a U-match decomposition. Thus, even in above mentioned optimized approaches to computing the barcode, we are able to compute a decomposition of the filtered chain complex.

We conduct computational experiments that illustrate the additional resources needed in order to conduct said reconstruction.

CONTENTS

1 Introduction 1
2 Preliminaries 2
2.1 Notation 2
2.2 Persistent homology 3
2.3 Computing persistent homology 5
2.4 Persistent relative cohomology 7
2.5 Motivation 10
3 U-match decompositions 11
3.1 Introducing U-match decompositions 11
3.2 Inner identities 14
3.3 Reconstructing U-match decompositions 17
4 Decomposition algorithms 19
5 Optimizations 23
6 Implementation 24
7 Experiments 26
Appendix 27
References 30

1 INTRODUCTION

Persistent homology of a finite-dimensional filtered chain complex of vector spaces decomposes into a direct sum of interval persistence modules. We may, on one hand, be interested in the corresponding barcode of persistent homology and, on the other hand, in a basis of the filtered chain complex, which fulfills a certain requirement, and which we refer to as a decomposition of the filtered chain complex. This requirement is that the basis consists of generators of each interval of the barcode and their bounding chains. We are interested in computationally decomposing filtered chain complexes.

In Section 2, we will introduce the relevant notion of a filtered chain complex and its persistent homology, and, furthermore, the barcode of the latter and the type of basis which we refer to as a decomposition of the former. We introduce an algorithm that computes both of these. We then introduce the persistence module of persistent cohomology of a filtered chain complex, whose barcode is identical to that of persistent homology. We also introduce the persistence module of persistent relative homology, whose barcode is in bijection to that of persistent homology as well. Lastly, we introduce persistent relative cohomology and illustrate structural similarities between it and persistent homology; the most relevant consequence of that being that above algorithm applied in persistent relative cohomology computes, at least equally well, a barcode and a basis, which can be translated into the barcode of persistent homology and a decomposition of the filtered chain complex. The latter translation potentially being prohibitively computationally expensive and not even feasible when optimizing above algorithm with the aim to only return the barcode, then serves as a motivation for the rest of our discussion.

In Section 3, we introduce the notion of a U-match decomposition of an arbitrary matrix. If this matrix is the boundary matrix of a filtered chain complex, we can read off from the U-match decomposition a decomposition of the filtered chain complex. We then show that, under reasonable restrictions, a U-match decomposition can be reconstructed from certain data; this data being what is actually saved in above mentioned optimized approaches to computing the barcode.

In Section 4, we illustrate two algorithms that return data from which we may reconstruct a U-match decomposition of the boundary matrix of a filtered chain complex; hence a decomposition of it. The first algorithm is an adaptation of the algorithm mentioned above applied to the coboundary matrix of the filtered chain complex. The second algorithm is a compressed, i.e., optimized, version of aforementioned algorithms and structurally identical to an algorithm that aims to compute a barcode only.

In Section 5, we discuss further optimizations of the compressed algorithm, which partially depend on the matrix to which the algorithm is applied being the boundary matrix of a filtered chain complex.

As a proof of concept, we adapt a well-known software, which computes barcodes of Vietoris-Rips complexes and shares similarities with above mentioned compressed algorithm, such that it reconstructs a decomposition of the corresponding filtered chain complexes. The changes we have made to said software are discussed in Section 6.

In Section 7, we show the results of some computational experiments, where we compare the performance of aforementioned implementation computing a decomposition of filtered chain complexes to the performance when just computing their barcodes.

2 PRELIMINARIES

Adapted from [5], we introduce filtered chain complexes and their persistent homology. From there on out, we base our statements on [4]; exceptions are indicated. Persistent homology decomposes into a direct sum of interval persistence modules. There exists a decomposition of the chain complex, i.e., a basis which fits this decomposition in persistent homology. We briefly recap an algorithm that allows us to compute the barcode of persistent homology as well as aforementioned basis. We then introduce cochain complexes and the persistence modules that are persistent cohomology, persistent relative homology and persistent relative cohomology of a filtered chain complex. We illustrate structural similarities between persistent homology and persistent relative cohomology. There exists a basis of the cochain complex underlying persistent relative cohomology, which can, at least in theory, be translated into aforementioned basis in persistent homology and can be computed by the algorithm mentioned above as well. We then sketch why some computational approaches aiming to compute the barcode of persistent homology via persistent relative cohomology do not allow us to carry out this translation. This then serves as the motivation for introducing the U-match decomposition of the boundary matrix of a filtered chain complex.

2.1 NOTATION

Throughout our discussion, let $\mathbb{F}$ be a field. Given a matrix D with entries in $\mathbb{F}$, we denote the entry in the i-th row and j-th column with $D[i,j]$. If we wish to refer to the i-th row resp. j-th column of D, we write $\mathrm{ROW}_i(D)$ resp. $\mathrm{COL}_j(D)$. Given two matrices D and E, we write $D \equiv E$, if they are equal up to a permutation of rows and columns. Let D be an $m \times n$ matrix. We write $D^{\perp}$ for the matrix that is defined via $D^{\perp}[i,j] = D[n+1-j, m+1-i]$; we refer to it as the antitranspose of D. We will often be interested in matrices that are the result of first considering submatrices of D and then permuting rows and columns. We will have two conventions for this. Consider the matrix

$$D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

The first convention arises when we consider the 2×2 submatrix in the upper left-hand corner and switch both the first and second rows and columns. We write this matrix as

$$\begin{array}{cc} & \begin{array}{cc} 2 & 1 \end{array} \\ \begin{array}{c} 2 \\ 1 \end{array} & \begin{pmatrix} 5 & 4 \\ 2 & 1 \end{pmatrix} \end{array}.$$

The second convention arises when we define sequences $\boldsymbol{\rho} = (2,1)$ and $\boldsymbol{\kappa} = (2,1)$ and write $D_{\boldsymbol{\rho\kappa}}$ to denote the same matrix. For any positive integer n, we write $\mathbf{n} = (1,\ldots,n)$. I denotes the unit matrix and 0 denotes the zero matrix; their dimensions will only be stated if not clear from context. We write e_k for the k-th unit vector; whether this is a row or column vector will always be clear from context.

2.2 PERSISTENT HOMOLOGY

Throughout our discussion, we consider chain complexes of vector spaces only. This setting, for example, arises when considering a finite simplicial complex and the chain complex we obtain by considering formal sums of simplices of equal dimension with coefficients in a field as chains.

Definition 2.1. A chain complex $\mathbf{C}$ is a pair (C, ∂), where $C = \bigoplus_{i \in \mathbb{Z}} C_i$ is a graded $\mathbb{F}$-vector space and $\partial = \bigoplus_{i \in \mathbb{Z}} \partial_i : C \to C$ is a boundary operator, i.e., for each $i \in \mathbb{Z}$, $\partial_i : C_i \to C_{i-1}$ is a linear map such that $\partial^2 = \bigoplus_{i \in \mathbb{Z}} \partial_i \circ \partial_{i+1}$ is the zero map.

Let $\mathbf{C}$ be a chain complex. We restrict ourselves to chain complexes that are supported on $\mathbb{N}$, i.e., for all $i < 0, C_i = \{0\}$, and finite-dimensional, i.e., C is finite-dimensional as an $\mathbb{F}$-vector space.

We refer to ∂_i as the i-th boundary map and to a vector σ in C_i as an i-chain; i is then referred to as the dimension of σ. Let n be the dimension of C as an $\mathbb{F}$-vector space. We denote a basis of C consisting of chains by $\{\sigma_1, \ldots, \sigma_n\}$. We will often refer to $\{\sigma_1, \ldots, \sigma_n\}$ as a basis of $\mathbf{C}$ as well, or equivalently $\mathbf{C} = \langle \sigma_1, \ldots, \sigma_n \rangle$.

Definition 2.2. Let $\mathbf{C}$ be a chain complex. Consider the vector spaces $Z_i = \ker(\partial_i)$, whose vectors we refer to as i-cycles, and $B_i = \operatorname{im}(\partial_{i+1})$, whose vectors we refer to as i-boundaries. The quotient $H_i(\mathbf{C}) = Z_i / B_i$ is called the i-th homology of $\mathbf{C}$. We write $H_*(\mathbf{C}) = \bigoplus_{i \in \mathbb{Z}} H_i(\mathbf{C})$.

Remark 2.3. Based on [9], we make the following observation. Consider the short exact sequence of finite-dimensional vector spaces given by

$$\{0\} \to U \hookrightarrow V \twoheadrightarrow W \to \{0\}$$

and bases $u = (u_1, \ldots, u_k)$ resp. $w = (w_1, \ldots, w_l)$ of U resp. W. We can lift each w_i to some $\tilde{w}_i \in V$. Then $(u_1, \ldots, u_k, \tilde{w}_1, \ldots, \tilde{w}_l)$ is a basis of V. If the i-th homology of $\mathbf{C}$ is trivial, then

$$\{0\} \to B_i \hookrightarrow C_i \twoheadrightarrow B_{i-1} \to \{0\}$$

is a short exact sequence of finite-dimensional vector spaces. We might thus obtain a basis of C_i by considering bases of B_i and B_{i-1}. In general, we have

$$\{0\} \to B_i \hookrightarrow Z_i \hookrightarrow C_i \quad .$$

Recall and note that $Z_i / B_i = H_i(\mathbf{C})$ and $C_i / Z_i \cong B_{i-1}$. Let b_i, h_i and resp. b_{i-1} be bases of $B_i, H_i(\mathbf{C})$ and resp. B_{i-1}. Then b_i, h_i and lifts of b_{i-1} form a basis of C_i.

Definition 2.4. Let $\mathbf{C} = (C, \partial_{\mathbf{C}})$ be a chain complex. A chain complex $\mathbf{D} = (D, \partial_{\mathbf{D}})$ is a subcomplex of $\mathbf{C}$, if D is a subspace of C and $\partial_{\mathbf{D}}$ is a (well-defined) restriction of $\partial_{\mathbf{C}}$.

Definition 2.5. Let $\mathbf{C} = (C, \partial)$ be a chain complex. A filtration of $\mathbf{C}$ is a sequence of chain complexes

$$\mathbf{C}_\bullet : \quad \mathbf{C}_0 \hookrightarrow \mathbf{C}_1 \hookrightarrow \ldots \hookrightarrow \mathbf{C}_i \hookrightarrow \ldots \quad ,$$

where $\mathbf{C}_i$ is a subcomplex of $\mathbf{C}_{i+1}$ and $\mathbf{C}$ for each $i \in \mathbb{N}$, and the union of all $\mathbf{C}_i$ equals $\mathbf{C}$ with respect to their vector spaces. We refer to $\mathbf{C}_\bullet$ as a filtered chain complex.

Let $\mathbf{C}_\bullet$ be a filtered chain complex. We restrict ourselves to the case that $\mathbf{C}_0 = \mathbf{0}$, $\mathbf{C}_i = \langle \sigma_1, \ldots, \sigma_i \rangle$ for each $0 < i \leq n$, and $\mathbf{C}_i = \mathbf{C}$ else. Here $\mathbf{0}$ is the trivial chain complex with $\{0\}$ as its vector space. When considering finite simplicial complexes, this is equivalent to an essential simplexwise filtration, where

in each filtration step exactly one simplex is added as a basis vector. These restrictions give rise to the following diagram of chain complexes:

$$\mathbf{C}_\bullet : \quad \mathbf{0} = \mathbf{C}_0 \hookrightarrow \ldots \hookrightarrow \mathbf{C}_n = \mathbf{C} \ .$$

It is indeed reasonable to view filtered chain complexes as diagrams in a category of chain complexes, where objects are chain complexes as defined above and morphisms are chain maps as defined below, as the inclusions in the filtration sequence commute with the boundary operators by construction. We may thus interpret a filtration $\mathbf{C}_\bullet$ as a functor from the poset category of $\mathbb{N}$, referred to as $\mathbf{N}$, to a category of chain complexes.

Definition 2.6. Let $\mathbf{C} = (C, \partial_{\mathbf{C}})$ and $\mathbf{D} = (D, \partial_{\mathbf{D}})$ be chain complexes. A linear map

$$f = \bigoplus_{i \in \mathbb{Z}} f_i : C \to D$$

is a chain map, if, for each $i \in \mathbb{Z}$, $f_i : C_i \to D_i$ is a linear map such that $\partial_{\mathbf{D}} f_i = f_{i-1} \partial_{\mathbf{C}}$.

Lemma 2.7. *Let $\mathbf{C} = (C, \partial_{\mathbf{C}})$ and $\mathbf{D} = (D, \partial_{\mathbf{D}})$ be chain complexes and $f : C \to D$ a chain map. Then f induces a linear map $H_*(\mathbf{C}) \to H_*(\mathbf{D})$.*

We may thus interpret H_* as a functor from a category of chain complexes to the category of $\mathbb{F}$-vector spaces, to which we refer to as $\mathbf{Vect}$. Consider the following diagram of vector spaces by applying homology to the above diagram:

$$H_*(\mathbf{C}_\bullet) : \quad \{0\} = H_*(\mathbf{C}_0) \to \ldots \to H_*(\mathbf{C}_n) = H_*(\mathbf{C}) \ .$$

Hence, we may interpret $H_*(\mathbf{C}_\bullet)$ as a functor from $\mathbf{N}$ to $\mathbf{Vect}$. We refer to $H_*(\mathbf{C}_\bullet)$ as the persistent homology of $\mathbf{C}_\bullet$. It is a pointwise finite-dimensional persistence module as defined below.

Definition 2.8 ([3])**.** Let R be a totally ordered set and $\mathbf{R}$ its poset category. A functor

$$P_\bullet : \mathbf{R} \to \mathbf{Vect}$$

is called a persistence module. $P_\bullet$ is pointwise finite-dimensional, if, for each $r \in R$, P_r as an $\mathbb{F}$-vector space is finite-dimensional.

Theorem 2.9 ([3])**.** *Let $P_\bullet : \mathbf{R} \to \mathbf{Vect}$ be a pointwise finite-dimensional persistence module. Then $P_\bullet$ decomposes, uniquely up to isomorphism, into a direct sum of interval persistence modules, i.e., we have that*

$$P_\bullet = \bigoplus_{I \in \mathcal{I}} I \ ,$$

where I is an interval persistence module. Given an interval $S \subseteq R$, the corresponding interval persistence module I is given by $I_r = \mathbb{F}$, if $r \in S$, $I_r = \{0\}$ else, and each arrow between non-zero vector spaces being an isomorphism. We call the multiset of these intervals the barcode of $P_\bullet$, which we denote as $\mathrm{Barc}(P_\bullet)$.

Thus, as can also be seen in [10], $H_*(\mathbf{C}_\bullet)$ decomposes into a direct sum of interval persistence modules, each of which is isomorphic to an interval persistence module of form

$$\ldots \to \{0\} \to \mathbb{F} \xrightarrow{\cong} \ldots \xrightarrow{\cong} \mathbb{F} \to \{0\} \to \ldots \ .$$

Hence, we may consider $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$. Its intervals are of the form $[i, j]$ for some $i, j \in \{1, \ldots, n\}$. We interpret them as half-open intervals of form $[i, j+1)$, if $j < n$, and $[i, \infty)$ else. We will refer to

the set of these half-open intervals as $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$ as well. Note that in our restricted setting we may refer to the barcode as a set, as each filtration step either reduces or increases the dimension of some i-th homology by exactly one. Each $H_i(\mathbf{C}_\bullet)$ is a persistence module on its own. Thus, we have

$$\mathrm{Barc}(H_*(\mathbf{C}_\bullet)) = \bigsqcup_{n\in\mathbb{N}} \mathrm{Barc}(H_n(\mathbf{C}_\bullet)) \ .$$

Therefore, we may assign to each interval a dimension.

Theorem 2.10 ([6],[10])**.** *Consider the filtered chain complex*

$$\mathbf{C}_\bullet : \quad \mathbf{0} = \mathbf{C}_0 \hookrightarrow \ldots \hookrightarrow \mathbf{C}_n = \mathbf{C} \ ,$$

where $\mathbf{C}_i = \langle \sigma_1, \ldots, \sigma_i \rangle$ *for each* $0 < i \leq n$*. Then there exists a partition*

$$\{1, \ldots, n\} = \mathcal{E} \cup \mathcal{B} \cup \mathcal{D}$$

such that the following holds:

1. *There exists a bijection* $\mathcal{B} \leftrightarrow \mathcal{D}$ *such that* $b \leftrightarrow d$*, if and only if* $[b, d) \in \mathrm{Barc}(H_*(\mathbf{C}_\bullet))$.
2. *There exists a basis* $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_n\}$ *of* $\mathbf{C}$ *such that* $\mathbf{C}_i = \langle \hat{\sigma}_1, \ldots, \hat{\sigma}_i \rangle$ *for all* $i \in \{1, \ldots, n\}$.
3. $\partial \hat{\sigma}_e = 0$ *for all* $e \in \mathcal{E}$.
4. $\partial \hat{\sigma}_d = \hat{\sigma}_b$*, and thus also* $\partial \hat{\sigma}_b = 0$*, for all* $[b, d) \in \mathrm{Barc}(H_*(\mathbf{C}_\bullet))$.

Remark 2.11. Note that the basis shown in Theorem 2.10 not only is of the form sketched in Remark 2.3 but also respects the filtration sequence of the filtered chain complex. We thus refer to this basis as a decomposition of the filtered chain complex.

2.3 COMPUTING PERSISTENT HOMOLOGY

Definition 2.12. Let R be a matrix. Define $\mathrm{pivot}(\mathrm{COL}_i(R))$ as the largest row index with non-zero entry in $\mathrm{COL}_i(R)$, if well defined. We refer to it as the pivot of $\mathrm{COL}_i(R)$. R is reduced, if there do not exist distinct column indices i and j such that $\mathrm{pivot}(\mathrm{COL}_i(R)) = \mathrm{pivot}(\mathrm{COL}_j(R))$.

Let $\mathbf{C}_\bullet$ be a filtered chain complex. Given a basis $\{\sigma_1, \ldots, \sigma_n\}$ of $\mathbf{C}$, we may represent ∂ by a strictly upper-triangular matrix D, which we implicitly define via $\partial \sigma_j = \sum_{i<j} D[i, j]\sigma_i$. We refer to D as the boundary matrix of the filtered chain complex. Both $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$ and the basis $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_n\}$ from Theorem 2.10 can be computed by decomposing D as $R = DV$, where R is a reduced matrix and V is invertible and upper triangular. We refer to decompositions of form $R = DV$ as right-reductions. An algorithm for computing a right-reduction of D is given by Algorithm 1, which was introduced in [2] and which we have adapted from [5].

Given a right-reduction $R = DV$, we may read off $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$ as follows:

- We get a finite interval of form $[b, d)$, if and only if $\mathrm{pivot}(\mathrm{COL}_d(R)) = b$.
- We get an infinite interval of form $[e, \infty)$, if and only if e is not used as an interval bound in one of the finite intervals above.

Algorithm 1 Right-reduction

Require: $D \in \mathbb{F}^{n \times n}$
Ensure: $R \in \mathbb{F}^{n \times n}$ reduced and $V \in \mathbb{F}^{n \times n}$ invertible as well as upper triangular such that $R = DV$

```
R := D, V := I_{n×n}
for j ∈ {1, ..., n} in increasing order do
    while there exist i < j and k such that k = pivot(COL_i(R)) = pivot(COL_j(R)) do
        λ := R[k, j] / R[k, i]
        COL_j(R) := COL_j(R) − λ · COL_i(R)
        COL_j(V) := COL_j(V) − λ · COL_i(V)
    end while
end for
```

Furthermore, we may read off a decomposition of the filtered chain complex, i.e., a basis as in Theorem 2.10, as follows:

- A generator $\hat{\sigma}_e$ of $[e, \infty)$ is given by $\mathrm{COL}_e(V)$, that is $\hat{\sigma}_e = \sum_{i=1}^{n} V[i, e]\sigma_i$.
- A generator $\hat{\sigma}_b$ of $[b, d)$ is given by $\mathrm{COL}_d(R)$.
- A bounding chain $\hat{\sigma}_d$ of $\hat{\sigma}_b$ is given by $\mathrm{COL}_d(V)$.

Example 2.13. *As an illustrative example, which will permeate the rest of our discussion, we consider the full simplicial complex on four vertices. We consider the chain complex* $\mathbf{C}$ *that arises when taking formal sums of simplices with coefficients in the field with two elements. We consider a filtered chain complex* $\mathbf{C}_\bullet$*, where in each filtration step exactly one simplex is added as a basis vector, and which is given by the following boundary matrix:*

$$D = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{array} \\ \begin{pmatrix} & & & & 1 & 0 & 0 & 1 & 1 & 0 & & & & & \\ & & & & 1 & 1 & 0 & 0 & 0 & 1 & & & & & \\ & & & & 0 & 1 & 1 & 0 & 1 & 0 & & & & & \\ & & & & 0 & 0 & 1 & 1 & 0 & 1 & & & & & \\ & & & & & & & & & & 1 & 1 & 0 & 0 & \\ & & & & & & & & & & 1 & 0 & 0 & 1 & \\ & & & & & & & & & & 0 & 0 & 1 & 1 & \\ & & & & & & & & & & 0 & 1 & 1 & 0 & \\ & & & & & & & & & & 1 & 0 & 1 & 0 & \\ & & & & & & & & & & 0 & 1 & 0 & 1 & \\ & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & & & \end{pmatrix} \end{array}.$$

For the rest of our discussion, we will restrict ourselves to submatrices of D. If we apply Algorithm 1 to D, we get the following restricted to appropriate submatrices:

$$\begin{array}{c|cccccc} & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{array} = \begin{array}{c|cccccc} & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 1 & 0 & 1 \end{array} \cdot \begin{array}{c|cccccc} & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline 5 & 1 & 0 & 0 & 1 & 1 & 0 \\ 6 & 0 & 1 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 1 & 1 & 0 & 1 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 1 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} .$$

Note that we have written entries corresponding to pivots bold. We may thus read off

$$\{[2,5),[3,6),[4,7)\} \subseteq \mathrm{Barc}(H_*(\mathbf{C}_\bullet)) \ .$$

Hence, for example, a generator of $[2,5)$ is given by $\mathrm{COL}_5(R)$, which has as a bounding chain $\mathrm{COL}_5(V)$.

2.4 PERSISTENT RELATIVE COHOMOLOGY

Definition 2.14 ([5])**.** A cochain complex $\mathbf{C}$ is a pair (C,δ), where $C = \bigoplus_{i\in\mathbb{Z}} C_i$ is a graded $\mathbb{F}$-vector space and $\delta = \bigoplus_{i\in\mathbb{Z}} \delta_i : C \to C$ is a coboundary operator, i.e., for each $i \in \mathbb{Z}$, $\delta_i : C_i \to C_{i+1}$ is a linear map such that $\delta^2 = \bigoplus_{i\in\mathbb{Z}} \delta_i \circ \delta_{i-1}$ is the zero map.

Let $\mathbf{C}$ be a finite-dimensional cochain complex. We refer to δ_i as the i-th coboundary map and to a vector τ in C_i as an i-cochain; i is then also referred to as the dimension of τ. Let n be the dimension of C as an $\mathbb{F}$-vector space. We denote a basis of C consisting of cochains by $\{\tau_1,\dots,\tau_n\}$. We will again refer to $\{\tau_1,\dots,\tau_n\}$ as a basis of $\mathbf{C}$, or equivalently $\mathbf{C} = \langle \tau_1,\dots,\tau_n\rangle$.

When discussing diagrams of cochain complexes, we again have to make sure that arrows are morphisms in an appropriate category of cochain complexes. For that we consider a category where objects are cochain complexes and morphisms are cochain maps; the latter being defined analogously to chain maps.

Let $\mathbf{C}_\bullet$ be a filtered chain complex. Consider the diagram of cochain complexes

$$\mathbf{C}^*_\bullet : \quad \mathbf{0} = \mathbf{C}^*_0 \twoheadleftarrow \dots \twoheadleftarrow \mathbf{C}^*_n = \mathbf{C}^* \ ,$$

where $\mathbf{C}^*_i = (\langle \sigma^*_1,\dots,\sigma^*_i\rangle,\delta)$; $\{\sigma^*_1,\dots,\sigma^*_n\}$ being the dual basis to some $\{\sigma_1,\dots,\sigma_n\}$. The coboundary operator of $\mathbf{C}^*$ is given by $\delta = \partial^*$, i.e., the adjoint of ∂. Again, $\mathbf{0}$ denotes the trivial cochain complex with $\{0\}$ as its vector space.

Definition 2.15. Let $\mathbf{C}$ be a cochain complex. Consider the vector spaces $Z^i = \ker(\delta_i)$, whose vectors we refer to as i-cocycles, and $B^i = \mathrm{im}(\delta_{i-1})$, whose vectors we refer to as i-coboundaries. We refer to the quotient $\ker(\delta)/\,\mathrm{im}(\delta)$ as the cohomology of $\mathbf{C}$.

Cohomology may again be interpreted as a functor from a cochain category to $\mathbf{Vect}$, as cochain maps induce linear maps in cohomology. Applying cohomology to above diagram yields

$$H^*(\mathbf{C}_\bullet) : \quad \{0\} = H^*(\mathbf{C}_0) \leftarrow \dots \leftarrow H^*(\mathbf{C}_n) = H^*(\mathbf{C}) \ ,$$

where we write $H^*(\mathbf{C}_i)$ for the cohomology of $\mathbf{C}^*_i$. As $H^*(\mathbf{C}_\bullet)$ defines a pointwise finite-dimensional persistence module, we refer to this as the persistent cohomology of the filtered chain complex $\mathbf{C}_\bullet$.

Remark 2.16. The barcodes $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$ and $\mathrm{Barc}(H^*(\mathbf{C}_\bullet))$ are in fact identical. This follows from the universal coefficient theorem, see [8], which yields a natural isomorphism between $H_*(\mathbf{C}_\bullet)$ and $H^*(\mathbf{C}_\bullet)$; the arrows in the respective diagrams thus being adjoint. Similarly, the barcodes for persistent relative homology and persistent relative cohomology as introduced below are identical as well.

Definition 2.17. Let $\mathbf{D}$ be a subcomplex of the chain complex $\mathbf{C}$. The quotient complex $\mathbf{C}/\mathbf{D}$ has as its vector space the quotient space C/D and as its boundary operator the induced boundary operator on that quotient.

Consider the following diagram of chain complexes with the boundary operators induced by ∂:

$$\mathbf{C}/\mathbf{C}_\bullet : \quad \mathbf{C} = \mathbf{C}/\mathbf{C}_0 \twoheadrightarrow \ldots \twoheadrightarrow \mathbf{C}/\mathbf{C}_n = \mathbf{0} \ .$$

Applying homology to above diagram yields the following pointwise finite-dimensional persistence module, which we call persistent relative homology:

$$H_*(\mathbf{C}/\mathbf{C}_\bullet) : \quad H_*(\mathbf{C}) = H_*(\mathbf{C}/\mathbf{C}_0) \to \ldots \to H_*(\mathbf{C}/\mathbf{C}_n) = \{0\} \ .$$

Hence, we may consider the barcode of $H_*(\mathbf{C}/\mathbf{C}_\bullet)$. Its intervals are $[i, j]$ for some $i, j \in \{0, \ldots, n-1\}$. We interpret them as half-open intervals of form $[i, j+1)$, if $0 < i$, and $[-\infty, j+1)$ else. We will refer to the set of these half-open intervals as $\mathrm{Barc}(H_*(\mathbf{C}/\mathbf{C}_\bullet))$ as well.

Remark 2.18. There is a bijection between $\mathrm{Barc}(H_*(\mathbf{C}_\bullet))$ and $\mathrm{Barc}(H_*(\mathbf{C}/\mathbf{C}_\bullet))$, which can be obtained by considering the concatenated sequence $H_*(\mathbf{C}_\bullet) \to H_*(\mathbf{C}/\mathbf{C}_\bullet)$. Finite intervals are identical, with a shift in dimension, however, which is made explicit below. With respect to infinite intervals, the bijection is given by $[e, \infty) \leftrightarrow [-\infty, e)$. An analogous statement can be made for persistent cohomology and persistent relative cohomology as defined further below. Furthermore, the basis from Theorem 2.10 also yields generators for persistent relative homology as is shown in Table 1.

generator	$\hat{\sigma}_e$	$\hat{\sigma}_b$	$\hat{\sigma}_d$
persistent homology	$[e, \infty)$	$[b, d)$	
persistent relative homology	$[-\infty, e)$		$[b, d)$

Table 1. The table illustrates the interplay between the basis from Theorem 2.10 and the intervals in the barcodes of persistent homology and persistent relative homology. Note that a finite interval has larger dimension in persistent relative homology, as its generator has larger dimension.

Analogously to persistent cohomology, we may consider the following diagram of cochain complexes with the coboundary operators induced by δ:

$$(\mathbf{C}/\mathbf{C}_\bullet)^* : \quad \mathbf{C}^* = (\mathbf{C}/\mathbf{C}_0)^* \hookleftarrow \ldots \hookleftarrow (\mathbf{C}/\mathbf{C}_n)^* = \mathbf{0} \ .$$

Applying cohomology to above diagram yields the following pointwise finite-dimensional persistence module, which we call persistent relative cohomology:

$$H^*(\mathbf{C}/\mathbf{C}_\bullet) : \quad H^*(\mathbf{C}) = H^*(\mathbf{C}/\mathbf{C}_0) \leftarrow \ldots \leftarrow H^*(\mathbf{C}/\mathbf{C}_n) = \{0\} \ .$$

There is a structural similarity between persistent homology and persistent relative cohomology, which becomes apparent when writing the reverse of the diagram of cochain complexes underlying persistent relative cohomology

$$\mathbf{C}_\bullet^\perp : \quad \mathbf{0} = \mathbf{C}_0^\perp \hookrightarrow \ldots \hookrightarrow \mathbf{C}_n^\perp = \mathbf{C}^\perp \ ,$$

where $\mathbf{C}_i^\perp = (\mathbf{C}/\mathbf{C}_{n-i})^*$. If we then denote $\tau_i = \sigma^*_{n+1-i}$, we get that

$$\mathbf{C}_i^\perp = (\mathbf{C}/\mathbf{C}_{n-i})^* = \langle \sigma_n^*, \ldots, \sigma^*_{n+1-i} \rangle = \langle \tau_1, \ldots, \tau_i \rangle \ .$$

We write $H^*(\mathbf{C}_\bullet^\perp)$ for cohomology applied to above diagram of cochain complexes, that is

$$H^*(\mathbf{C}_\bullet^\perp): \quad \{0\} = H^*(\mathbf{C}_0^\perp) \to \ldots \to H^*(\mathbf{C}_n^\perp) = H^*(\mathbf{C}^\perp) \ .$$

As it is the reverse of $H^*(\mathbf{C}/\mathbf{C}_\bullet)$, we may consider $\mathrm{Barc}(H^*(\mathbf{C}_\bullet^\perp))$. Its intervals are of form $[i,j]$ for some $i,j \in \{1,\ldots,n\}$. We may translate each interval $[i,j] \in \mathrm{Barc}(H^*(\mathbf{C}_\bullet^\perp))$ into an interval in $\mathrm{Barc}(H^*(\mathbf{C}/\mathbf{C}_\bullet))$ by considering $[n-j, n-i]$, which we can then translate into persistent cohomology; hence into persistent homology. Note that, in the following discussion, we will write an interval in persistent relative cohomology with the same convention as used for persistent homology.

Theorem 2.19. *Consider the diagram of cochain complexes*

$$\mathbf{C}_\bullet^\perp: \quad \mathbf{0} = \mathbf{C}_0^\perp \hookrightarrow \ldots \hookrightarrow \mathbf{C}_n^\perp = \mathbf{C}^\perp \ ,$$

where $\mathbf{C}_i^\perp = \langle \tau_1, \ldots, \tau_i \rangle$ for each $0 < i \leq n$. Then there exists a partition

$$\{1,\ldots,n\} = \mathcal{E} \cup \mathcal{B} \cup \mathcal{D}$$

such that the following holds:

1. *There exists a bijection $\mathcal{B} \leftrightarrow \mathcal{D}$ such that $b \leftrightarrow d$, if and only if $[b,d) \in \mathrm{Barc}(H^*(\mathbf{C}_\bullet^\perp))$.*
2. *There exists a basis $\{\hat\tau_1, \ldots, \hat\tau_n\}$ of $\mathbf{C}^\perp$ such that $\mathbf{C}_i^\perp = \langle \hat\tau_1, \ldots, \hat\tau_i \rangle$ for all $i \in \{1,\ldots,n\}$.*
3. *$\delta\hat\tau_e = 0$ for all $e \in \mathcal{E}$.*
4. *$\delta\hat\tau_d = \hat\tau_b$, and thus also $\delta\hat\tau_b = 0$, for all $[b,d) \in \mathrm{Barc}(H^*(\mathbf{C}_\bullet^\perp))$.*

Let $\mathbf{C}_\bullet$ be a filtered chain complex. Consider the diagram of cochain complexes that is $\mathbf{C}_\bullet^\perp$. Given the basis $\{\tau_1 \ldots, \tau_n\}$ of $\mathbf{C}^\perp$, we may represent δ by the strictly upper-triangular matrix $D^\perp$, which can also be implicitly defined via $\delta\tau_j = \sum_{i<j} D^\perp[i,j]\tau_i$. We refer to $D^\perp$ as the coboundary matrix of the filtered chain complex. Similar as before, both $\mathrm{Barc}(H^*(\mathbf{C}_\bullet^\perp))$ and the basis $\{\hat\tau_1, \ldots, \hat\tau_n\}$ from Theorem 2.19 can be obtained by computing a right-reduction of $D^\perp$.

Theorem 2.20. *Let $\mathbf{C}_\bullet$ be a filtered chain complex. Let $\{\hat\tau_1^*, \ldots, \hat\tau_n^*\}$ be the dual basis to $\{\hat\tau_1, \ldots, \hat\tau_n\}$ as given by Theorem 2.19. Denote $\hat\sigma_i = \hat\tau^*_{n+1-i}$. Then $\{\hat\sigma_1, \ldots, \hat\sigma_n\}$ is a basis of $\mathbf{C}$ as described in Theorem 2.10, up to non-zero scalar multiples.*

Example 2.21. *Recall Example 2.13. If we apply Algorithm 1 to $D^\perp$, yielding a right-reduction $R^\perp = D^\perp V^\perp$, we get the following restricted to appropriate submatrices:*

$$\begin{array}{c} \\ 10^* \\ 9^* \\ 8^* \\ 7^* \\ 6^* \\ 5^* \end{array}
\begin{array}{c} \begin{array}{cccc} 4^* & 3^* & 2^* & 1^* \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 0 & 0 \\ 0 & \mathbf{1} & 1 & 0 \\ 0 & 0 & \mathbf{1} & 0 \end{pmatrix} \end{array}
=
\begin{array}{c} \\ 10^* \\ 9^* \\ 8^* \\ 7^* \\ 6^* \\ 5^* \end{array}
\begin{array}{c} \begin{array}{cccc} 4^* & 3^* & 2^* & 1^* \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}
\cdot
\begin{array}{c} \\ 4^* \\ 3^* \\ 2^* \\ 1^* \end{array}
\begin{array}{c} \begin{array}{cccc} 4^* & 3^* & 2^* & 1^* \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} .$$

Note that, for row resp. column indices $n+1-i$ resp. $n+1-j$, we write i^* resp. j^*. Again, we have written entries corresponding to pivots bold. We may thus again read off

$$\{[2,5),[3,6),[4,7)\} \subseteq \mathrm{Barc}(H_*(\mathbf{C}_\bullet)) \ .$$

Furthermore, a generator of $[2,5)$ will be given by the dual of $\mathrm{COL}_{2^*}(V^\perp)$, which has as a bounding chain the dual of $\mathrm{COL}_{2^*}(R^\perp)$.

2.5 MOTIVATION

We are interested in computing a decomposition of a filtered chain complex $\mathbf{C}_\bullet$, i.e., a basis of $\mathbf{C}$ as described in Theorem 2.10. We have seen that, in theory, we may equally well compute a basis of $\mathbf{C}_\bullet^\perp$ as described in Theorem 2.19, from which we can deduce the basis we are interested in via Theorem 2.20.

As seen in [1] and [4], algorithms, such as Algorithm 1, perform better when applied to the coboundary matrix, especially when using optimization techniques such as clearing as discussed in Section 5. If the algorithm returns the barcode of persistent relative cohomology, we may easily translate it into persistent homology. Translating a basis as in Theorem 2.19 into a basis as in Theorem 2.10, however, requires computing the dual basis of the former. This involves inverting a possibly prohibitively large matrix.

Furthermore, as also seen in [1], an algorithm interested only in computing the barcode of persistent homology might not even return a basis as in Theorem 2.19 at all. For example, when computing a right-reduction $R^\perp = D^\perp V^\perp$, in order to increase performance, we might want to only save those columns of $V^\perp$ that are required to carry out lines 3-7 in Algorithm 1.

We therefore introduce a decomposition of the boundary matrix, from which we can read off a basis as in Theorem 2.10, but which can also be reconstructed from above mentioned columns of $V^\perp$.

3 U-MATCH DECOMPOSITIONS

Adapted from [7], we introduce the notion of a U-match decomposition of an arbitrary matrix. We highlight its connection to right-reductions and illustrate how we may extract from it a decomposition of a filtered chain complex, i.e., a basis as in Theorem 2.10, when the matrix is the boundary matrix of the filtered chain complex. We show that, for a given matrix D, one may uniquely reconstruct a (proper) U-match decomposition, if given data in form of a matching matrix M and a matrix $R^{-1}_{\boldsymbol{\rho\rho}}$. In the case of D being the boundary matrix of a filtered chain complex, M encodes finite intervals in the barcode of persistent homology and $R^{-1}_{\boldsymbol{\rho\rho}}$ encodes those columns of $V^{\perp}$ required to carry out lines 3-7 in Algorithm 1 when applying it to the coboundary matrix $D^{\perp}$ of the filtered chain complex.

3.1 INTRODUCING U-MATCH DECOMPOSITIONS

In this section, unless otherwise stated, we consider an arbitrary matrix $D \in \mathbb{F}^{m\times n}$.

Definition 3.1. A U-match decomposition of D is a tuple (R, M, D, C) such that $RM = DC$, where M is a matching matrix as defined below, and R and C are upper unitriangular matrices. Note that $R \in \mathbb{F}^{m\times m}$, $C \in \mathbb{F}^{n\times n}$ and $M \in \mathbb{F}^{m\times n}$.

Definition 3.2. A matching matrix M has in any row or column at most one non-zero entry.

Definition 3.3. Let $RM = DC$ be a U-match decomposition. We define a relation

$$\mu \subseteq \{1, \dots, m\} \times \{1, \dots, n\}$$

such that $(r, c) \in \mu$, if and only if $M[r, c] \neq 0$. Consider the sets $\mathrm{def}(\mu) = \{r : (r, c) \in \mu\}$ and $\mathrm{val}(\mu) = \{c : (r, c) \in \mu\}$, which are in bijection because M is a matching matrix. Therefore, we may, for any $(r, c) \in \mu$, define $\mathrm{row}(c) = r$ and $\mathrm{col}(r) = c$.

Lemma 3.4. *Let $RM = DC$ be a U-match decomposition. For each $c \in \mathrm{val}(\mu)$, we have*

$$D \cdot \mathrm{COL}_c(C) = \mathrm{COL}_{\mathrm{row}(c)}(R) \cdot M[\mathrm{row}(c), c] \quad .$$

Else, if $c \notin \mathrm{val}(\mu)$, we have $D \cdot \mathrm{COL}_c(C) = 0$.

Proof. Consider that

$$\begin{aligned} D \cdot \mathrm{COL}_c(C) &= \mathrm{COL}_c(DC) \\ &= \mathrm{COL}_c(RM) \\ &= \begin{cases} \mathrm{COL}_{\mathrm{row}(c)}(R) \cdot M[\mathrm{row}(c), c] & c \in \mathrm{val}(\mu), \\ 0 & \text{else.} \end{cases} \end{aligned}$$

□

Thus, we may indeed interpret M as matching columns of C, after left-multiplying with D, with columns of R. An analogous statement can be made for rows of C and R. These matchings are indeed unique and thus a property inherent to D as we will now prove.

Proposition 3.5. *Let $RM = DC$ and $\tilde{R}\tilde{M} = D\tilde{C}$ be two U-match decompositions of D. Then $M = \tilde{M}$.*

Proof. Rearranging both U-match decompositions and equating them yields a U-match decomposition of $\tilde{M}$, that is

$$\tilde{R}^{-1}RM = \tilde{M}\tilde{C}^{-1}C \quad .$$

Consider any $i \times j$ submatrix in the lower left-hand corners of M and $\tilde{M}$. Matrix multiplication by upper unitriangular matrices, i.e., invertible upper triangular matrices, does not change their ranks. Therefore, each pair of $i \times j$ submatrices has the same amount of non-zero entries and hence M and $\tilde{M}$ have to have the same sparsity pattern. Lastly, non-zero entries of M and resp. $\tilde{M}$ are not changed when multiplying by upper unitriangular matrices. Thus, the entries of M and $\tilde{M}$ are the same. □

While M is thus uniquely determined by D, this is neither the case for R nor C. To see this, consider U-match decompositions of the unit matrix. Given certain requirements upon R and C, however, they each determine the other.

Definition 3.6. A U-match decomposition $RM = DC$ is proper, if $\mathrm{COL}_k(M) = 0$ implies that $\mathrm{ROW}_k(C) = e_k$, and, if $\mathrm{ROW}_k(M) = 0$ implies that $\mathrm{COL}_k(R) = e_k$.

The algorithms described in Section 4 compute proper U-match decompositions. This will then also answer the question of existence for (proper) U-match decompositions.

For sake of simplicity with respect to statements using matrices in block structure and, later on, statements regarding the reconstruction of a decomposition from certain data, we now introduce some further notation. We arrange $\mathrm{def}(\mu)$ and resp. $\{1, \ldots, m\} \setminus \mathrm{def}(\mu)$ into increasing sequences

$$\boldsymbol{\rho} = (\rho_1 < \ldots < \rho_k) \quad \text{and resp.} \quad \bar{\boldsymbol{\rho}} = (\bar{\rho}_1 < \ldots < \bar{\rho}_{m-k}) \quad .$$

For $\mathrm{val}(\mu)$ and resp. $\{1, \ldots, n\} \setminus \mathrm{val}(\mu)$, we similarly define

$$\boldsymbol{\kappa} = (\kappa_1 < \ldots < \kappa_k) \quad \text{and resp.} \quad \bar{\boldsymbol{\kappa}} = (\bar{\kappa}_1 < \ldots < \bar{\kappa}_{n-k}) \quad .$$

Proposition 3.7. *If $RM = DC$ is a proper U-match decomposition, then*

$$R^{-1} \equiv \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array} \begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I & -D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}}(D_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1} \\ 0 & * \end{pmatrix} \end{array} \qquad \textit{and} \qquad C \equiv \begin{array}{c} \\ \boldsymbol{\kappa} \\ \bar{\boldsymbol{\kappa}} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \end{array} \\ \begin{pmatrix} * & -(D_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} \\ 0 & I \end{pmatrix} \end{array} .$$

Proof. The claim that $R^{-1}_{\bar{\boldsymbol{\rho}}\bar{\boldsymbol{\rho}}}$ and $C_{\bar{\boldsymbol{\kappa}}\bar{\boldsymbol{\kappa}}}$ are unit matrices follows from the properness of the U-match decomposition. Same holds true for the claim that $R^{-1}_{\boldsymbol{\rho}\bar{\boldsymbol{\rho}}}$ and $C_{\bar{\boldsymbol{\kappa}}\boldsymbol{\kappa}}$ are zero matrices. The claims regarding $R^{-1}_{\bar{\boldsymbol{\rho}}\boldsymbol{\rho}}$ and $C_{\boldsymbol{\kappa}\bar{\boldsymbol{\kappa}}}$ follow from Theorem 3.16. Note that we write, e.g., $R^{-1}_{\bar{\boldsymbol{\rho}}\bar{\boldsymbol{\rho}}}$ for $(R^{-1})_{\bar{\boldsymbol{\rho}}\bar{\boldsymbol{\rho}}}$. □

Note that for a (proper) U-match decomposition $RM = DC$ of D, we get another (proper) U-match decomposition $(C^{-1})^{\perp}M^{\perp} = D^{\perp}(R^{-1})^{\perp}$ of $D^{\perp}$. We refer to the latter U-match decomposition as the antitranspose of the former.

Proposition 3.8. *If $RM = DC$ and $\tilde{R}M = D\tilde{C}$ are two proper U-match decompositions of D, then $R = \tilde{R}$, if and only if $C = \tilde{C}$*

Proof. R determines C uniquely by Theorem 3.16. C determines R uniquely by considering the antitransposes of the given U-match decompositions and then also by Theorem 3.16. □

Given a matrix D, Proposition 3.7 and Proposition 3.8 imply that data in form of its matching matrix M and some $R^{-1}_{\rho\rho}$ suffice to uniquely determine a proper U-match decomposition. That is under the reservation that $R^{-1}_{\rho\rho}$ comes from a proper U-match decomposition to begin with.

There is a close connection between right-reductions and U-match decompositions of D. For sake of simplicity, we assume, given a right-reduction $\tilde{R} = DV$, that V is upper unitriangular.

Theorem 3.9. *To any right-reduction $\tilde{R} = DV$ corresponds some U-match decomposition $RM = DC$, and the other way around as well.*

Proof. Let $RM = DC$ be a U-match decomposition. Then RM is reduced and thus $(RM) = DC$ is a right-reduction of D. Let $\tilde{R} = DV$ be a right-reduction. We may transform $\tilde{R}$ into a matching matrix by adding rows with higher indices to rows with lower indices. This can be encoded by an upper unitriangular matrix R^{-1}. This yields $M = R^{-1}DV$. We get the desired U-match decomposition by left-multiplying by R. □

We call a right-reduction $\tilde{R} = DV$ proper, if V obeys the same requirement as the matrix C does in Definition 3.6, that is with respect to M as computed in Theorem 3.9.

Proposition 3.10. *To any proper right-reduction $\tilde{R} = DV$ corresponds a unique proper U-match decomposition $RM = DC$, and vice versa.*

Proof. Existence is answered in Theorem 3.9. By Proposition 3.8, R is uniquely determined by V, and the other way around as well. Hence, the correspondence is unique. □

Remark 3.11. Note that Algorithm 1 computes a proper right-reduction of D and thus, using Theorem 3.9 and Proposition 3.10, also yields an approach to computing a proper U-match decomposition. Consider a U-match decomposition $RM = DC$ of a boundary matrix. In light of Lemma 3.4, we may compute a decomposition of the filtered chain complex, i.e., a basis in the sense of Theorem 2.10, by considering suitable columns of C and, if appropriate, their boundaries, i.e., suitable columns of R. Furthermore, we might also apply Algorithm 1 to the coboundary matrix, compute a U-match decomposition via Theorem 3.9 and then consider its antitranspose. This then also yields a U-match decomposition of the boundary matrix; hence also a decomposition of the filtered chain complex.

Example 3.12. *Recall Example 2.13. If we compute, in accordance with the proof of Theorem 3.9, a U-match decomposition from the right-reduction shown there, we get, restricted to appropriate submatrices, that*

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} .$$

Note that we have written the non-zero entries of M bold. Furthermore, notice that the given U-match decomposition is proper. We find that $\boldsymbol{\rho} = (\ldots, 2, 3, 4, \ldots)$ as well as $\boldsymbol{\kappa} = (\ldots, 5, 6, 7, \ldots)$.

3.2 INNER IDENTITIES

In the following paragraphs, let $RM = DC$ be a proper U-match decomposition of D. We may write it in block structure as follows:

$$\begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array} \begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I & R_{\bar{\boldsymbol{\rho}}\boldsymbol{\rho}} \\ 0 & R_{\boldsymbol{\rho}\boldsymbol{\rho}} \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \end{array} \\ \begin{pmatrix} 0 & 0 \\ M_{\boldsymbol{\rho}\boldsymbol{\kappa}} & 0 \end{pmatrix} \end{array} = \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \end{array} \\ \begin{pmatrix} D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}} & D_{\bar{\boldsymbol{\rho}}\bar{\boldsymbol{\kappa}}} \\ D_{\boldsymbol{\rho}\boldsymbol{\kappa}} & D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ \boldsymbol{\kappa} \\ \bar{\boldsymbol{\kappa}} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \end{array} \\ \begin{pmatrix} C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} & C_{\boldsymbol{\kappa}\bar{\boldsymbol{\kappa}}} \\ 0 & I \end{pmatrix} \end{array} .$$

Lemma 3.13. *In a proper U-match decomposition, $R_{\boldsymbol{\rho}\boldsymbol{\rho}}$ and $C_{\boldsymbol{\kappa}\boldsymbol{\kappa}}$ are upper unitriangular.*

Proof. $R_{\boldsymbol{\rho}\boldsymbol{\rho}}$ and $C_{\boldsymbol{\kappa}\boldsymbol{\kappa}}$ are upper unitriangular, as $\boldsymbol{\rho}$ and $\boldsymbol{\kappa}$ are increasing sequences and R and C are upper unitriangular to begin with. □

Lemma 3.14. *$M_{\boldsymbol{\rho}\boldsymbol{\kappa}}$ and $D_{\boldsymbol{\rho}\boldsymbol{\kappa}}$ are invertible.*

Proof. $M_{\boldsymbol{\rho}\boldsymbol{\kappa}}$ is invertible by definition of $\boldsymbol{\rho}$ and $\boldsymbol{\kappa}$ and as M is a matching matrix. By the block structure, we have that

$$D_{\boldsymbol{\rho}\boldsymbol{\kappa}} = R_{\boldsymbol{\rho}\boldsymbol{\rho}} M_{\boldsymbol{\rho}\boldsymbol{\kappa}} C_{\boldsymbol{\kappa}\boldsymbol{\kappa}}^{-1} .$$

The claim follows by Lemma 3.13. □

Recall that $\boldsymbol{\kappa} = (\kappa_1 < \ldots < \kappa_k)$. Now define $\boldsymbol{\kappa}^* = (\mathrm{row}(\kappa_1), \ldots, \mathrm{row}(\kappa_k))$. Note that $\boldsymbol{\kappa}^*$ is not increasing in general.

Lemma 3.15. *$(R^{-1}D)_{\boldsymbol{\kappa}^*\boldsymbol{\kappa}}$ is invertible and upper triangular.*

Proof. We want to show that the matrix in consideration is invertible and has the following form:

$$(R^{-1}D)_{\boldsymbol{\kappa}^*\boldsymbol{\kappa}} = \begin{array}{c} \\ \kappa_1^* \\ \vdots \\ \kappa_k^* \end{array} \begin{array}{c} \begin{array}{ccc} \kappa_1 & \ldots & \kappa_k \end{array} \\ \begin{pmatrix} * & \ldots & * \\ & \ddots & \vdots \\ & & * \end{pmatrix} \end{array} .$$

We rearrange the U-match decomposition as $R^{-1}D = MC^{-1}$. Consider $\mathrm{COL}_{\kappa_p}(C^{-1})$, which has no non-zero entries below index κ_p, as C^{-1} is upper unitriangular. Hence, $\mathrm{COL}_{\kappa_p}(MC^{-1})$ may only have

non-zero entries in indices $\mathrm{row}(\kappa_q)$ with $q \leq p$, as M is a matching matrix. Thus, above matrix is upper triangular. To establish invertibility, consider that

$$(R^{-1}D)_{\boldsymbol{\kappa}^*\boldsymbol{\kappa}} \equiv (R^{-1}D)_{\boldsymbol{\rho}\boldsymbol{\kappa}} \ .$$

Furthermore, notice that

$$(R^{-1}D)_{\boldsymbol{\rho}\boldsymbol{\kappa}} C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}} D_{\boldsymbol{\rho}\boldsymbol{\kappa}} C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} = M_{\boldsymbol{\rho}\boldsymbol{\kappa}} \ .$$

The claim follows by Lemma 3.13 and Lemma 3.14. □

Theorem 3.16. *Consider the matrix $A = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}} D_{\boldsymbol{\rho}\boldsymbol{\kappa}} = (R^{-1}D)_{\boldsymbol{\rho}\boldsymbol{\kappa}}$. Then the following holds:*

$$C \equiv \begin{array}{c|cc} & \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \\ \hline \boldsymbol{\kappa} & A^{-1}M_{\boldsymbol{\rho}\boldsymbol{\kappa}} & -A^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} \\ \bar{\boldsymbol{\kappa}} & 0 & I \end{array} ,$$

$$C^{-1} \equiv \begin{array}{c|c} & \mathbf{n} \\ \hline \boldsymbol{\kappa} & (M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\mathbf{n}} \\ \bar{\boldsymbol{\kappa}} & I_{\bar{\boldsymbol{\kappa}}\mathbf{n}} \end{array} ,$$

$$R^{-1} \equiv \begin{array}{c|cc} & \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \\ \hline \bar{\boldsymbol{\rho}} & I & -D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}}(D_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1} \\ \boldsymbol{\rho} & 0 & R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}} \end{array} ,$$

$$R \equiv \begin{array}{c|cc} & \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \\ \hline \mathbf{m} & I_{\mathbf{m}\bar{\boldsymbol{\rho}}} & D_{\mathbf{m}\boldsymbol{\kappa}}A^{-1} \end{array} .$$

Proof. Within this proof, we will refer to the equivalences we are proving as the first through fourth inner identities. Recall that we may write the U-match decomposition in block structure, namely that

$$\begin{array}{c|cc} & \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \\ \hline \bar{\boldsymbol{\rho}} & I & R_{\bar{\boldsymbol{\rho}}\boldsymbol{\rho}} \\ \boldsymbol{\rho} & 0 & R_{\boldsymbol{\rho}\boldsymbol{\rho}} \end{array} \cdot \begin{array}{c|cc} & \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \\ \hline \bar{\boldsymbol{\rho}} & 0 & 0 \\ \boldsymbol{\rho} & M_{\boldsymbol{\rho}\boldsymbol{\kappa}} & 0 \end{array} = \begin{array}{c|cc} & \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \\ \hline \bar{\boldsymbol{\rho}} & D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}} & D_{\bar{\boldsymbol{\rho}}\bar{\boldsymbol{\kappa}}} \\ \boldsymbol{\rho} & D_{\boldsymbol{\rho}\boldsymbol{\kappa}} & D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} \end{array} \cdot \begin{array}{c|cc} & \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \\ \hline \boldsymbol{\kappa} & C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} & C_{\boldsymbol{\kappa}\bar{\boldsymbol{\kappa}}} \\ \bar{\boldsymbol{\kappa}} & 0 & I \end{array} .$$

As an intermediate claim, where the equalities follow from the block structure, we prove that

$$MC^{-1} = R^{-1}D \equiv \begin{array}{c|c} & \mathbf{n} \\ \hline \bar{\boldsymbol{\rho}} & 0 \\ \boldsymbol{\rho} & R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\mathbf{n}} \end{array} \quad \text{and} \quad RM = DC \equiv \begin{array}{c|cc} & \boldsymbol{\kappa} & \bar{\boldsymbol{\kappa}} \\ \hline \mathbf{m} & D_{\mathbf{m}\boldsymbol{\kappa}}C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} & 0 \end{array} .$$

From the block structure of the decomposition, we get that

$$R^{-1}D \;\equiv\; \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array}\!\!\begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I & R^{-1}_{\bar{\boldsymbol{\rho}}\boldsymbol{\rho}} \\ 0 & R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}} \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array}\!\!\begin{array}{c} \mathbf{n} \\ \begin{pmatrix} D_{\bar{\boldsymbol{\rho}}\mathbf{n}} \\ D_{\boldsymbol{\rho}\mathbf{n}} \end{pmatrix} \end{array} .$$

We have that $(R^{-1}D)_{\boldsymbol{\rho}\mathbf{n}} = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\mathbf{n}}$. Furthermore, each row of M with index in $\bar{\boldsymbol{\rho}}$ is zero. Thus, $(MC^{-1})_{\bar{\boldsymbol{\rho}}\mathbf{n}}$ is zero. This proves the first intermediate claim. Analogously, by considering the antitranspose of the U-match decomposition, we may prove the second intermediate claim. By properness and the two intermediate claims, it follows that

$$C^{-1} \;\equiv\; \begin{array}{c} \\ \boldsymbol{\kappa} \\ \bar{\boldsymbol{\kappa}} \end{array}\!\!\begin{array}{c} \mathbf{n} \\ \begin{pmatrix} (M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\mathbf{n}} \\ I_{\bar{\boldsymbol{\kappa}}\mathbf{n}} \end{pmatrix} \end{array} \qquad \text{and} \qquad R \;\equiv\; \begin{array}{c} \\ \mathbf{m} \end{array}\!\!\begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I_{\mathbf{m}\bar{\boldsymbol{\rho}}} & D_{\mathbf{m}\boldsymbol{\kappa}}C_{\boldsymbol{\kappa}\boldsymbol{\kappa}}(M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1} \end{pmatrix} \end{array} .$$

We have thus shown the second inner identity. Recall that

$$A = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\boldsymbol{\kappa}} \;,$$

and hence

$$(D_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1} = A^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}} \;.$$

From the block structure, we get that

$$R_{\boldsymbol{\rho}\boldsymbol{\rho}} = (M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}D_{\boldsymbol{\rho}\boldsymbol{\kappa}}C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} \;,$$

and thus

$$C_{\boldsymbol{\kappa}\boldsymbol{\kappa}} = A^{-1}M_{\boldsymbol{\rho}\boldsymbol{\kappa}} \;.$$

Therefore, we get the fourth inner identity, namely that

$$R \;\equiv\; \begin{array}{c} \\ \mathbf{m} \end{array}\!\!\begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I_{\mathbf{m}\bar{\boldsymbol{\rho}}} & D_{\mathbf{m}\boldsymbol{\kappa}}A^{-1} \end{pmatrix} \end{array} .$$

Now, write R as

$$R \;\equiv\; \begin{array}{c} \\ \bar{\boldsymbol{\rho}} \\ \boldsymbol{\rho} \end{array}\!\!\begin{array}{c} \begin{array}{cc} \bar{\boldsymbol{\rho}} & \boldsymbol{\rho} \end{array} \\ \begin{pmatrix} I & D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}}A^{-1} \\ 0 & R_{\boldsymbol{\rho}\boldsymbol{\rho}} \end{pmatrix} \end{array} .$$

Above matrix and the one shown in the third inner identity are inverse to each other. Thus, we have proven the third inner identity, as inverses are unique. Now, assume that the first inner identity is true. Then it holds that

$$(CC^{-1})_{\boldsymbol{\kappa}\boldsymbol{\kappa}} = A^{-1}M_{\boldsymbol{\rho}\boldsymbol{\kappa}}(M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\boldsymbol{\kappa}} = I \;.$$

Furthermore, we have that

$$(CC^{-1})_{\boldsymbol{\kappa}\bar{\boldsymbol{\kappa}}} = A^{-1}M_{\boldsymbol{\rho}\boldsymbol{\kappa}}(M_{\boldsymbol{\rho}\boldsymbol{\kappa}})^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} - A^{-1}R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}} = 0 \;.$$

Hence, the matrix shown in the first inner identity is indeed the inverse to that shown in the second inner identity. Thus, by uniqueness of inverses, we have proven the first inner identity. □

Remark 3.17. Given a matrix D, we have previously seen that data in form of its matching matrix M and some $R^{-1}_{\boldsymbol{\rho\rho}}$ belonging to a proper U-match decomposition suffice to uniquely determine it. Theorem 3.16 tells us the necessary computations to explicitly reconstruct this decomposition.

3.3 RECONSTRUCTING U-MATCH DECOMPOSITIONS

Based on Theorem 3.16, we now make explicit the computations necessary to reconstruct a proper U-match decomposition given the mentioned data, that is D, M and $R^{-1}_{\boldsymbol{\rho\rho}}$. We will see that reconstructing an arbitrary row or column of one of the matrices R and C or their inverses requires at most one back-substitution. Note that, given an invertible upper triangular matrix T, we may evaluate $T^{-1}b$ resp. cT^{-1} by solving $Tx = b$ resp. $yT = c$ via back-substitution. Setting b resp. c to unit vectors allows us to solve for columns resp. rows of T^{-1}. Furthermore, note that any row or column of $A = R^{-1}_{\boldsymbol{\rho\rho}} D_{\boldsymbol{\rho\kappa}}$ may simply be computed directly from the mentioned data. Hence, using Lemma 3.15, we may solve $Ax = b$ and $yA = c$ via back-substitution as well. Lastly, as $D_{\boldsymbol{\rho\kappa}} = R_{\boldsymbol{\rho\rho}} A$, we can also solve $D_{\boldsymbol{\rho\kappa}} x = b$ and $yD_{\boldsymbol{\rho\kappa}} = c$ via back-substitution and some matrix-vector multiplication. Note that we may easily access $(M_{\boldsymbol{\rho\kappa}})^{-1}$ by considering its transpose and replacing entries by their multiplicative inverses.

Theorem 3.18. *Let $RM = DC$ be a proper U-match decomposition. Assume that we are given data in form of D, M and $R^{-1}_{\boldsymbol{\rho\rho}}$. We obtain any row or column of R and C and any column of R^{-1} via back-substitution. Furthermore, we obtain rows and columns of C^{-1} and rows of $R^{-1}_{\boldsymbol{\rho m}}$ without back-substitution.*

Proof. See above paragraph as well as Table 2 for the necessary computations. Do note however that, if T is an upper triangular matrix and S a matrix of equal dimensions and we evaluate $\mathrm{COL}_i(T^{-1}S)$, we solve $Tx = \mathrm{COL}_i(S)$. If we wish to evaluate $\mathrm{COL}_i(ST^{-1})$ however, we first solve $Tx = e_i$ and left-multiply the solution with S. An analogous statement holds when solving for rows of $T^{-1}S$. This explains why the linear equations for the κ_i-th row of C and the ρ_i-th column of R in Table 2 might look odd at first glance. □

Remark 3.19. We have previously seen that we may obtain a decomposition of a filtered chain complex by applying Algorithm 1 to its coboundary matrix, computing the corresponding U-match decomposition via Theorem 3.9 and then considering its antitranspose. We now have an alternative approach. Theorem 3.18 tells us that when data in form of D, M and $R^{-1}_{\boldsymbol{\rho\rho}}$ is given, we are able to compute a decomposition of the filtered chain complex. Note that M encodes finite intervals in the barcode of persistent homology and $R^{-1}_{\boldsymbol{\rho\rho}}$ encodes those columns of $V^{\perp}$ required to carry out lines 3-7 of Algorithm 1 when applying it to the coboundary matrix $D^{\perp}$. An illustration of this can be seen in Section 4.

Row	ρ_i / κ_i	$\bar{\rho}_i / \bar{\kappa}_i$
R^{-1}	(1)	$xD_{\boldsymbol{\rho\kappa}} = -\operatorname{ROW}_i(D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}})$
R	$xR^{-1}_{\boldsymbol{\rho\rho}} = e_i$	$xA = \operatorname{ROW}_i(D_{\bar{\boldsymbol{\rho}}\boldsymbol{\kappa}})$
C^{-1}	(2)	(3)
C	$xA = e_i$	(3)

Column	ρ_i / κ_i	$\bar{\rho}_i / \bar{\kappa}_i$
R^{-1}	$D_{\boldsymbol{\rho\kappa}}x = e_i$	(3)
R	$Ax = e_i$	(3)
C^{-1}	(2)	(2)
C	$Ax = \operatorname{COL}_i(M_{\boldsymbol{\rho\kappa}})$	$Ax = -R^{-1}_{\boldsymbol{\rho\rho}} \operatorname{COL}_i(D_{\boldsymbol{\rho}\bar{\boldsymbol{\kappa}}})$

Table 2. The table shows the computations necessary in order to obtain any row or column of the matrices R and C or their inverses. If a row or column requires the solution of a linear equation via back-substitution, this is indicated by either $Tx = b$ or $cT = y$. If the row or column can be readily read off the available data, this is indicated by (1). If the row or column can be computed via matrix-vector-multiplication, we write (2). If the row or column in question is a unit vector, we denote this by (3).

4 DECOMPOSITION ALGORITHMS

Adapted from [7], we present two algorithms for computing a proper U-match decomposition of an arbitrary matrix D, i.e., $RM = DC$. Note that, in light of Theorem 3.9, we have already seen an algorithm that allows us to compute a U-match decomposition, namely Algorithm 1. Algorithm 2 computes the matrices M and the inverses to R and C. In that sense, it is Algorithm 1 applied to $D^{\perp}$ and an implementation of what is described in Theorem 3.9. Algorithm 3 computes M and $R^{-1}_{\boldsymbol{\rho\rho}}$ only. In that sense, it is a compressed version of Algorithm 1 applied to $D^{\perp}$ or equivalently of Algorithm 2 applied to D.

We will formulate Algorithm 2 and Algorithm 3 as row algorithms as opposed to the column algorithm that is Algorithm 1. Hence, the former being applied to D are analogous to the latter being applied to $D^{\perp}$. We have already introduced the notion of a reduced matrix in Definition 2.12, which can be interpreted as a notion of column-reducedness. We will now introduce a similar but different notion of a reduced matrix, which can be interpreted as a notion of row-reducedness.

Definition 4.1. Let $\bar{D}$ be a matrix. Define $\mathrm{pivot}(\mathrm{ROW}_i(\bar{D}))$ as the smallest column index with non-zero entry in $\mathrm{ROW}_i(\bar{D})$, if well defined. We refer to it as the pivot of $\mathrm{ROW}_i(\bar{D})$. $\bar{D}$ is row-reduced, if there do not exist distinct row indices i and j such that $\mathrm{pivot}(\mathrm{row}_i(\bar{D})) = \mathrm{pivot}(\mathrm{row}_j(\bar{D}))$.

Algorithm 2 U-match decomposition

Require: $D \in \mathbb{F}^{m\times n}$
Ensure: $R^{-1} \in \mathbb{F}^{m\times m}, C^{-1} \in \mathbb{F}^{n\times n}$ upper unitriangular and $M \in \mathbb{F}^{m\times n}$ a matching matrix such that $RM = DC$ is a U-match decomposition

```
D̄ := D, R^{-1} := I_{m×m}, C^{-1} := I_{n×n}, M := 0_{m×n}
for i ∈ {1, ..., m} in decreasing order do
    while there exist i < j and k such that k = pivot(ROW_i(D̄)) = pivot(ROW_j(D̄)) do
        λ := D̄[i, k] / D̄[j, k]
        ROW_i(D̄) := ROW_i(D̄) − λ · ROW_j(D̄)
        ROW_i(R^{-1}) := ROW_i(R^{-1}) − λ · ROW_j(R^{-1})
    end while
end for
for i ∈ {1, ..., m} in increasing order do
    if there exists k such that k = pivot(ROW_i(D̄)) then
        ROW_k(C^{-1}) := 1/D̄[i, k] · ROW_i(D̄)
        M[i, k] := D̄[i, k]
    end if
end for
```

Proposition 4.2. *Algorithm 2 returns a proper U-match decomposition.*

Proof. The first for loop in Algorithm 2 adds to any $\mathrm{ROW}_i(\bar{D})$ multiplies of $\mathrm{ROW}_j(\bar{D})$ with $i < j$. R^{-1} encodes these row operations and is an upper unitriangular matrix. Furthermore, $R^{-1}D$ will be row-reduced as soon as the for loop terminates. The second for loop rescales non-zero rows of $R^{-1}D$ and assembles them into an upper unitriangular matrix C^{-1}. M is a matching matrix, as $R^{-1}D$ is row-

reduced. In order to see that $R^{-1}D = MC^{-1}$, note that, if $\mathrm{ROW}_i(R^{-1}D) = 0$, then $\mathrm{ROW}_i(M) = 0$, and, if $\mathrm{ROW}_i(M) \neq 0$, then

$$\begin{aligned}\mathrm{ROW}_i(MC^{-1}) &= \mathrm{ROW}_i(M)C^{-1} \\ &= M[i,k] \cdot \mathrm{ROW}_k(C^{-1}) \\ &= D[i,k] \cdot \mathrm{ROW}_k(C^{-1}) \\ &= \mathrm{ROW}_i(R^{-1}D) \quad .\end{aligned}$$

In order to see the properness of the output, recall that Algorithm 1 returns a proper U-match decomposition. The first for loop of Algorithm 2 is analogous to that of the former and thus $(R^{-1})^{\perp}$ belongs to a proper U-match decomposition of $D^{\perp}$; hence R belongs to a proper U-match decomposition of D. C is then uniquely determined by R and thus the entire decomposition is proper. □

Remark 4.3. Algorithms, that are adaptations of Algorithm 1 applied to $D^{\perp}$ and thus share great similarity with Algorithm 2, can also be found in [1] and [4].

Example 4.4. *Recall Example 2.13. If we apply Algorithm 2 to the boundary matrix D shown there, we get, when restricted to appropriate submatrices, that*

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

$$= \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} .$$

Note that above matrix equation is taken from $R^{-1}D = MC^{-1}$ in order to highlight that Algorithm 2 is an adapted version of Algorithm 1 applied to $D^{\perp}$ as can be seen by comparing the above to Example 2.21. Non-zero entries of M are written bold as usual. We again find that $\boldsymbol{\rho} = (\ldots, 2, 3, 4, \ldots)$ as well as $\boldsymbol{\kappa} = (\ldots, 5, 6, 7, \ldots)$.

Algorithm 3 Compressed U-match decomposition

Require: $D \in \mathbb{F}^{m \times n}$
Ensure: $M \in \mathbb{F}^{m \times n}$ a matching matrix and $\bar{R}$ upper triangular such that $\bar{R} = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}$ with R^{-1} being put out by Algorithm 2

```
R̄ := ∅, M := 0_{m×n}, indices := ∅
for i ∈ {1, ..., m} in decreasing order do
    vector := 0_{1×length(indices)}
    row := ROW_i(D)
    while there exist j, k, l such that M[j, k] ≠ 0 and k = pivot(row) and j = indices[l] do
        λ := row[k] / M[j, k]
        reduced := ROW_l(R̄) · D_{indices,n}
        row := row − λ · reduced
        vector := vector − λ · ROW_l(R̄)
    end while
    if row ≠ 0 then
        k := pivot(row)
        R̄ := (1  vector; 0  R̄)
        M[i, k] := D[i, k]
        indices.push(i)
        indices.sort()
    end if
end for
```

Proposition 4.5. *Let R^{-1} and $\bar{R}$ be the matrices put out by Algorithm 2 and Algorithm 3, then $\bar{R} = R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}$.*

Proof. The claim follows from the fact that Algorithm 3 is a compressed version of Algorithm 2 in which we record less data, e.g., only rows and columns of R^{-1} that correspond to non-zero rows in $R^{-1}D$. □

Remark 4.6. An algorithm that shares great similarity with Algorithm 3 can be found in [1].

Example 4.7. *Recall Example 4.4 and that we had already found out that $\boldsymbol{\rho} = (\ldots, 2, 3, 4, \ldots)$ as well as $\boldsymbol{\kappa} = (\ldots, 5, 6, 7, \ldots)$. When restricted to an appropriate submatrix, $R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}}$ is given by*

$$\begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Hence, restricted to an appropriate submatrix, A is given by

$$\begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Therefore, solving the following system of linear equations yields some of the columns of C we are interested in when decomposing the filtered chain complex:

$$\begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array} \cdot \begin{array}{c} \\ 5 \\ 6 \\ 7 \end{array}\begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \end{array} = \begin{array}{c} \\ 5 \\ 6 \\ 7 \end{array}\begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} .$$

The solution to above system of equations is given by

$$\begin{array}{c} \\ 5 \\ 6 \\ 7 \end{array}\begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \end{array} .$$

The columns of this matrix correspond to bounding chains of the intervals $[2,5)$, $[3,6)$ and $[4,7)$ in persistent homology. Their boundaries, i.e., generators of aforementioned intervals, are given by the columns of the matrix

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array} .$$

5 OPTIMIZATIONS

Adapted from [1], [4] and [7], we briefly discuss some optimizations which can be applied to Algorithm 3 or have been made already.

Optimizations implicitly made are, on one hand, the computation in cohomology in the sense that we consider a row algorithm applied to D, which is equivalent to a column algorithm applied to $D^{\perp}$. On the other hand, we use a form of implicit matrix reduction in the sense that we only save data relevant for reduction, that is lines 5-10 in Algorithm 3, and compute rows of $R^{-1}D$, which are analogous to coboundaries, whenever needed on-the-fly, that is line 7 in Algorithm 3.

An optimization which allows us to skip some iterations of the for loop in Algorithm 3 is so-called clearing, and based on Proposition 5.1. Especially when combined with computation in cohomology, this can lead to a substantial increase in performance in the case of filtered chain complexes arising from Vietoris-Rips complexes, see [1]. Clearing allows us to skip an iteration of the for loop in Algorithm 3 whenever we know that the current row index is featured in $\mathrm{val}(\mu)$. This requires however a computation in chunks, i.e., an application of Algorithm 3 to submatrices of D in a reasonable order as, for all $(r, c) \in \mu, r < c$. If the indices of D are ordered in increasing dimension with respect to their corresponding basis vectors, i.e. chains, this implies applying Algorithm 3 first to the submatrix of D whose row indices correspond to zero-dimensional chains and column indices correspond to one-dimensional chains.

Proposition 5.1. *Let $RM = DC$ be a U-match decomposition of a square matrix D such that $D^2 = 0$. Then* $\mathrm{def}(\mu) \cap \mathrm{val}(\mu) = \emptyset$.

Proof. Consider the right-reduction $(RM) = DC$ and the matrix E which is defined via

$$\mathrm{COL}_j(E) = \begin{cases} \mathrm{COL}_i(RM) & j = \mathrm{pivot}(\mathrm{COL}_i(RM)) \text{ for some } i, \\ \mathrm{COL}_j(C) & \text{else.} \end{cases}$$

Note that $EM = DE$ is almost a U-match decomposition; E need not be unitriangular however. We mend this by multiplying E by an invertible diagonal matrix F. This yields the U-match decomposition $(EF)M = D(EF)$. Then, $M = (EF)^{-1}D(EF)$ and hence $M^2 = 0$. Since M is a matching matrix, this then implies $\mathrm{def}(\mu) \cap \mathrm{val}(\mu) = \emptyset$. □

In applications where D is not given directly but must rather be constructed, e.g., from point cloud data or a distance matrix in the case of Vietoris-Rips complexes, it can be advantageous to rely on an implicit representation of the matrix, which allows us to compute rows of the boundary matrix on-the-fly, see [1] and [7]. This optimization is then especially advantageous when implemented in unison with aforementioned clearing, as the initial construction of certain rows of D, which are skipped because of clearing, can be avoided in the first place.

Lastly, assume that the i-th iteration of the for loop in Algorithm 3 corresponds to the j-th matched row index, i.e., $i = \rho_j$. If $\mathrm{col}(\rho_j) = \mathrm{pivot}(\mathrm{ROW}_{\rho_j}(D))$, then lines 5-10 in Algorithm 3 will perform no operations on that row. $(\rho_j, \mathrm{col}(\rho_j))$ then is a so-called emergent pair. In applications where D is constructed on-the-fly, we might avoid constructing the ρ_j-th row of the boundary matrix entirely, if we are able to detect this emergent pair early. If furthermore all entries in D below $D[\rho_j, \mathrm{col}(\rho_j)]$ are zero to begin with, then $(\rho_j, \mathrm{col}(\rho_j))$ is a so-called apparent pair, which can be detected even before starting to apply Algorithm 3 to D. Note that, for non-apparent emergent pairs, we may only detect them when all rows below the ρ_j-th have been processed already, i.e., when we know their pivots.

6 IMPLEMENTATION

As a proof of concept, we adapt the software Ripser (v1.2.1), see [1]. It computes barcodes of Vietoris-Rips complexes based on data such as point clouds or distance matrices. The algorithm implemented in Ripser is almost identical to Algorithm 3. Furthermore, Ripser employs all optimization techniques mentioned in Section 5. For simplicity, we treat Ripser as a black box unless relevant to our discussion.

Our aim is to adapt Ripser in such a way that we can compute a decomposition of the filtered chain complex corresponding to the simplexwise refinement of the Vietoris-Rips complex that is generated from the input data. As we are computing with a full simplicial complex, any interval in the barcode, except exactly one in dimension zero, is finite. If $RM = DC$ is a U-match decomposition of the boundary matrix of the filtered chain complex, we are therefore, on one hand, interested in computing columns of C whose index belongs to $\boldsymbol{\kappa}$, i.e., bounding chains of generators of homology. On the other hand, we are interested in the boundaries of those bounding chains. Note that we do not compute bounding chains and their boundaries when their corresponding interval in the barcode of the filtered chain complex is an artifact resulting from the simplexwise refinement of the Vietoris-Rips complex, which is the case if the diameters of the simplices corresponding to the interval bounds are equal, see [1].

Changes we have made to the software, especially code that implements the back-substitution solving for columns of C whose index is in $\boldsymbol{\kappa}$, are illustrated in the Appendix. The entirety of the code can be found at `https://github.com/ahase96/Decomposition-of-Filtered-Chain-Complexes/tree/main`.

Ripser computes in chunks as sketched in Section 5 in order to successfully utilize clearing. For each non-zero dimension, it then executes an algorithm similar to Algorithm 3. After computing intervals of the barcode in a non-zero dimension, we have saved columns of $(R^{-1}_{\boldsymbol{\rho}\mathbf{m}})^{\perp}$ in memory and using an implicit representation of the coboundary operator may compute columns of $((R^{-1}D)_{\boldsymbol{\rho}\mathbf{n}})^{\perp}$. We use this matrix to solve for bounding chains corresponding to endpoints of intervals in persistent homology via back-substitution. Note that this is equivalent to solving for columns of C whose index is in $\boldsymbol{\kappa}$, where C is from the U-match decomposition $RM = DC$ which is determined by $(R^{-1}_{\boldsymbol{\rho}\mathbf{m}})^{\perp}$. We obtain the corresponding boundaries, hence columns of R, by applying an implicit representation of the boundary operator to aforementioned bounding chains.

In order to speed up the computation of bounding chains, we employ an optimization based on Proposition 6.1, which is adapted from [7]. While back-substituting, this allows us to stop solving the linear equation early. In essence, it allows us to obtain columns that are as sparse as possible for some $\tilde{C}$ belonging to some proper U-match decomposition and not necessarily the one determined by $(R^{-1}_{\boldsymbol{\rho}\mathbf{m}})^{\perp}$. This observation is based on the fact that C and $\tilde{C}$ may only differ in columns whose indices lie in $\boldsymbol{\kappa}$.

Proposition 6.1. *Let $RM = DC$ be a proper U-match decomposition. Consider κ_p and some column vector v of appropriate length. Swapping $\mathrm{COL}_{\kappa_p}(C)$ with v results in a matrix $\tilde{C}$ belonging to a proper U-match decomposition $\tilde{R}M = D\tilde{C}$, if and only if $\tilde{C}[\kappa_p, \kappa_p] = 1$ and $\tilde{C}[j, \kappa_p] = 0$ for all $\kappa_p < j$, and $(D\tilde{C})[i, \kappa_p] = 0$ for all $\kappa_p^* < i$.*

As can be seen in Section 7, our implementation of back-substitution requires significant additional resources compared to just computing the barcode. In the following paragraphs, we sketch some alternative approaches to solving the linear equations needed.

First, we might be interested in computing bounding chains whenever we determine a new interval in the barcode. This on-the-fly computation is not feasible however, as the equations we are trying to solve fundamentally depend on the entirety of the barcode in a given dimension; especially the order of the sequence $\boldsymbol{\kappa}$.

Second, we might be interested in solving the linear equations not via back-substitution but via reduction. We have access to columns of $((R^{-1}D)_{\boldsymbol{\rho}\mathbf{n}})^{\perp}$, which is column-reduced. The back-substitution aims to solve an equation of form

$$x((R^{-1}D)_{\boldsymbol{\rho}\mathbf{n}})^{\perp} = b \ .$$

Hence, such an approach via reduction is not feasible.

Third, we might adapt Ripser such that we have rows of $(R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}})^{\perp}$ in memory. Due to the nature of the implicit representation of the boundary operator, we would then have row access to $((R^{-1}D)_{\mathbf{m}\boldsymbol{\kappa}})^{\perp}$, which is row-reduced and would allow for an approach to solve the linear equations via reduction. Note however that an implementation of this approach would have to make sure that indices in $\bar{\boldsymbol{\kappa}}$ are not treated as valid pivots for the appended row corresponding to the solution.

Lastly, assuming that we have access to rows of $(R^{-1}_{\boldsymbol{\rho}\boldsymbol{\rho}})^{\perp}$ and hence also $((R^{-1}D)_{\mathbf{m}\boldsymbol{\kappa}})^{\perp}$, we could also implement an approach to solve the linear equations similar to how back-substitution is implemented in our implementation.

7 EXPERIMENTS

We show the results of some computational experiments, where we compare the performance of our implementation computing decompositions of filtered chain complexes to the performance when just computing their barcodes. Furthermore, we also illustrate the performance when just computing the bounding chain and boundary corresponding to an interval with maximal persistence, i.e., an interval where the difference in diameter between the simplices corresponding to the interval bounds is maximal. The results are shown in Table 3. The experiments were performed on a laptop computer with a 2.40 GHz Intel Core i7 processor and 8 GB RAM. The data sets sphere_3 and rp_2 are taken from [1], while the rest of the data sets are taken from [7].

Data set	# points	(1)	(2)	(3)
sphere_3	192	0s 103ms	0s 549ms	0s 116ms
torus	500	0s 609ms	10s 626ms	0s 708ms
uniform	500	0s 785ms	-	0s 891ms
rp_2	600	2s 491ms	29s 610ms	2s 707ms
cyclo_1000	1000	2s 601ms	-	2s 944ms
henneberg_1000	1000	3s 193ms	-	13s 459ms

Table 3. In (1), we compute only the barcode up to dimension one. In (2), we compute bounding chains and boundaries for all intervals in dimension one whose persistence is positive, i.e., the difference in diameter between the simplices corresponding to the interval bounds is positive. Whenever we write - , the computation has taken more than 60s. In (3), we compute bounding chain and boundary corresponding to an interval with maximal persistence.

We conclude that the computation of bounding chains and boundaries for all intervals requires significant additional resources. The computation of a single bounding chain and boundary corresponding to an interval with maximal persistence requires significantly fewer additional resources, however. That holds at least for most data sets. In the case of henneberg_1000 however, even this computation is significantly more resource-consuming than just computing the barcode. Thus, the results of our experiments suggest that a reconstruction of individual basis elements of a decomposition of a filtered chain complex often but not always can be done with little extra resources. A reconstruction of large parts of a decomposition however requires significant additional resources. It could be of interest to investigate whether alternative approaches as mentioned in Section 6 are feasible and yield better results.

APPENDIX: DETAILS OF THE IMPLEMENTATION

We deactivate the ability to detect apparent pairs in order to have access to $(R_{\rho\mathbf{m}}^{-1})^{\perp}$. We focus on an implementation using coefficients in the field with two elements. We have added methods that compute the boundary or coboundary for a sum of simplices, e.g., boundaries of bounding chains or coboundaries of columns of $(R_{\rho\mathbf{m}}^{-1})^{\perp}$, as needed for our implementation of back-substitution. Note that these are adaptations of already present methods to which we have made only very minor changes. Our methods can be found in lines 735-791 of ripser.cpp in aforementioned repository. Below we display the code that computes bounding chains and their boundaries as mentioned in Section 6 via back-substitution.

After having computed the barcode in a non-zero dimension, we have access to the following data:

- $(R_{\rho\mathbf{m}}^{-1})^{\perp}$ given by reduction_matrix.
- $\boldsymbol{\kappa}$ given by kappa_heap.
- $\boldsymbol{\kappa}^*$ given by global_hash.
- A bijection between $\boldsymbol{\kappa}$ and column indices of reduction_matrix given by local_hash.

```
std::vector<diameter_entry_t> kappa_list = {};

while (true) {

    diameter_entry_t e = pop_pivot(kappa_heap);

    if (get_index(e) == -1) {
        break;
    }

    kappa_list.push_back(e);

}

size_t size_of_matrix = kappa_list.size();

for (size_t j = 0; j < size_of_matrix; j++) {

    diameter_entry_t for_kappa = kappa_list[j];

    diameter_entry_t for_rho = global_hash.find(get_index(for_kappa))->second;

    if (!(get_diameter(for_kappa) > (get_diameter(for_rho) * ratio))) {
        continue;
    }

    bool shortcircuit = false;

    std::priority_queue<diameter_entry_t, std::vector<diameter_entry_t>,
        greater_diameter_or_smaller_index_comp<diameter_entry_t>>
        solution_vector;

    entry_hash_map solution_hash;

    size_t i = j;

    while (true) {
```

```

        diameter_entry_t while_kappa = kappa_list[i];

        diameter_entry_t while_rho = global_hash.find(get_index(while_kappa))->second;

        size_t local_rho = local_hash.find(get_index(while_kappa))->second;

        std::priority_queue<diameter_entry_t, std::vector<diameter_entry_t>,
            greater_diameter_or_smaller_index_comp<diameter_entry_t>>
            working_coboundary;

        std::priority_queue<diameter_entry_t, std::vector<diameter_entry_t>,
            greater_diameter_or_smaller_index_comp<diameter_entry_t>>
            working_boundary;

        own_fast_coboundary(reduction_matrix,while_rho,local_rho,dim,working_coboundary);

        pop_pivot(working_coboundary);

        std::priority_queue<diameter_entry_t, std::vector<diameter_entry_t>,
            greater_diameter_or_smaller_index_comp<diameter_entry_t>>
            solution_scalar;

        while (true) {

            diameter_index_t e = pop_pivot(working_coboundary);

            if (get_index(e) == -1) {
                break;
            }

            if (solution_hash.find(get_index(e)) != solution_hash.end()) {
                solution_scalar.push(for_kappa);
            }

        }

        diameter_entry_t e = pop_pivot(solution_scalar);

        if (i == j) {
            if (get_index(e) == -1) {
                solution_vector.push(while_kappa);
                solution_hash.insert({get_index(while_kappa),get_index(for_kappa)});
            }
        } else {
            if (get_index(e) != -1) {
                solution_vector.push(while_kappa);
                solution_hash.insert({get_index(while_kappa),get_index(for_kappa)});
            }
        }

        own_fast_boundary(solution_vector, dim, working_boundary);

        while (true) {

            diameter_entry_t e = pop_pivot(working_boundary);

            if ((get_index(e) == get_index(for_rho) &&
```

```
            get_index(get_pivot(working_boundary)) == -1)) {
                shortcircuit = true;
                break;
            }

            if (get_index(e) == -1) {
                break;
            }

        }

        if (i == 0 || shortcircuit) {
            break;
        }

        i = i-1;

    }

    std::priority_queue<diameter_entry_t, std::vector<diameter_entry_t>,
        greater_diameter_or_smaller_index_comp<diameter_entry_t>>
        solution_boundary;

    own_fast_boundary(solution_vector, dim, solution_boundary);

    std::cout << "Start of boundary" << std::endl;

    while (true) {

        diameter_entry_t e = pop_pivot(solution_boundary);

        if (get_index(e) == -1) {
            std::cout << "End of boundary" << std::endl;
            break;
        }

        std::cout << get_index(e) << std::endl;

    }

    std::cout << "Start of bounding chain" << std::endl;

    while (true) {

        diameter_entry_t e = pop_pivot(solution_vector);

        if (get_index(e) == -1) {
            std::cout << "End of bounding chain" << std::endl;
            break;
        }

        std::cout << get_index(e) << std::endl;

    }

}
```

REFERENCES

[1] Bauer, Ulrich. Ripser: Efficient computation of Vietoris–Rips persistence barcodes. *Journal of Applied and Computational Topology*, 5(3):391–423, June 2021.

[2] Cohen-Steiner, David and Edelsbrunner, Herbert and Morozov, Dmitriy. Vines and vineyards by updating persistence in linear time. *Proceedings of the Annual Symposium on Computational Geometry*, 2006:119–126, June 2006.

[3] Crawley-Boevey, William. Decomposition of pointwise finite-dimensional persistence modules. arXiv:1210.0819, July 2014.

[4] de Silva, Vin and Morozov, Dmitriy and Vejdemo-Johansson, Mikael. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, November 2011.

[5] Edelsbrunner, Herbert and Harer, John. *Computational Topology: An Introduction*. January 2010.

[6] Edelsbrunner, Herbert and Letscher, David and Zomorodian, Afra. *Topological Persistence and Simplification*. November 2002.

[7] Hang, Haibin and Giusti, Chad and Ziegelmeier, Lori and Henselman-Petrusek, Gregory. U-match factorization: Sparse homological algebra, lazy cycle representatives, and dualities in persistent (co)homology. arXiv:2108.08831, November 2021.

[8] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[9] Turaev, Vladimir. *Lectures in Mathematics. ETH Zürich. Introduction to Combinatorial Torsions*. Birkhäuser Basel, January 2001.

[10] Zomorodian, Afra and Carlsson, Gunnar. Computing persistent homology. *Discrete Computational Geometry*, 3:249–274, February 2005.