The Homology of Data



Nina Otter

New College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2018

for Manfred

as your ashes swim down
the Aare

your voice
reverberates in my ears

Acknowledgements

Thank you to

My supervisors Heather Harrington and Ulrike Tillmann, for your guidance, patience and perseverance in these years that we spent trying to bridge the gap between pure and applied mathematics. For teaching me that being a researcher in mathematics involves so much more than the individual process of discovery, and for guiding my first unsteady steps into the world of applied mathematics. For all the opportunities that you gave me, and your constant support. And for the roadmap paper; when I embarked on this project I hadn't the slightest idea of how many times my dreams would be crushed, or how much mathematics I could possibly forget while mindlessly testing software for over two years, and shifting commas around for the umpteenth revision of the manuscript; yet now that it's all over I am extremely grateful that you believed in me and the project, and kept pushing me when I'd plant my feet, with my hope and motivation long gone. For all of this and much more that I could possibly fit into a paragraph, thank you.

Mason Porter, for your frankness; for being able to keep up a good fight over punctuation, and for teaching me many things about applied mathematics, and clear writing (this, in itself, is a belated declaration of defeat). Hal Schenck, for teaching me much of what I know about combinatorial commutative algebra, for being such an enthusiastic and kind collaborator, and for unblinkingly dealing with my manic rewriting of whatever you wrote. John Baez, for your ongoing mentorship and guidance. Bernd Sturmfels, for being such an inspiration. Martin Bridson and Peter Grindrod, for your vision and support.

Uli Bauer, Mike Lesnick, Matthew Wright, for many helpful discussions and pieces of advice. Jose Perea, for inspiring me with your talks what feels like so many years ago, prompting me to work in topological data analysis. The topological data analysis group at CIMAT, for your enduring hospitality and generosity. The technical support at the Advanced Research Computing Centre, for their help and patience while I was maxing out the machines trying to

make libraries fail. The faculty, researchers, staff and students at the Mathematical Institute in Oxford, for providing such a great environment in which to do mathematics and share ideas; in particular Agnese Barbensi, Bruce Bartlett, Frances Kirwan, Barbara Mahler, Philip Maini, Vicky Neal, Sandy Patel, Claudia Scheimbauer, and Bernadette Stolz.

Barbara, for all our Port Meadow walks and late breakfasts, for giving me a shoulder on which to cry, and a desk under which to crawl. And for teaching me the hard way that first impressions last: no matter what I do, for you I'll probably always just be the "trifle girl". Yoel, for not being afraid of mathematics. R1, Jan, R2, Didi, Clare, Fiona, Gabriele, Gill, Juliet, Lisa, Marina, Steve, Sue, for filling this past year with deep and mundane, life-changing moments — and probably too much chocolate. And for the Roman dinner-clad cockroach.

My family, in particular my mother and my sister, for your unconditional support. And for nodding and smiling while I explain what I do; even if you don't understand a word, you keep trying.

Adrian, you are my pillar.

Oxford, April 2018

Abstract

Techniques and ideas from topology — the mathematical area that studies shapes — are being applied to the study of data with increasing frequency and success. Persistent homology (PH) is a method that uses homology — a tool in topology that gives a measure of the number of holes of a space — to study qualitative features of data across different values of a parameter, which one can think of as scales of resolution, and provides a summary of how long individual features persist across the different resolution scales. In many applications, data depend not only on one, but several parameters, and to apply PH to such data one therefore needs to study the evolution of qualitative features across several parameters. The theory of one-parameter PH is well-understood, but PH is computationally expensive and ways to speed up the computation are an object of current research. By contrast, the theory of multiparameter PH is hard, and it presents one of the biggest challenges in the topological study of data. In this thesis I address computational and theoretical problems in the application of homology to the study of data: I give a survey of the computation of PH and its challenges, and benchmark all state-of-the-art libraries for the computation of one-parameter PH; I propose new invariants suitable for applications for multiparameter persistent homology; finally, I relate magnitude homology, a homology theory for finite metric spaces that has been recently introduced, to persistent homology.

Contents

Introduction **1**

1.1 Persistent homology . . . 2
1.2 Magnitude homology . . . 4
1.3 State of the field . . . 4
1.3.1 One-parameter PH . . . 4
1.3.2 Multiparameter PH . . . 5
1.3.3 Magnitude homology . . . 6
1.4 Contributions . . . 7
1.4.1 A roadmap for the computation of persistent homology . . . 7
1.4.2 Stratifying multiparameter persistent homology . . . 7
1.4.3 Magnitude meets persistence. Homology theories for persistent simplicial sets . . . 8
1.5 Statement of authorship . . . 8

2 A roadmap for the computation of persistent homology **9**

2.1 Introduction . . . 9
2.2 Related work . . . 15
2.3 Homology . . . 16
2.3.1 Simplicial complexes and their homology . . . 16
2.3.2 Building simplicial complexes . . . 20
2.4 Persistent homology . . . 21
2.4.1 Filtered complexes and homology . . . 22
2.5 Computation of PH for data . . . 25
2.5.1 Data . . . 25
2.5.1.1 Networks . . . 25
2.5.1.2 Digital images . . . 26
2.5.1.3 Finite metric spaces . . . 26
2.5.2 Filtered Simplicial Complexes . . . 27
2.5.2.1 Vietoris–Rips complex . . . 27

2.5.2.2 The Delaunay complex . . . 28
2.5.2.3 Alpha complex . . . 29
2.5.2.4 Witness complexes . . . 30
2.5.2.5 Additional complexes . . . 31
2.5.2.6 Reduction techniques . . . 31
2.5.3 From a Filtered Simplicial Complex to Barcodes . . . 33
2.5.3.1 Standard algorithm . . . 33
2.5.3.2 Reading off the intervals . . . 34
2.5.3.3 Other algorithms . . . 34
2.5.4 Statistical interpretation of topological summaries . . . 36
2.5.5 Stability . . . 38
2.6 Excursus: Generalised persistence . . . 38
2.7 Software . . . 39
2.7.1 Benchmarking . . . 40
2.7.1.1 Data sets . . . 42
2.7.1.2 Machines and compilers . . . 46
2.7.1.3 Tests and results . . . 46
2.7.2 Conclusions from our benchmarking . . . 47
2.8 Future directions . . . 51
2.9 Availability of data and materials . . . 51
2.10 Additional acknowledgements . . . 51

3 Stratifying multiparameter persistent homology 53
3.1 Introduction . . . 53
3.1.1 Persistent homology . . . 55
3.1.2 Related work . . . 56
3.1.3 Structure of the chapter . . . 57
3.2 Multiparameter persistence and multigraded algebra . . . 58
3.2.1 Multiparameter persistence . . . 59
3.2.2 Multigraded algebra . . . 60
3.2.3 Multigraded modules as multiparameter persistence modules . . . 63
3.3 The rank of a module and its Hilbert function . . . 65
3.3.1 The Hilbert function . . . 66
3.3.2 The Hilbert polynomial and Hilbert series . . . 67
3.3.3 Examples . . . 68
3.4 Associated primes and stratification . . . 73
3.4.1 Associated primes of multigraded modules . . . 73

3.4.2 Minimal associated primes and support shape . . . 75
3.4.3 Embedded associated primes and stratification of the support shape 77
3.4.4 Persistent but not fully persistent components: the rank of M along a coordinate subspace . . . 80
3.4.5 Syzygies and the rank invariant . . . 82
3.4.6 Computation of associated primes . . . 83
3.5 From MPH to PH: restriction to a line and intersection with the diagonal . 84
3.5.1 Restriction to a line . . . 85
3.5.2 Intersecting with the diagonal . . . 86
3.6 Conclusion . . . 88
3.7 Additional acknowledgments . . . 88

4 Magnitude meets persistence. Homology theories for filtered simplicial sets 89
4.1 Introduction . . . 90
4.1.1 Structure of the chapter . . . 94
4.2 Simplicial complexes and simplicial sets . . . 94
4.3 Enriched categories and Lawvere metric spaces . . . 97
4.4 Filtered simplicial sets . . . 98
4.5 Persistent vs. graded objects . . . 99
4.6 Coends . . . 100
4.7 Magnitude . . . 101
4.8 Magnitude homology . . . 102
4.8.1 Magnitude homology as Hochschild homology . . . 103
4.8.2 Magnitude homology: an alternative viewpoint . . . 104
4.8.3 Magnitude homology categorifies magnitude . . . 107
4.9 Persistent homology . . . 108
4.10 Magnitude meets persistence . . . 109
4.11 Limit homology . . . 111
4.12 An Euler characteristic for persistent homology and the persistence function 115
4.13 Additional acknowledgments . . . 116

A Tutorial 118
A.1 Installation . . . 118
A.1.1 Dionysus . . . 118
A.1.2 DIPHA . . . 120
A.1.3 Gudhi . . . 120

A.1.4 javaPlex . . . 120
A.1.5 hera . . . 120
A.1.6 jHoles . . . 121
A.1.7 Perseus . . . 121
A.1.8 ripser . . . 121
A.2 Computation of PH for networks . . . 121
A.2.1 Sample network data . . . 121
A.2.2 Adjacency matrix versus edge-list file . . . 122
A.2.3 PH with the WRCF (with jHoles) . . . 122
A.2.4 Networks as point clouds . . . 123
A.3 Computation of PH for point clouds . . . 123
A.3.1 Sample point-cloud data . . . 124
A.3.2 Distance matrices versus points clouds . . . 124
A.3.3 VR complex . . . 124
A.3.3.1 Input data . . . 124
A.3.3.2 Dionysus . . . 125
A.3.3.3 DIPHA . . . 126
A.3.3.4 Gudhi . . . 127
A.3.3.5 javaPlex . . . 127
A.3.3.6 Perseus . . . 128
A.3.3.7 ripser . . . 128
A.3.4 Alpha . . . 129
A.3.4.1 Dionysus . . . 129
A.3.4.2 Gudhi . . . 130
A.3.5 Čech . . . 130
A.3.6 Witness . . . 131
A.4 Computation of PH for image data . . . 132
A.4.1 DIPHA . . . 132
A.4.2 Perseus . . . 133
A.4.3 Gudhi . . . 133
A.4.4 Images as point clouds . . . 133
A.5 Barcodes and persistence diagrams . . . 134
A.6 Statistical interpretation of barcodes . . . 136
A.6.1 Bottleneck distance . . . 137
A.6.1.1 Dionysus . . . 137
A.6.1.2 hera . . . 137
A.6.2 Wasserstein distance . . . 138

A.6.2.1 Dionysus . . . 138
A.6.2.2 hera . . . 138

B Additional computations **139**
B.1 Vietoris–Rips complex . . . 139
B.2 Image data set . . . 140
B.3 Additional functionalities . . . 142

C Code **144**
C.1 Computation of homology of chain complex . . . 144
C.2 Hilbert function and series, local cohomology . . . 146
C.3 Rank . . . 150

Bibliography **152**

List of Figures

1.1 Examples of complex data. 2
1.2 Pipeline for (one-parameter) persistent homology 3
1.3 Examples of data sets that one would wish to study with multiparameter persistent homology. 4

2.1 Example of persistent homology for a point cloud 11
2.2 Example of persistent homology for a gray-scale digital image 13
2.3 A simple example 18
2.4 Examples to illustrate simplicial homology 19
2.5 Example of persistent homology for a finite filtered simplicial complex . . . 23
2.6 PH pipeline. 25
2.7 Example of PH computation with the standard algorithm. 35
2.8 Two well-known examples 45

3.1 A one-critical bifiltration. 69
3.2 A one-critical bifiltration. 70
3.3 A one-critical bifiltration with a one-cycle that is the boundary of two different 2-chains with non-comparable entry degrees. 72
3.4 Two filtrations which are not one-critical. 73
3.5 Examples of stratifications of modules. 76
3.6 Example of information forgotten by the associated primes. 78

A.1 Schematic for how to navigate the tutorial. 119
A.2 Persistence diagram and barcode for a computation of PH in dimension 1 with the alpha complex. 135
A.3 Barcode and persistence diagram for a computation of PH in dimension 0 with the VR complex. 136

List of Tables

2.1 Overview of complexes used for PH. 32
2.2 Overview of software for the computation of PH. 41
2.3 Performance of the software packages measured in wall-times and CPU seconds. 48
2.4 Memory usage of the software packages. 49
2.5 Maximal size of simplicial complex supported by the libraries. 49
2.6 Guidelines for which implementation is best-suited to which data set. . . . 50

3.1 A brief overview of the dictionary between terms used in persistent homology and commutative algebra proposed. 54

B.1 Computations of PH with VR complex for the data set **drag 1**. 139
B.2 Computations of PH with VR complex for the data set **house**. 140
B.3 Computations of PH with VR complex for the data set **senate**. 140
B.4 Computations of PH with VR complex for the data set **netw-sc**. 140
B.5 Computations on the shared-memory system. 141
B.6 Computations of PH with cubical complexes for the data set **vertebra** on the shared-memory system. 141
B.7 Computations of alpha and Čech complexes on the shared-memory system. 142
B.8 Computations of PH with the witness complex on the shared-memory system. 143

Introduction

> Many people consider mathematics to be a boring and formal science. However, any really good work in mathematics always has in it: beauty, simplicity, exactness, and crazy ideas. This is a strange combination. I understood earlier that this combination is essential in classical music and poetry, for example. But it is also typical in mathematics. Perhaps it is not by chance that many mathematicians enjoy serious music. [...] When we think about music, we do not divide it into specific areas as we often do in mathematics. If we ask a composer what is his profession, he will answer, "I am a composer." He is unlikely to answer, "I am a composer of quartets." Maybe this is the reason why, when I am asked what kind of mathematics I do, I just answer, "I am a mathematician."
>
> — Israel M. Gelfand [91]

Scientists are increasingly confronted with data that are produced using new technologies, which are often not yet theoretically understood. The size of such data presents many challenges, but what is called "complexity" of the data presents equal if not greater challenges. "Complex data" is an umbrella term that is used to denote data that is unstructured, noisy, dynamic, incomplete, or corrupted. Not all these terms have a well-defined agreed-upon meaning. In the last 20 years methods from topology, the mathematical area that studies "shapes", have proven successful in studying data that is complex, and whose underlying shape is not known a priori, from discovering new cancer subtypes [164], to classifying nanoporous materials [134]. This practice has become known as topological data analysis (TDA). As additional methods from topology still find their application in the study of complex structure in data, the practice is evolving and expanding, and now moreover draws increasingly upon data science, computer science, computational algebra, computational topology, computational geometry, and statistics. In this synthesis of seemingly disparate areas of mathematics, TDA is developing paradigms of its own, and is becoming an area of specialisation in its own right.

1.1 Persistent homology

In topology, one is interested in studying homotopy invariants of spaces, namely properties that are not changed when the space is continuously deformed. One such invariant is the Euler characteristic χ, which can be thought of as an analogue of the cardinality of a set, as was first made precise by Schanuel [181].

The Euler characteristic is closely related to another invariant, namely the homology of a space. For every $i = 0, 1, 2, \ldots$ the ith (singular) homology of a space X is an abelian group $H_i(X)$, whose rank is called the i-th Betti number $\beta_i(X)$ of the space and gives a count of its number of i-dimensional holes. One then has

$$\chi(X) = \sum_{i=0}^{\infty} (-1)^i \beta_i(X) \tag{1.1}$$

whenever the sum and summands are finite; one says that homology **categorifies** the Euler characteristic.

Given a data set such as a finite set of points sampled from a circle with noise (see Fig. 1.1(a)), we would like to ask questions such as: how many clusters does this data set have? How many holes? **Persistent homology** (PH) gives an automated way of detecting the shape that would be observed by squinting, by describing how the number of holes in different dimensions change as we change the scale of the metric space.



(a) (b)

Figure 1.1: Examples of complex data. (a) A noisy sample of points from a circle, and (b) a blurred grey-scale digital image.

Persistent homology is one of the most successful methods in TDA. Given a finite metric space as in Fig. 1.1(a), one considers a "thickening" of the metric space at different distance scales, which gives a nested sequence (a so-called **filtration**) of spaces, and then analyses the evolution ("persistence") of topological features across this filtration (see Fig. 1.2(a)). Given a grey-scale digital image, one can associate to it a filtration

by thresholding the pixels or voxels by increasing grey values (see Fig. 1.2(b)). The topological features that one examines include connected components (H_0), holes (H_1), and higher-dimensional holes. The **barcode** is an algebraic invariant that summarises how topological features of a certain dimension evolve across the nested sequence: the left endpoint of an interval represents the birth of a feature, while its right endpoint represents the death of the same feature. When a feature is still "alive" at the largest radius or grey-scale value that one considers, the lifetime interval is an infinite interval.

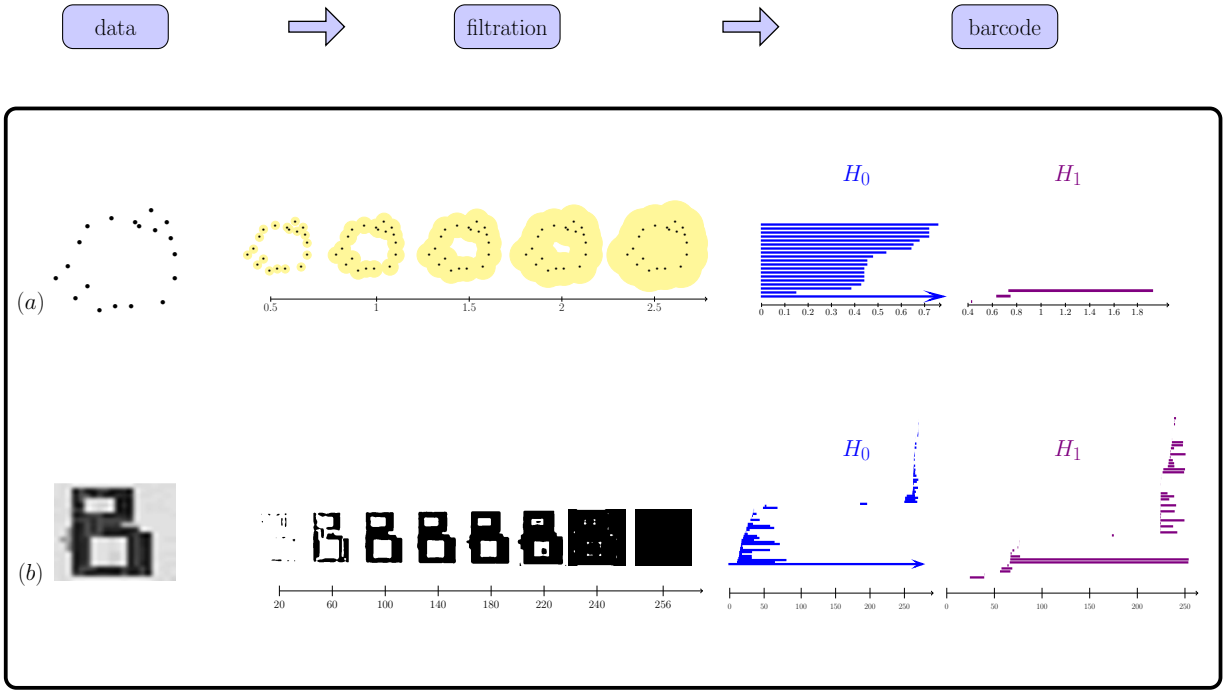
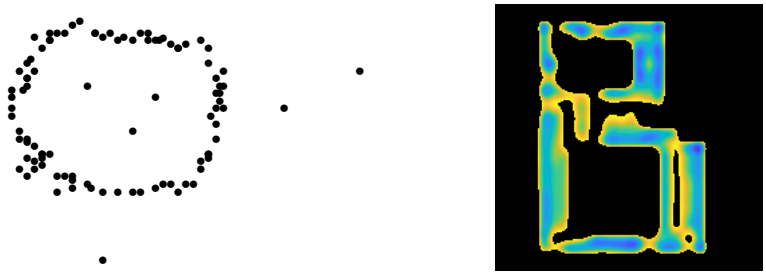


Figure 1.2: Pipeline for (one-parameter) persistent homology for the data sets in Fig. 1.1. In (a) we take the union of balls with same radius around the points; letting the radius vary we obtain a filtration. In (b) we threshold the pixels by increasing grey values.

In many applications one would like to associate to a data set a family of spaces that depend on multiple parameters, either because the data set intrinsically depends on more than one parameter, such as, e.g., coloured digital images (see Fig. 1.3(b)), or because the complexity of the data requires a more refined analysis. The latter case arises for instance in the presence of samples of points with outliers, as illustrated in Fig. 1.3(a). While one-parameter PH is stable with respect to small perturbations in the input data, it is not stable with respect to outliers; for instance, the barcode for homology in degree 1 that one obtains for the point cloud in Fig. 1.3(a) contains no interval that is significantly longer than the other intervals. Thus, one wishes to associate to such a data set a 2-parameter family of spaces, where the parameters are given by the distance and a density function of the points, see [140, Section 1.5] for details.



(a) (b)

Figure 1.3: Examples of data sets that one would wish to study with multiparameter persistent homology. (a) Points sampled from a circle with noise and outliers, and (b) a coloured digital image.

1.2 Magnitude homology

In his work on the generalisation of cardinality-like invariants, Leinster introduced the **magnitude** of a metric space [137], which is a real number that gives the "effective number of points" of the space. The definition of magnitude of a metric space depends on a choice of scale of the distances of the points. Thus, one usually introduces the **magnitude function**, which describes how the "effective number of points" changes as one varies the scale. The magnitude is analogous to the Euler characteristic of a topological space, and thus a natural question to ask is whether there is a homology theory for metric spaces that categorifies the magnitude; namely, a homology theory that is related to the magnitude in a similar way as the homology of a topological space is related to its Euler characteristic, as expressed in (1.1).

Recently, Leinster and Shulman answered this question affirmatively, by introducing **magnitude homology**, a homology theory for metric spaces [139], which encodes the information given by the magnitude function.

1.3 State of the field

Here we summarise briefly the state of the field for one-parameter and multiparameter persistent homology, and for magnitude homology.

1.3.1 One-parameter PH

One-parameter PH is one of the most successful methods in TDA. The first algorithm for the computation of PH was introduced in [76]. Since then the theory of PH has

been extensively developed and studied, and is considered to be well-understood. Most efforts have been and still are currently being directed towards addressing computational challenges: many methods used in applications to associate filtered spaces to data sets yield spaces that have size $\mathcal{O}(2^n)$, where n is the number of data points (in practice, one thresholds the dimension of the filtered space by some value $k << n$, and the size is then reduced to $\mathcal{O}(n^{k+1})$); the algorithm for the computation of PH introduced in [76] has computational complexity that is cubical in the worst case with respect to the size of the filtered space associated to the data set (we note however that the complexity observed in practice is linear for most data sets); the geometric properties of the space of barcodes pose challenges from the statistical point of view. We refer the reader to Section 2.5 for a survey of what has and is being done to address these computational challenges.

1.3.2 Multiparameter PH

Multiparameter persistent homology (MPH) was first introduced in [38], where the authors showed that in a precise sense there is no generalisation of the barcode for $r > 1$ (where r denotes the number of parameters). Research on MPH subsequently stagnated, however in recent years there has been renewed interest and currently many different avenues to finding invariants for MPH that are suitable for applications are being explored. In the following we briefly discuss the main approaches:

- Efficient computation of homology of multifiltrations: in [37] the authors propose a polynomial-time algorithm to compute a presentation of homology of "one-critical" multifiltered simplicial complexes, which roughly are multifiltered simplicial complexes in which each simplex enters the complex at exactly one filtration value. In [37] a polynomial-time algorithm for the computation of the homology of an arbitrary multifiltered simplicial complex is proposed. We are currently working on an optimisation of this algorithm [63], with plans to release an implementation of this optimisation as a software package for `Macaulay2` [99], one of the standard computational algebra libraries.

- Studying multiparameter modules that come from applications: while it is hopeless to find complete invariants for general multiparameter persistence modules, it is reasonable to assume that multiparameter persistence modules that arise in applications are simpler to understand. Researchers exploring this avenue of research include U. Bauer, M. Botnan, S. Oppermann, J. Steen, W. Chacholski, and the author of this thesis.

- Restricting 2-parameter persistence to one-parameter persistence: for $r = 2$ one can probe the support of a persistence module with a line of positive slope and study the barcode associated to the corresponding one-parameter module. This line of research was first introduced in [20], where the authors show that the collection of one-parameter modules associated to a 2-parameter module by restricting it to all possible lines with positive slope gives the same information as the rank invariant, an invariant that was introduced in [38] (and which in the one-parameter case is equivalent to the barcode). Lesnick and Wright have developed this approach further and released a software package that allows to visualise barcodes along one-dimensional slices of 2-parameter persistence modules [140].

- Noise: in [183] the authors take the view that the information contained in a multiparameter persistence module is only defined up to some notion of noise in the data set. They develop a general framework to make this precise and propose invariants associated to multiparameter persistence modules that depend on the choice of a specific abstract noise system. In general, these invariants are not computable.

- Data structures for non-discrete multiparameter persistence modules: in [154] Miller develops a theoretical framework to deal with persistence modules that arise from a specific application in which the data set is not finite, and thus one needs to study the evolution of topological features in a continuous setting.

We note that while there are further approaches that are being explored, such as [46, 118, 124], these belong more properly to the wider context of the study of algebras of wild type, rather than multiparameter persistent homology.

1.3.3 Magnitude homology

The magnitude of a metric space was introduced by Leinster in [137]; recently, Leinster and Shulman have proposed magnitude homology as a categorification of the magnitude [139]. Magnitude (homology) is currently a very active area of research, see e.g., [10, 95, 151, 152, 212] for magnitude and [105, 110, 117] for magnitude homology. At the end of [139] the authors list open problems, among which are the following:

- Is there a "blurred" version of magnitude homology that recognises when the triangle inequality is not a strict equality?

- Is magnitude homology related to persistent homology?

1.4 Contributions

In the following we discuss our contributions to the field.

1.4.1 A roadmap for the computation of persistent homology

The paper "A roadmap for the computation of persistent homology" [165], joint with M. Porter, U. Tillmann, P. Grindrod and H. Harrington, was published in *EPJ Data Science*, Springer Nature, and we reproduce it in Chapter 2. In this paper we introduce PH to computational scientists, give a survey of the computation of PH and its challenges, and benchmark all state-of-the-art open source libraries for the computation of persistent homology. Based on these benchmarkings we give guidelines for which libraries are best suited to specific data sets, and we provide specific directions on how to use the libraries in a tutorial and accompanying scripts. The benchmarkings, as well as test data sets used, are nowadays widely considered to be the standard of reference for the test of new algorithms and implementations for the computation of persistent homology. Furthermore, this survey is to our knowledge the only introduction to persistent homology with an eye towards applications and concrete computational challenges; as persistent homology is increasingly used in a wide range of applications it is of fundamental importance that its theory and techniques are made easily accessible to a broad audience of computational scientists.

1.4.2 Stratifying multiparameter persistent homology

The paper "Stratifying multiparameter persistent homology" [102], joint with H. Harrington, U. Tillmann and H. Schenck is currently in the second stage of revisions for *Journal of Applied and Computational Topology*, Springer, and we reproduce it in Chapter 3. In this paper we use methods from algebraic geometry to propose new invariants for multiparameter persistent homology that are suitable for applications. Specifically, we propose multigraded associated primes and local cohomology as invariants that generalise in a suitable sense the barcode; these give information about submodules of a multiparameter persistent module that are generated by elements that live along certain subsets of the support of the module. This approach is novel in both the techniques used, as well as in the question being studied, as most of the approaches that are currently being undertaken focus on direct summands of modules, or on restrictions to subsets of the support (see overview in 1.3.2). The potential impact of our invariants for applications is still to be explored.

1.4.3 Magnitude meets persistence. Homology theories for persistent simplicial sets

In the manuscript "Magnitude meets persistence. Homology theories for persistent simplicial sets" (see Chapter 4), we relate persistent homology to magnitude homology as two different ways of computing homology of filtered simplicial sets. We give an answer to the two open problems listed in 1.3.3, which we show are intertwined. We show that magnitude homology cannot be directly related to persistent homology, answering the second question; we define a blurred version of magnitude homology and show that it is the persistent homology of a certain filtered simplicial set, hence answering the first question. Futhermore, we show how blurred and ordinary magnitude homology differ in the limit: blurred magnitude homology coincides with Vietoris homology, while magnitude homology is trivial. Based on this comparison we conjecture the existence of an Euler characteristic for persistent homology and the magnitude (function) that it would encode. Whether this coincides with the magnitude function given by magnitude homology is an open question; if yes, the behaviour in the limit of magnitude homology, as well as recent findings [110], clarify that the right way to categorify the magnitude is given by the blurred version of magnitude homology. If not, then this invariant should give much richer information about the geometry of a metric space than the magnitude. Thus, this manuscript gives an answer to an open problem in magnitude homology, which is currently a very active field of research. Furthermore, the connection between magnitude homology and persistent homology opens a new avenue of research for persistent homology, of which our conjecture might be but the first manifestation.

1.5 Statement of authorship

In addition to where this is specified directly in the text, in the following we clarify which parts of the thesis are not solely our own work:

- For the paper "A roadmap for the computation of persistent homology" (Chapter 2) N. Otter did the benchmarking and wrote the original version of the manuscript; all authors designed the study and revised the manuscript; N. Otter, M. Porter and H. Harrington obtained the example data sets to be analysed.

- The paper "Stratifying multiparameter persistent homology" (Chapter 3) was a collaboration. The names of the authors are listed in alphabetical order.

Chapter 2

A roadmap for the computation of persistent homology[1]

Persistent homology (PH) is a method used in topological data analysis (TDA) to study qualitative features of data that persist across multiple scales. It is robust to perturbations of input data, independent of dimensions and coordinates, and provides a compact representation of the qualitative features of the input. The computation of PH is an open area with numerous important and fascinating challenges. The field of PH computation is evolving rapidly, and new algorithms and software implementations are being updated and released at a rapid pace. The purposes of our work are to (1) introduce theory and computational methods for PH to a broad range of computational scientists and (2) provide benchmarks of state-of-the-art implementations for the computation of PH. We give a friendly introduction to PH, navigate the computation of PH with an eye towards applications, and use a range of synthetic and real-world data sets to evaluate currently available open-source implementations for the computation of PH. Based on our benchmarking, we indicate which algorithms and implementations are best suited to different types of data sets. In an accompanying tutorial, we provide guidelines for the computation of PH. We make publicly available all scripts that we wrote for the tutorial, and we make available the processed version of the data sets used in the benchmarking.

2.1 Introduction

The amount of available data has increased dramatically in recent years, and this situation — which will only become more extreme — necessitates the development of innovative and efficient data-processing methods. Making sense of the vast amount of data is difficult: on

[1]This is the reproduction of the paper A roadmap for the computation of persistent homology, N. Otter, M. Porter, U. Tillmann, P. Grindrod, H. Harrington, published in EPJ Data Science (2017) **6**:17, Springer Nature.

one hand, the sheer size of the data poses challenges; on the other hand, the complexity of the data, which includes situations in which data is noisy, high-dimensional, and/or incomplete, is perhaps an even more significant challenge. The use of clustering techniques and other ideas from areas such as computer science, machine learning, and uncertainty quantification — along with mathematical and statistical models — are often very useful for data analysis (see, e.g., [90, 97, 119, 180] and many other references). However, recent mathematical developments are shedding new light on such "traditional" ideas, forging new approaches of their own, and helping people to better decipher increasingly complicated structure in data.

Techniques from the relatively new subject of "topological data analysis" (TDA) have provided a wealth of new insights in the study of data in an increasingly diverse set of applications — including sensor-network coverage [60], proteins [89, 125, 214, 215], three-dimensional structure of DNA [84], development of cells [178], stability of fullerene molecules [213], robotics [19,172,199], signals in images [45,101], periodicity in time series [169], cancer [52,65,164,189], phylogenetics [33,41,83], natural images [36], the spread of contagions [142,194], self-similarity in geometry [146], materials science [106,126,127,134], financial networks [94,135], diverse applications in neuroscience [18,55,56,70,96,116,143, 217], classification of weighted networks [190], collaboration networks [39,167], analysis of mobile phone data [9], collective behavior in biology [196], time-series output of dynamical systems [147], and more. There are numerous others, and new applications of TDA appear in journals and preprint servers increasingly frequently. There are also interesting computational efforts, such as [203].

TDA is a field that lies at the intersection of data analysis, algebraic topology, computational geometry, computer science, statistics, and other related areas. The main goal of TDA is to use ideas and results from geometry and topology to develop tools for studying qualitative features of data. To achieve this goal, one needs precise definitions of qualitative features, tools to compute them in practice, and some guarantee about the robustness of those features. One way to address all three points is a method in TDA called *persistent homology* (PH). This method is appealing for applications because it is based on algebraic topology, which gives a well-understood theoretical framework to study qualitative features of data with complex structure, is computable via linear algebra, and is robust with respect to small perturbations in input data.

Types of data sets that can be studied with PH include finite metric spaces, digital images, level-sets of real-valued functions, and networks (see Section 2.5.1). In the next two paragraphs, we give some motivation for the main ideas of persistent homology by discussing two examples of such data sets.

Finite metric spaces are also called "point cloud" data sets in the TDA literature. From a topological point of view, finite metric spaces do not contain any interesting information. One thus considers a "thickening" of a point cloud at different "scales of resolution" and then analyses the evolution of the resulting shape across the different resolution scales. The qualitative features are given by topological invariants, and one can represent the variation of such invariants across the different resolution scales in a compact way to summarise the "shape" of the data.

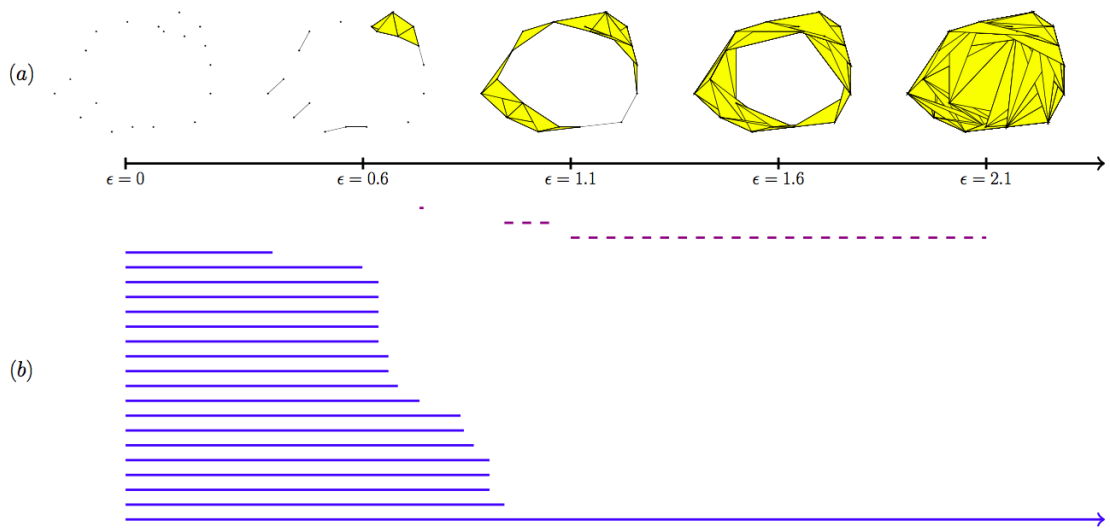


Figure 2.1: Example of persistent homology for a point cloud. (a) A finite set of points in $\mathbb{R}^2$ (for $\epsilon = 0$) and a nested sequence of spaces obtained from it (from $\epsilon = 0$ to $\epsilon = 2.1$). (b) Barcode for the nested sequence of spaces illustrated in (a). Solid lines represent the lifetime of components, and dashed lines represent the lifetime of holes.

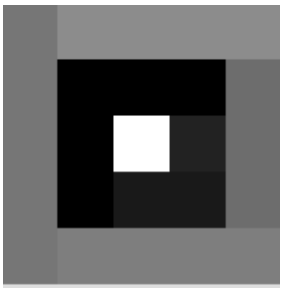
As an illustration, consider the set of points in $\mathbb{R}^2$ that we show in Fig. 2.1. Let ϵ, which we interpret as a "distance parameter," be a nonnegative real number (so $\epsilon = 0$ gives the set of points). For different values of ϵ, we construct a space S_ϵ composed of vertices, edges, triangles, and higher-dimensional polytopes according to the following rule: We include an edge between two points i and j if and only if the Euclidean distance between them is no larger than ϵ; we include a triangle if and only if all of its edges are in S_ϵ; we include a tetrahedron if and only if all of its face triangles are in S_ϵ; and so on. For $\epsilon \leq \epsilon'$, it then follows that the space S_ϵ is contained in the space $S_{\epsilon'}$. This yields a nested sequence of spaces, as we illustrate in Fig. 2.1(a). Our construction of nested spaces gives an example of a "filtered Vietoris–Rips complex," which we define and discuss in Section 2.5.2.

By using homology, a tool in algebraic topology, one can measure several features of the spaces S_ϵ — including the numbers of components, holes, and voids (higher-dimensional versions of holes). One can then represent the lifetime of such features using a finite collection of intervals known as a "barcode." Roughly, the left endpoint of an interval represents the birth of a feature, and its right endpoint represents the death of the same feature. In Fig. 2.1(b), we reproduce such intervals for the number of components (blue solid lines) and the number of holes (violet dashed lines). In Fig. 2.1(b), we observe a dashed line that is significantly longer than the other dashed lines. This indicates that the data set has a long-lived hole. By contrast, one can potentially construe the shorter dashed lines as noise. However, note that while widespread, such an interpretation is not correct in general; for applications in which one considers some short and medium-sized intervals as features rather than noise, see [18, 192]. When a feature is still "alive" at the largest value of ϵ that we consider, the lifetime interval is an infinite interval, which we indicate by putting an arrowhead at the right endpoint of the interval. In Fig. 2.1(b), we see that there is exactly one solid line that lives up to $\epsilon = 2.1$. One can use information about shorter solid lines to extract information about how data is clustered in a similar way as with linkage-clustering methods [90].

One of the most challenging parts of using PH is the statistical interpretation of results. From a statistical point of view, a barcode like the one in Fig. 2.1(b) is an unknown quantity that one is trying to estimate; one therefore needs methods for quantitatively assessing the quality of the barcodes that one obtains with computations. The challenge is twofold. On one hand, there is a cultural obstacle: practitioners of TDA often have backgrounds in pure topology and are not well-versed in statistical approaches to data analysis [5]. On the other hand, the space of barcodes lacks geometric properties that would make it easy to define basic concepts such as mean, median, and so on. Current research is focused both on studying geometric properties of this space and on studying methods that map this space to spaces that have better geometric properties for statistics. In Section 2.5.4, we give a brief overview of the challenges and current approaches for statistical interpretation of barcodes. This is an active area of research and an important endeavor, as few statistical tools are currently available for interpreting results in applications of PH.

We now discuss a second example related to digital images. (For an illustration, see Fig. 2.2(a).) Digital images have a "cubical" structure, given by the pixels (for 2-dimensional digital images) or voxels (for 3-dimensional images). Therefore, one approach to study digital images uses combinatorial structures called "cubical complexes." (For a different approach to the study of digital images, see Section 2.5.1.) Roughly, cubical complexes are topological spaces built from a union of vertices, edges, squares, cubes,

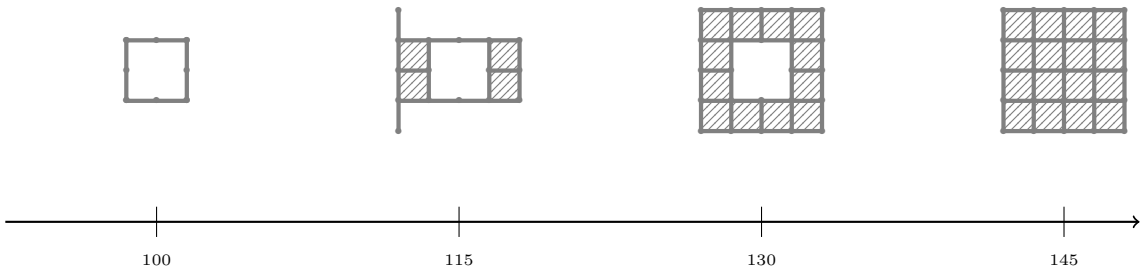
(a)



(b)

$$G = \begin{pmatrix} 115 & 119 & 119 & 119 & 119 \\ 115 & 94 & 94 & 94 & 114 \\ 115 & 94 & 139 & 100 & 114 \\ 115 & 94 & 99 & 99 & 114 \\ 115 & 117 & 117 & 117 & 117 \end{pmatrix}$$

(c)



(d)



Figure 2.2: Example of persistent homology for a gray-scale digital image. (a) A gray-scale image, (b) the matrix of gray values, (c) the filtered cubical complex associated to the digital image, and (d) the barcode for the nested sequence of spaces in panel (c). A solid line represents the lifetime of a component, and a dashed line represents the lifetime of a hole.

and higher-dimensional hypercubes. An efficient way [202] to build a cubical complex from a 2-dimensional digital image consists of assigning a vertex to every pixel, then joining vertices corresponding to adjacent pixels by an edge, and filling in the resulting squares. One proceeds in a similar way for 3-dimensional images. One then labels every vertex with an integer that corresponds to the gray value of the pixel, and one labels edges (respectively, squares) with the maximum of the values of the adjacent vertices (respectively, edges). One can then construct a nested sequence of cubical complexes $C_0 \subset C_1 \subset \cdots \subset C_{256}$, where for each $i \in \{0, 1, \ldots, 256\}$, the cubical complex C_i contains all vertices, edges, squares, and cubes that are labeled by a number less than or equal to i. (See Fig. 2.2(c) for an example.) Such a sequence of cubical complexes is also called a "filtered cubical complex." Similar to the previous example, one can use homology to measure several features of the spaces C_i (see Fig. 2.2(d)).

In the present chapter, we focus on persistent homology, but there are also other methods in TDA — including the Mapper algorithm [187], Euler calculus (see [93] for an introduction with an eye towards applications), cellular sheaves [53, 93], and many more. We refer readers who wish to learn more about the foundations of TDA to the article [34], which discusses why topology and functoriality are essential for data analysis. We point to several introductory papers, books, and two videos on PH at the end of Section 4.9.

The first algorithm for the computation of PH was introduced for computation over $\mathbb{F}_2$ (the field with two elements) in [76] and over general fields in [224]. Since then, several algorithms and optimisation techniques have been presented, and there are now various powerful implementations of PH [12, 13, 16, 149, 158, 161, 193]. Those wishing to try PH for computations may find it difficult to discern which implementations and algorithms are best suited for a given task. The field of PH is evolving continually, and new software implementations and updates are released at a rapid pace. Not all of them are well-documented, and (as is well-known in the TDA community), the computation of PH for large data sets is computationally very expensive.

To our knowledge, there exists neither an overview of the various computational methods for PH nor a comprehensive benchmarking of the state-of-the-art implementations for the computation of persistent homology. In the present chapter, we close this gap: we introduce computation of PH to a general audience of applied mathematicians and computational scientists, offer guidelines for the computation of PH, and test the existing open-source published libraries for the computation of PH.

The rest of our paper is organised as follows. In Section 2.2, we discuss related work. We then introduce homology in Section 2.3 and introduce PH in Section 4.9. We discuss the various steps of the pipeline for the computation of PH in Section 2.5, and we briefly examine algorithms for "generalised persistence" in Section 2.6. In Section 2.7, we give

an overview of software libraries, discuss our benchmarking of a collection of them, and provide guidelines for which software or algorithm is better suited to which data set. (We provide specific guidelines for the computation of PH with the different libraries in the tutorial in Appendix A.) In Section 2.8, we discuss future directions for the computation of PH.

2.2 Related work

In our work, we introduce PH to non-experts with an eye towards applications, and we benchmark state-of-the-art libraries for the computation of PH. In this section, we discuss related work for both of these points.

There are several excellent introductions to the theory of PH (see the references at the end of Section 2.4.1), but none of them emphasises the actual computation of PH by providing specific guidelines for people who want to do computations. In the present paper, we navigate the theory of PH with an eye towards applications, and we provide guidelines for the computation of PH using the open-source libraries Perseus, Dionysus, DIPHA, javaPlex, and Gudhi. We include a tutorial (see Appendix A) that gives specific instructions for how to use the different functionalities that are implemented in these libraries. Much of this information is scattered throughout numerous different papers, websites, and even source code of implementations, and we believe that it is beneficial to the applied mathematics community (especially people who seek an entry point into PH) to find all of this information in one place. The functionalities that we cover include plots of barcodes and persistence diagrams and the computation of PH with Vietoris–Rips complexes, alpha complexes, Čech complexes, witness complexes, and cubical complexes for image data. We thus believe that our paper closes a gap in introducing PH to people interested in applications, while our tutorial complements existing tutorials (see, e.g. [3, 30, 86]).

We believe that there is a need for a thorough benchmarking of the state-of-the-art libraries. In our work, we use twelve different data sets to test and compare the libraries javaPlex, Gudhi, DIPHA, Dionysus, and Perseus, and we obtain some surprising results (see Section 2.7.2). There are several benchmarkings in the PH literature; we are aware of the following ones: the benchmarking in [61] compares the implementations of standard and dual algorithms in Dionysus; the one in [162] compares the Morse-theoretic reduction algorithm with the standard algorithm; the one in [16] compares all of the data structures and algorithms implemented in PHAT; the benchmarking in [15] compares PHAT and its spin-off DIPHA; and the benchmarking in C. Maria's doctoral thesis [148] is to our knowledge the only existing benchmarking that compares packages from different

authors. However, Maria compares only up to three different implementations at one time, and he used the package JPLEX (which is no longer maintained) instead of the JAVAPLEX library (its successor). Additionally, the widely used library PERSEUS (e.g., it was used in [126, 127, 189, 194]) does not appear in Maria's benchmarking.

2.3 Homology

Assume that one is given data that lies in a metric space, such as a subset of Euclidean space with an inherited distance function. In many situations, one is not interested in the precise geometry of these spaces, but instead seeks to understand some basic characteristics, such as the number of components or the existence of holes and voids. Algebraic topology captures these basic characteristics either by counting them or by associating vector spaces or more sophisticated algebraic structures to them. Here we are interested in **homology**, which associates one vector space $H_i(X)$ to a space X for each natural number $i \in \{0, 1, 2, \dots\}$. The dimension of $H_0(X)$ counts the number of path components in X, the dimension of $H_1(X)$ is a count of the number of holes, and the dimension of $H_2(X)$ is a count of the number of voids. An important property of these algebraic structures is that they are robust, as they do not change when the underlying space is transformed by bending, stretching, or other deformations. In technical terms, they are **homotopy invariant**.[2]

It can be very difficult to compute the homology of arbitrary topological spaces. We thus approximate our spaces by combinatorial structures called "simplicial complexes," for which homology can be easily computed algorithmically. Indeed, often one is not even given the space X, but instead possesses only a discrete sample set S from which to build a simplicial complex following one of the recipes described in Sections 2.3.2 and 2.5.2.

2.3.1 Simplicial complexes and their homology

We begin by giving the definitions of simplicial complexes and of the maps between them. Roughly, a simplicial complex is a space that is built from a union of points, edges, triangles, tetrahedra, and higher-dimensional polytopes. We illustrate the main definitions given in this section with the example in Fig. 2.3. As we pointed out in Section 2.1, "cubical complexes" give another way to associate a combinatorial structure to a topological space. In TDA, cubical complexes have been used primarily to study image data sets. One can compute PH for a nested sequence of cubical complexes in a similar

[2] Conversely, under favorable conditions (see [103, Corollary 4.33]), these algebraic invariants determine the topology of a space up to homotopy— an equivalence relation that is much coarser (and easier to work with) than the more familiar notion of homeomorphy.

way as for simplicial complexes, but the theory of PH for simplicial complexes is richer, and we therefore examine only simplicial homology and complexes in our discussions. See [113] for a treatment of cubical complexes and their homology.

Definition 1. A **simplicial complex**[3] is a collection K of non-empty subsets of a set K_0 such that $\tau \subset \sigma$ and $\sigma \in K$ guarantees that $\tau \in K$ and $\{v\} \in K$ for all $v \in K_0$. The elements of K_0 are called **vertices** of K, and the elements of K are called **simplices**. Additionally, we say that a simplex has **dimension p** or is a **p-simplex** if it has a cardinality of $p+1$. We use K_p to denote the collection of p-simplices. The **k-skeleton** of K is the union of the sets K_p for all $p \in \{0, 1, \dots, k\}$. If τ and σ are simplices such that $\tau \subset \sigma$, then we call τ a *face* of σ. The **dimension** of K is defined as the maximum of the dimensions of its simplices. A **map of simplicial complexes**, $f : K \to L$, is a map $f \colon K_0 \to L_0$ such that $f(\sigma) \in L$ for all $\sigma \in K$.

We give an example of a simplicial complex in Fig. 2.3(a) and an example of a map of simplicial complexes in Fig. 2.3(b). Definition 1 is rather abstract, but one can always interpret a finite simplicial complex K geometrically as a subset of $\mathbb{R}^N$ for sufficiently large N; such a subset is called a "geometric realisation," and it is unique up to a canonical piecewise-linear homeomorphism. For example, the simplicial complex in Fig. 2.3(a) has a geometric realisation given by the subset of $\mathbb{R}^2$ in Fig. 2.3(c).

We now define homology for simplicial complexes. Let $\mathbb{F}_2$ denote the field with two elements. Given a simplicial complex K, let $C_p(K)$ denote the $\mathbb{F}_2$-vector space with basis given by the p-simplices of K. For any $p \in \{1, 2, \dots\}$, we define the linear map

$$d_p \colon C_p(K) \to C_{p-1}(K) \colon \sigma \mapsto \sum_{\tau \subset \sigma, \tau \in K_{p-1}} \tau \,.$$

For $p = 0$, we define d_0 to be the zero map. In words, d_p maps each p-simplex to its boundary, the sum of its faces of codimension 1. Because the boundary of a boundary is always empty, the linear maps d_p have the property that composing any two consecutive maps yields the zero map: for all $p \in \{0, 1, 2, \dots\}$, we have $d_p \circ d_{p+1} = 0$. Consequently, the image of d_{p+1} is contained in the kernel of d_p, so we can take the quotient of $\ker(d_p)$ by $\operatorname{image}(d_{p+1})$. We can thus make the following definition.

Definition 2. For any $p \in \{0, 1, 2, \dots\}$, the **pth homology** of a simplicial complex K is the quotient vector space

$$H_p(K) := \ker(d_p) \,/\, \operatorname{image}(d_{p+1}) \,.$$

[3]Note that this is usually called an"abstract simplicial complex" in the literature.

(a) A simplicial complex:

$$K = \{\{a\},\{b\},\{c\},\{d\},\{e\},\{a,b\},\{a,c\},\{a,d\},\{b,c\},\{c,d\},\{a,b,c\}\}\,.$$

(b) Let K be the simplicial complex in (a), and let K' be the following simplicial complex:

$$K' = \{\{g\},\{h\},\{l\},\{g,h\},\{g,l\},\{h,l\},\{g,h,l\}\}\,.$$

We define a map of simplicial complexes, $f\colon K \to K'$, as follows:

$$f\colon K_0 \to K'_0\colon x \mapsto \begin{cases} h\,, & \text{if } x{=}\text{a} \\ g\,, & \text{otherwise}\,. \end{cases}$$

(c) Geometric realisations of the simplicial complexes in (a) and (b) are the following subsets of $\mathbb{R}^2$:

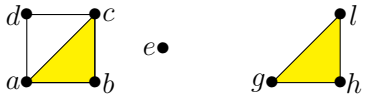


Figure 2.3: A simple example. (a) Example of a simplicial complex, (b) a map of simplicial complexes, and (c) a geometric realisation of the simplicial complex in (a).

Its dimension

$$\beta_p(K) := \dim H_p(K) = \dim\ker(d_p) - \dim\operatorname{image}(d_{p+1})$$

is called the ***p*th Betti number** of K. Elements in the image of d_{p+1} are called ***p*-boundaries**, and elements in the kernel of d_p are called ***p*-cycles**.

Intuitively, the p-cycles that are not boundaries represent p-dimensional holes. Therefore, the pth Betti number "counts" the number of p-holes. Additionally, if K is a simplicial complex of dimension n, then for all $p > n$, we have that $H_p(K) = 0$, as K_p is empty and hence $C_p(K) = 0$. We therefore obtain the following sequence of vector spaces and linear maps:

$$0 \xrightarrow{d_{n+1}} C_n(K) \xrightarrow{d_n} \ldots\ldots \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \xrightarrow{d_0} 0\,.$$

We give an example of such a sequence in Fig. 2.4(a), for which we also report the Betti numbers.

One of the most important properties of simplicial homology is "functoriality." Any map $f\colon K \to K'$ of simplicial complexes induces the following $\mathbb{F}_2$-linear map:

$$\widetilde{f}_p\colon C_p(K) \to C_p(K')\colon \sum_{\sigma\in K_p} c_\sigma\sigma \mapsto \sum_{\sigma\in K_p \text{ such that } f(\sigma)\in K'_p} c_\sigma f(\sigma)\,, \quad \text{for any} \quad p \in \{0,1,2,\ldots\}\,,$$

(a) We compute the simplicial homology for the simplicial complex in Fig. 2.3(a). We have the following sequence of vector spaces and linear maps:

$$0 \longrightarrow \mathbb{F}_2 \xrightarrow{d_2} \mathbb{F}_2^5 \xrightarrow{d_1} \mathbb{F}_2^5 \xrightarrow{d_0} 0\,.$$

Let abc denote the basis vector that corresponds to the simplex $\{a, b, c\}$. Similarly, we use ab, ac, ad, bc, and cd to denote the basis vectors that correspond to the 1-simplices; and we use a, b, c, d, and e to denote the basis vectors that correspond to the 0-simplices. We order the bases of the vector spaces using lexicographic order. We then have

$$d_2 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \end{pmatrix}^{\mathrm{t}}$$

and

$$d_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

One can then compute that $\beta_0(K) = 2$, $\beta_1(K) = 1$, and all higher Betti numbers are 0.

(b) The map of simplicial complexes, $f\colon K \to K'$, in Fig. 2.3(b) induces the map $\widetilde{f}_0\colon C_0(K) \to C_0(K')$ on chain complexes that sends the basis element a to h (note that we use the same notation as in part (a) of this box). Furthermore, it sends the basis elements that correspond to the other 0-simplices of K to g, and it induces the map $\widetilde{f}_1\colon C_1(K) \to C_1(K')$ that sends the basis elements ab, ac, and ad to cd and sends the basis elements cd and bc to 0. This gives a map f_0 on degree-0 homology that sends both generators of $H_0(K)$ to the generator of $H_0(K')$. It follows that $H_i(K') = 0$ for all $i \geq 1$, so f_i is the zero map for $i \geq 1$.

Figure 2.4: Examples to illustrate simplicial homology. (a) Computation of simplicial homology for the simplicial complex in Fig. 2.3(a) and (b) induced map in degree-0 homology for the map of simplicial complexes in Fig. 2.3(b).

where $c_\sigma \in \mathbb{F}_2$. Additionally, $\widetilde{f}_p \circ d_{p+1} = d'_{p+1} \circ \widetilde{f}_{p+1}$, and the map $\widetilde{f}_p$ therefore induces the following linear map between homology vector spaces:

$$\begin{aligned} f_p\colon H_p(K) &\to H_p(K')\,, \\ [c] &\mapsto [\widetilde{f}_p(c)]\,. \end{aligned}$$

(We give an example of such a map in Fig. 2.4(b).) Consequently, to any map $f\colon K \to K'$ of simplicial complexes, we can assign a map $f_p\colon H_p(K) \to H_p(K')$ for any $p \in$

$\{0, 1, 2, \dots\}$. This assignment has the important property that given a pair of composable maps of simplicial complexes, $f\colon K \longrightarrow K'$ and $g\colon K' \longrightarrow K''$, the map $(g \circ f)_p\colon H_p(K) \to H_p(K'')$ is equal to the composition of the maps induced by f and g. That is, $(g \circ f)_p = g_p \circ f_p$. The fact that a map of simplicial complexes induces a map on homology that is compatible with composition is called **functoriality**, and it is crucial for the definition of persistent homology (see Section 2.4.1).

When working with simplicial complexes, one can modify a simplicial complex by removing or adding a pair of simplices (σ, τ), where τ is a codimension-1 face of σ and σ is the only simplex that has τ as a face. The resulting simplicial complex has the same homology as the one with which we started. In Fig. 2.3 (a), we can remove the pair $(\{a, b, c\}, \{b, c\})$ and then the pair $(\{a, b\}, \{b\})$ without changing the Betti numbers. Such a move is called an **elementary simplicial collapse** [47]. In Section 2.5.2.6, we will see an application of this for the computation of PH.

In this section, we have defined simplicial homology over the field $\mathbb{F}_2$ — i.e., "with coefficients in $\mathbb{F}_2$." One can be more general and instead define simplicial homology with coefficients in any field (or even in the integers). However, when $1 \neq -1$, one needs to take more care when defining the boundary maps d_p to ensure that $d_p \circ d_{p+1}$ remains the zero map. Consequently, the definition is more involved. For the purposes of the present paper, it suffices to consider homology with coefficients in the field $\mathbb{F}_2$. Indeed, we will see in Section 4.9 that to obtain topological summaries in the form of barcodes, we need to compute homology with coefficients in a field. Furthermore, as we summarise in Table 2.2 (in Section 2.7), most of the implementations for the computation of PH work with $\mathbb{F}_2$.

We conclude this section with a warning: changing the coefficient field can affect the Betti numbers. For example, if one computes the homology of the Klein bottle (see Section 2.7.1.1) with coefficients in the field $\mathbb{F}_p$ with p elements, where p is a prime, then $\beta_0(K) = 1$ for all primes p. However, $\beta_1(K) = 2$ and $\beta_2(K) = 1$ if $p = 2$, but $\beta_1(K) = 1$ and $\beta_2(K) = 0$ for all other primes p. The fact that $\beta_2(K) = 0$ for $p \neq 2$ arises from the nonorientability of the Klein bottle. The treatment of different coefficient fields is beyond the scope of our survey, but interested readers can peruse [103] for an introduction to homology and [113] for an overview of computational homology.

2.3.2 Building simplicial complexes

As we discussed in Section 2.3.1, computing the homology of finite simplicial complexes boils down to linear algebra. The same is not true for the homology of an arbitrary space

X, and one therefore tries to find simplicial complexes whose homology approximates the homology of the space in an appropriate sense.

An important tool is the Čech (Č) complex. Let $\mathcal{U}$ be a cover of X — i.e., a collection of subsets of X such that the union of the subsets is X. The k-simplices of the Čech complex are the non-empty intersections of $k+1$ sets in the cover $\mathcal{U}$. More precisely, we define the nerve of a collection of sets as follows.

Definition 3. Let $\mathcal{U} = \{U_i\}_{i \in I}$ be a non-empty collection of sets. The **nerve** of $\mathcal{U}$ is the simplicial complex with set of vertices given by I and k-simplices given by $\{i_0, \ldots, i_k\}$ if and only if $U_{i_0} \cap \cdots \cap U_{i_k} \neq \emptyset$.

If the cover of the sets is sufficiently "nice," then the Nerve Theorem implies that the nerve of the cover and the space X have the same homology [22, 74]. For example, suppose that we have a finite set of points S in a metric space X. We then can define, for every $\epsilon > 0$, the space S_ϵ as the union $\cup_{x \in S} B(x, \epsilon)$, where $B(x, \epsilon)$ denotes the closed ball with radius ϵ centered at x. It follows that $\{B(x, \epsilon) \mid x \in S\}$ is a cover of S_ϵ, and the nerve of this cover is the **Čech complex on S at scale ϵ**. We denote this complex by $\check{C}_\epsilon(S)$. If the space X is Euclidean space, then the Nerve Theorem guarantees that the simplicial complex $\check{C}_\epsilon(S)$ recovers the homology of S_ϵ.

From a computational point of view, the Čech complex is expensive because one has to check for large numbers of intersections. Additionally, in the worst case, the Čech complex can have dimension $|\mathcal{U}| - 1$, and it therefore can have many simplices in dimensions higher than the dimension of the underlying space. Ideally, it is desirable to construct simplicial complexes that approximate the homology of a space but are easy to compute and have "few" simplices, especially in high dimensions. This is a subject of ongoing research: In Subsection 2.5.2, we give an overview of state-of-the-art methods to associate complexes to point-cloud data in a way that addresses one or both of these desiderata. See [74, 80] for more details on the Čech complex, and see [22, 74] for a precise statement of the Nerve Theorem.

2.4 Persistent homology

Assume that we are given experimental data in the form of a finite metric space S; there are points or vectors that represent measurements along with some distance function (e.g., given by a correlation or a measure of dissimilarity) on the set of points or vectors. Whether or not the set S is a sample from some underlying topological space, it is useful to think of it in those terms. Our goal is to recover the properties of such an underlying space in a way that is robust to small perturbations in the data S. In a broad sense, this

is the subject of **topological inference**. (See [166] for an overview.) If S is a subset of Euclidean space, one can consider a "thickening" S_ϵ of S given by the union of balls of a certain fixed radius ϵ around its points and then compute the Čech complex. One can thus try to compute qualitative features of the data set S by constructing the Čech complex for a chosen value ϵ and then computing its simplicial homology. The problem with this approach is that there is a priori no clear choice for the value of the parameter ϵ. The key insight of PH is the following: To extract qualitative information from data, one considers several (or even all) possible values of the parameter ϵ. As the value of ϵ increases, simplices are added to the complexes. Persistent homology then captures how the homology of the complexes changes as the parameter value increases, and it detects which features "persist" across changes in the parameter value. We give an example of persistent homology in Fig. 2.5.

2.4.1 Filtered complexes and homology

Let K be a finite simplicial complex, and let $K_1 \subset K_2 \subset \cdots \subset K_l = K$ be a finite sequence of nested subcomplexes of K. The simplicial complex K with such a sequence of subcomplexes is called a **filtered simplicial complex**. See Fig. 2.5(a) for an example of filtered simplicial complex. We can apply homology to each of the subcomplexes. For all homology degrees p, the inclusion maps $K_i \to K_j$ induce $\mathbb{F}_2$-linear maps $f_{i,j}\colon H_p(K_i) \to H_p(K_j)$ for all $i, j \in \{1, \ldots, l\}$ with $i \leq j$. By functoriality (see Section 2.3.1), it follows that

$$f_{k,j} \circ f_{i,k} = f_{i,j} \qquad \text{for all } i \leq k \leq j\,. \tag{2.1}$$

We therefore give the following definition[4].

Definition 4. Let $K_1 \subset K_2 \subset \cdots \subset K_l = K$ be a filtered simplicial complex. The ***p*th persistent homology of *K*** is the pair

$$\Big(\{H_p(K_i)\}_{1\leq i\leq l}\,, \{f_{i,j}\}_{1\leq i\leq j\leq l}\Big)\,,$$

where for all $i, j \in \{1, \ldots, l\}$ with $i \leq j$, the linear maps $f_{i,j}\colon H_p(K_i) \to H_p(K_j)$ are the maps induced by the inclusion maps $K_i \to K_j$.

[4] A pair $(\{M_i\}_{i\in I}, \{\phi_{i,j}\colon M_i \to M_j\}_{i\leq j})$, where $(I, \leq)$ is a totally ordered set, such that for each i, we have that M_i is a vector space and the maps $\phi_{i,j}$ are linear maps satisfying the functoriality property (2.1), is usually called a **persistence module**. With this terminology, the homology of a filtered simplicial complex is an example of persistence module.

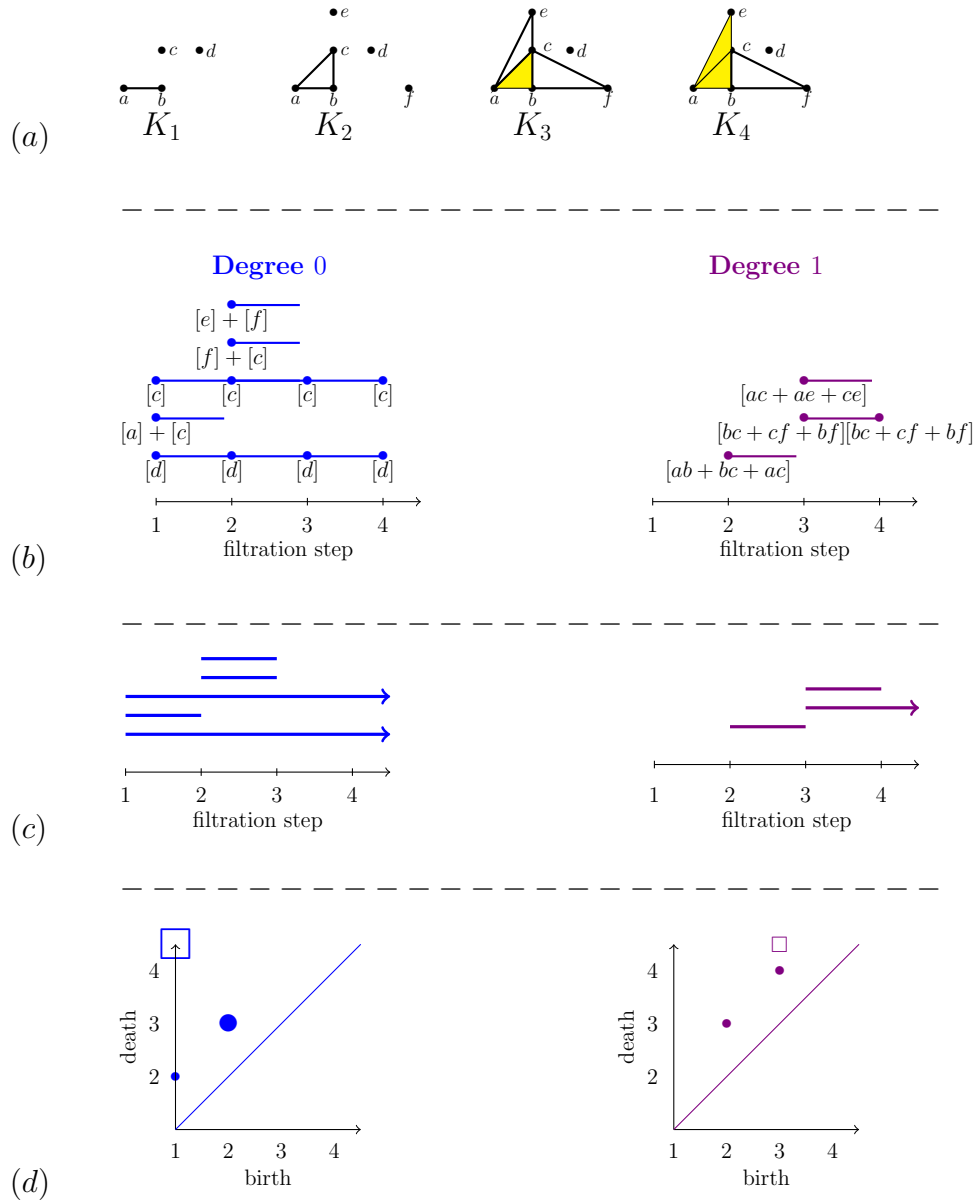


Figure 2.5: Example of persistent homology for a finite filtered simplicial complex. (a) We start with a finite filtered simplicial complex. (b) At each filtration step i, we draw as many vertices as the dimension of (left column) $H_0(K_i)$ and (right column) $H_1(K_i)$. We label the vertices by basis elements, the existence of which is guaranteed by the Fundamental Theorem of Persistent Homology, and we draw an edge between two vertices to represent the maps $f_{i,j}$, as explained in the main text. We thus obtain a well-defined collection of disjoint half-open intervals called a "barcode." We interpret each interval in degree p as representing the lifetime of a p-homology class across the filtration. (c) We rewrite the diagram in (b) in the conventional way. We represent classes that are born but do not die at the final filtration step using arrows that start at the birth of that feature and point to the right. (d) An alternative graphical way to represent barcodes (which gives exactly the same information) is to use **persistence diagrams**, in which an interval $[i, j)$ is represented by the point (i, j) in the extended plane $\overline{\mathbb{R}}^2$, where $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. Therefore, a persistence diagram is a finite multiset of points in $\overline{\mathbb{R}}^2$. We use squares to signify the classes that do not die at the final step of a filtration, and the size of dots or squares is directly proportional to the number of points being represented. For technical reasons, which we discuss briefly in Section 2.5.4, one also adds points on the diagonal to the persistence diagrams. (Each of the points on the diagonal has infinite multiplicity.)

The pth persistent homology of a filtered simplicial complex gives more refined information than just the homology of the single subcomplexes. We can visualize the information given by the vector spaces $H_p(K_i)$ together with the linear maps $f_{i,j}$ by drawing the following diagram: at filtration step i, we draw as many bullets as the dimension of the vector space $H_p(K_i)$. We then connect the bullets as follows: we draw an interval between bullet u at filtration step i and bullet v at filtration step $i+1$ if the generator of $H_p(K_i)$ that corresponds to u is sent to the generator of $H_p(K_{i+1})$ that corresponds to v. If the generator corresponding to a bullet u at filtration step i is sent to 0 by $f_{i,i+1}$, we draw an interval starting at u and ending at $i+1$. (See Fig. 2.5(b) for an example.) Such a diagram clearly depends on a choice of basis for the vector spaces $H_p(K_i)$, and a poor choice can lead to complicated and unreadable clutter. Fortunately, by the Fundamental Theorem of Persistent Homology [224], there is a choice of basis vectors of $H_p(K_i)$ for each $i \in \{1, \ldots, l\}$ such that one can construct the diagram as a well-defined and unique collection of disjoint half-open intervals, collectively called a **barcode**[5]. We give an example of a barcode in Fig. 2.5(c). Note that the Fundamental Theorem of PH, and hence the existence of a barcode, relies on the fact that we are using homology with field coefficients. (See [224] for more details.)

There is a useful interpretation of barcodes in terms of births and deaths of generators. Considering the maps $f_{i,j}$ written in the basis given by the Fundamental Theorem of Persistent Homology, we say that $x \in H_p(K_i)$ (with $x \neq 0$) is **born** in $H_p(K_i)$ if it is not in the image of $f_{i-1,i}$ (i.e., $f_{i-1,i}^{-1}(x) = \emptyset$). For $x \in H_p(K_i)$ (with $x \neq 0$), we say that x **dies** in $H_p(K_j)$ if $j > i$ is the smallest index for which $f_{i,j}(x) = 0$. The lifetime of x is represented by the half-open interval $[i, j)$. If $f_{i,j}(x) \neq 0$ for all $j > i$ in I, we say that x **lives forever**, and its lifetime is represented by the interval $[i, \infty)$.

Remark 5. Note that some references (e.g., [74]) introduce persistent homology by defining the birth and death of generators without using the existence of a choice of compatible bases, as given by the Fundamental Theorem of Persistent Homology. The definition of birth coincides with the definition that we have given, but the definition of death is different. One says that $x \in H_p(K_i)$ (with $x \neq 0$) **dies** in $H_p(K_j)$ if $j > i$ is the smallest index for which either $f_{i,j}(x) = 0$ or there exists $y \in H_p(K_i)$ with $0 \neq y$ and $\min_{i'}\{f_{i',i}^{-1}(y) \neq \emptyset\} < \min_{i'}\{f_{i',i}^{-1}(x) \neq \emptyset\}$ such that $f_{i,j}(y) = f_{i,j}(x)$. In words, this means that x and y merge at filtration step j, and the class that was born earlier is the one that survives. In the literature, this is called the **elder rule**. We do not adopt this definition,

[5]Although the collection of intervals is unique, note that one has to choose a vertical order when drawing the intervals in the diagram, and there is therefore an ambiguity in the representation of the intervals as a barcode. However, there is no ambiguity when representing the intervals as points in a persistence diagram (see Fig. 2.5(d)).

because the elder rule is not well-defined when two classes are born at the same time, as there is no way to choose which class will survive. For example, in Fig. 2.5, there are two classes in H_0 that are born at the same stage in K_1. These two classes merge in K_2, but neither dies. The class that dies is $[a] + [c]$.

There are numerous excellent introductions to PH, such as the books [74, 93, 166, 220] and the papers [34, 72, 77, 92, 207]. For a brief and friendly introduction to PH and some of its applications, see the video `https://www.youtube.com/watch?v=h0bnG1Wavag`. For a brief introduction to some of the ideas in TDA, see the video `https://www.youtube.com/watch?v=XfWibrh6stw`.

2.5 Computation of PH for data

We summarise the pipeline for the computation of PH from data in Fig. 2.6. In the following subsections, we describe each step of this pipeline and state-of-the-art algorithms for the computation of PH. The two features that make PH appealing for applications are that it is computable via linear algebra and that it is stable with respect to perturbations in the measurement of data. In Section 2.5.5, we give a brief overview of stability results.

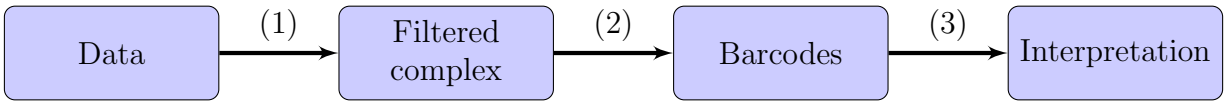


Figure 2.6: PH pipeline.

2.5.1 Data

As we mentioned in Section 2.1, types of data sets that one can study with PH include finite metric spaces, digital images, and networks. We now give a brief overview of how one can study these types of data sets using PH.

2.5.1.1 Networks

One can construe an undirected network as a 1-dimensional simplicial complex. If the network is weighted, then filtering by increasing or decreasing weight yields a filtered 1-dimensional simplicial complex. To obtain more refined information about the network, it is desirable to construct higher-dimensional simplices. There are various methods to do this. The simplest method, called a **weight rank clique filtration** (WRCF), consists of building a clique complex on each subnetwork. (See Section 2.5.2.1 for the definition

of "clique complex.") See [171] for an application of this method. Another method to study networks with PH consists of mapping the nodes of the network to points of a finite metric space. There are several ways to compute distances between nodes of a network; the method that we use in our benchmarking in Section 2.7 consists of computing a shortest path between nodes. For such a distance to be well-defined, note that one needs the network to be connected (although conventionally one takes the distance between nodes in different components to be infinity). There are many methods to associate an unfiltered simplicial complex to both undirected and directed networks. See the book [108] for an overview of such methods, and see the paper [107] for an overview of PH for networks.

2.5.1.2 Digital images

As we mentioned in Section 2.1, digital images have a natural cubical structure: 2-dimensional digital images are made of pixels, and 3-dimensional images are made of voxels. Therefore, to study digital images, cubical complexes are more appropriate than simplicial complexes. Roughly, cubical complexes are spaces built from a union of vertices, edges, squares, cubes, and so on. One can compute PH for cubical complexes in a similar way as for simplicial complexes, and we will therefore not discuss this further in this paper. See [113] for a treatment of computational homology with cubical complexes rather than simplicial complexes and for a discussion of the relationship between simplicial and cubical homology. See [202] for an efficient algorithm and data structure for the computation of PH for cubical data, and [17] for an algorithm that computes PH for cubical data in an approximate way. For an application of PH and cubical complexes to movies, see [162].

Other approaches for studying digital images are also useful. In general, given a digital image that consists of N pixels or voxels, one can consider this image as a point in a $c \times N$-dimensional space, with each coordinate storing a vector of length c representing the color of a pixel or voxel. Defining an appropriate distance function on such a space allows one to consider a collection of images (each of which has N pixels or voxels) as a finite metric space. A version of this approach was used in [36], in which the local structure of natural images was studied by selecting 3×3 patches of pixels of the images.

2.5.1.3 Finite metric spaces

As we mentioned in the previous two subsections, both undirected networks and image data can be construed as finite metric spaces. Therefore, methods to study finite metric spaces with PH apply to the study of networks and image data sets.

In some applications, points of a metric space have associated "weights." For instance, in the study of molecules, one can represent a molecule as a union of balls in Euclidean space [216,218]. For such data sets, one would therefore also consider a minimum filtration value (see Section 2.5.2 for the description of such filtration values) at which the point enters the filtration. In Table 2.2(g), we indicate which software libraries implement this feature.

2.5.2 Filtered Simplicial Complexes

In Section 2.3.2, we introduced the Čech complex, a classical simplicial complex from algebraic topology. However, there are many other simplicial complexes that are better suited for studying data from applications. We discuss them in this section.

To be a useful tool for the study of data, a simplicial complex has to satisfy some theoretical properties dictated by topological inference; roughly, if we build the simplicial complex on a set of points sampled from a space, then the homology of the simplicial complex has to approximate the homology of the space. For the Čech complex, these properties are guaranteed by the Nerve Theorem. Some of the complexes that we discuss in this subsection are motivated by a "sparsification paradigm": they approximate the PH of known simplicial complexes but have fewer simplices than them. Others, like the Vietoris–Rips complex, are appealing because they can be computed efficiently. In this subsection, we also review reduction techniques, which are heuristics that reduce the size of complexes without changing the PH. In Table 2.1, we summarise the simplicial complexes that we discuss in this subsection.

For the rest of this subsection (X, d) denotes a metric space, and S is a subset of X, which becomes a metric space with the induced metric. In applications, S is the collection of measurements together with a notion of distance, and we assume that S lies in the (unknown) metric space X. Our goal is then to compute persistent homology for a sequence of nested spaces $S_{\epsilon_1}, S_{\epsilon_2}, \ldots, S_{\epsilon_l}$, where each space gives a "thickening" of S in X.

2.5.2.1 Vietoris–Rips complex

We have seen that one of the disadvantages of the Čech complex is that one has to check for a large number of intersections. To circumvent this issue, one can instead consider the Vietoris–Rips (VR) complex, which approximates the Čech complex. For a non-negative real number ϵ, the **Vietoris–Rips complex $\mathbf{VR_\epsilon(S)}$ at scale $\boldsymbol{\epsilon}$** is defined as

$$\mathrm{VR}_\epsilon(S) = \{\sigma \subseteq S \mid \mathrm{d}(x, y) \leq 2\epsilon \text{ for all } x, y \in \sigma\} \ .$$

The sense in which the VR complex approximates the Čech complex is that, when S is a subset of Euclidean space, we have $\check{C}_\epsilon(S) \subseteq \mathrm{VR}_\epsilon(S) \subseteq \check{C}_{\sqrt{2}\epsilon}(S)$. Deciding whether a subset $\sigma \subseteq S$ is in $\mathrm{VR}_\epsilon(S)$ is equivalent to deciding if the maximal pairwise distance between any two vertices in σ is at most 2ϵ. Therefore, one can construct the VR complex in two steps. One first computes the **ϵ-neighborhood graph of S**. This is the graph whose vertices are all points in S and whose edges are

$$\{(i,j) \in S \times S \mid i \neq j \text{ and } \mathrm{d}(i,j) \leq 2\epsilon\}\,.$$

Second, one obtains the VR complex by computing the **clique complex** of the ϵ-neighborhood graph. The clique complex of a graph is a simplicial complex that is defined as follows: The subset $\{x_0, \ldots, x_k\}$ is a k-simplex if and only if every pair of vertices in $\{x_0, \ldots, x_k\}$ is connected by an edge. Such a collection of vertices is called a **clique**. This construction makes it very easy to compute the VR complex, because to construct the clique complex one has only to check for pairwise distances — for this reason clique complexes are also called "lazy" in the literature. Unfortunately, the VR complex has the same worst-case complexity as the Čech complex. In the worst case, it can have up to $2^{|S|} - 1$ simplices and dimension $|S| - 1$.

In applications, one therefore usually only computes the VR complex up to some dimension $k \ll |S| - 1$. In our benchmarking, we often choose $k = 2$ and $k = 3$.

The paper [221] overviews different algorithms to perform both of the steps for the construction of the VR complex, and it introduces fast algorithms to construct the clique complex. For more details on the VR complex, see [74, 201]. For a proof of the approximation of the Čech complex by the VR complex, see [74]; see [123] for a generalisation of this result.

2.5.2.2 The Delaunay complex

To avoid the computational problems of the Čech and VR complexes, we need a way to limit the number of simplices in high dimensions. The Delaunay complex gives a geometric tool to accomplish this task, and most of the new simplicial complexes that have been introduced for the study of data are based on variations of the Delaunay complex. The Delaunay complex and its dual, the Voronoi diagram, are central objects of study in computational geometry because they have many useful properties.

For the Delaunay complex, one usually considers $X = \mathbb{R}^d$, so we also make this assumption. We subdivide the space $\mathbb{R}^d$ into regions of points that are closest to any of the points in S. More precisely, for any $s \in S$, we define

$$V_s = \{x \in \mathbb{R}^d \mid \mathrm{d}(x,s) \leq \mathrm{d}(x,s') \text{ for all } s' \in S\}\,.$$

The collection of sets V_s is a cover for $\mathbb{R}^d$ that is called the **Voronoi decomposition of $\mathbb{R}^d$ with respect to S**, and the nerve of this cover is called the **Delaunay complex** of S and is denoted by $\mathrm{Del}(S;\mathbb{R}^d)$. In general, the Delaunay complex does not have a geometric realisation in $\mathbb{R}^d$. However, if the points S are "in general position" [6] then the Delaunay complex has a geometric realisation in $\mathbb{R}^d$ that gives a triangulation of the convex hull of S. In this case, the Delaunay complex is also called the **Delaunay triangulation**.

The complexity of the Delaunay complex depends on the dimension d of the space. For $d \leq 2$, the best algorithms have complexity $\mathcal{O}(N \log N)$, where N is the cardinality of S. For $d \geq 3$, they have complexity $\mathcal{O}(N^{\lceil d/2 \rceil})$. The construction of the Delaunay complex is therefore costly in high dimensions, although there are efficient algorithms for the computation of the Delaunay complex for $d = 2$ and $d = 3$. Developing efficient algorithms for the construction of the Delaunay complex in higher dimensions is a subject of ongoing research. See [24] for a discussion of progress in this direction, and see [98] for more details on the Delaunay complex and the Voronoi diagram.

2.5.2.3 Alpha complex

We continue to assume that S is a finite set of points in $\mathbb{R}^d$. Using the Voronoi decomposition, one can define a simplicial complex that is similar to the Čech complex, but which has the desired property that (if the points S are in general position) its dimension is at most that of the space. Let $\epsilon > 0$, and let S_ϵ denote the union $\bigcup_{s \in S} B(s, \epsilon)$. For every $s \in S$, consider the intersection $V_s \cap B(s, \epsilon)$. The collection of these sets forms a cover of S_ϵ, and the nerve complex of this cover is called the **alpha (α) complex of S at scale ϵ** and is denoted by $A_\epsilon(S)$. The Nerve Theorem applies, and it therefore follows that $A_\epsilon(S)$ has the same homology as S_ϵ.

Furthermore, $A_\infty(S)$ is the Delaunay complex; and for $\epsilon < \infty$, the alpha complex is a subcomplex of the Delaunay complex. The alpha complex was introduced for points in the plane in [75], in 3-dimensional Euclidean space in [79], and for Euclidean spaces of arbitrary dimension in [71]. For points in the plane, there is a well-known speed-up for the alpha complex that uses a duality between 0-dimensional and 1-dimensional persistence for alpha complexes [72]. (See [129] for the algorithm, and see [128] for an implementation.)

[6] A set S of points in $\mathbb{R}^d$ is **in general position** if no $d+2$ points of S lie on a $d-1$-dimensional sphere, and for any $d' < d$, no $d'+2$ points of S lie on a d'-dimensional subspace that is isometric to $\mathbb{R}^{d'}$. In particular, a set of points S in $\mathbb{R}^2$ is in general position if no four points lie on a 1-dimensional sphere and no three points lie on a line.

2.5.2.4 Witness complexes

Witness complexes are very useful for analyzing large data sets, because they make it possible to construct a simplicial complex on a significantly smaller subset $L \subseteq S$ of points that are called "landmark" points. Meanwhile, because one uses information about all points in S to construct the simplicial complex, the points in S are called "witnesses." Witness complexes can be construed as a "weak version" of Delaunay complexes. (See the characterisation of the Delaunay complex in [58].)

Definition 6. Let (S, d) be a metric space, and let $L \subseteq S$ be a finite subset. Suppose that σ is a non-empty subset of L. We then say that $s \in S$ is a **weak witness for $\boldsymbol{\sigma}$ with respect to $\boldsymbol{L}$** if and only if $\mathrm{d}(s, a) \leq \mathrm{d}(s, b)$ for all $a \in \sigma$ and for all $b \in L \setminus \sigma$. The **weak Delaunay complex $\boldsymbol{\mathrm{Del}^w(L; S)}$ of $\boldsymbol{S}$ with respect to $\boldsymbol{L}$** has vertex set given by the points in L, and a subset σ of L is in $\mathrm{Del}^w(L; S)$ if and only if it has a weak witness in S.

To obtain nested complexes, one can extend the definition of witnesses to ϵ-witnesses.

Definition 7. A point $s \in S$ is a **weak $\boldsymbol{\epsilon}$-witness** for σ with respect to L if and only if $\mathrm{d}(s, a) \leq \mathrm{d}(s, b) + \epsilon$ for all $a \in \sigma$ and for all $b \in L \setminus \sigma$.

Now we can define the **weak Delaunay complex $\boldsymbol{\mathrm{Del}^w(L; S, \epsilon)}$ at scale $\boldsymbol{\epsilon}$** to be the simplicial complex with vertex set L, and such that a subset $\sigma \subseteq L$ is in $\mathrm{Del}^w(L; S, \epsilon)$ if and only if it has a weak ϵ-witness in S. By considering different values for the parameter ϵ, we thereby obtain nested simplicial complexes. The weak Delaunay complex is also called the "weak witness complex" or just the "witness complex" in the literature.

There is a modification of the witness complex called the **lazy witness complex**, and denoted by $\mathrm{Del}^w_{\mathrm{lazy}}(L; X, \epsilon)$. It is a clique complex, and it can therefore be computed more efficiently than the witness complex. The lazy witness complex has the same 1-skeleton as $\mathrm{Del}^w(L; X, \epsilon)$, and one adds a simplex σ to $\mathrm{Del}^w_{\mathrm{lazy}}(L; X, \epsilon)$ whenever its edges are in $\mathrm{Del}^w_{\mathrm{lazy}}(L; X, \epsilon)$. Another type of modification of the witness complex yields **parametrised witness complexes**. Let $\nu = 1, 2, \ldots$ and for all $s \in S$ define $m_\nu(s)$ to be the distance to the νth closest landmark point. Furthermore, define $m_0(s) = 0$ for all $s \in S$. Let $\mathrm{W}_\nu(L; S, \epsilon)$ be the simplicial complex whose vertex set is L and such that a 1-simplex $\sigma = \{x_0, x_1\}$ is in $\mathrm{W}_\nu(L; X, \epsilon)$ if and only if there exists s in S for which

$$\max\{\mathrm{d}(x_0, s), \mathrm{d}(x_1, s)\} \leq m_\nu(s) + \epsilon\,.$$

A simplex σ is in $\mathrm{W}_\nu(L; X, \epsilon)$ if and only if all of its edges belong to $\mathrm{W}_\nu(L; X, \epsilon)$. For $\nu = 2$, note that $\mathrm{W}_2(L; X, \epsilon) = \mathrm{Del}^w_{\mathrm{lazy}}(L; X, \epsilon)$. For $\nu = 0$, we have that $\mathrm{W}_0(L; X, \epsilon)$

approximates the VR complex $\mathrm{VR}(L;\epsilon)$. That is,

$$\mathrm{W}_0(L;X,\epsilon) \subseteq \mathrm{VR}(L;2\epsilon) \subseteq \mathrm{W}_0(L;X,2\epsilon)\,.$$

Note that parametrised witness complexes are often called "lazy witness complexes" in the literature, because they are clique complexes.

The weak Delaunay complex was introduced in [58], and parametrised witness complexes were introduced in [59]. Witness complexes can be rather useful for applications. Because their complexity depends on the number of landmark points, one can reduce complexity by computing simplicial complexes using a smaller number of vertices. However, there are theoretical guarantees for the witness complex only when S is the metric space associated to a low-dimensional Euclidean submanifold. It has been shown that witness complexes can be used to recover the topology of curves and surfaces in Euclidean space [8,100], but they can fail to recover topology for submanifolds of Euclidean space of three or more dimensions [25]. Consequently, there have been studies of simplicial complexes that are similar to the witness complexes but with better theoretical guarantees (see Section 2.5.2.5).

2.5.2.5 Additional complexes

Many more complexes have been introduced for the fast computation of PH for large data sets. These include the graph-induced complex [66], which is a simplicial complex constructed on a subsample Q, and has better theoretical guarantees than the witness complex (see [111] for the companion software); an approximation of the VR complex that has a worst-case size that is linear in the number of data points [184]; an approximation of the Čech complex [123] whose worst-case size also scales linearly in the data; and an approximation of the VR complex via simplicial collapses [68]. We do not discuss such complexes in detail, because thus far (at the time of writing) none of them have been implemented in publicly-available libraries for the computation of PH. (See Table 2.2 in Section 2.7 for information about which complexes have been implemented.)

2.5.2.6 Reduction techniques

Thus far, we have discussed techniques to build simplicial complexes with possibly "few" simplices. One can also take an alternative approach to speed up the computation of PH. For example, one can use a heuristic (i.e., a method without theoretical guarantees on the speed-up) to reduce the size of a filtered complex while leaving the PH unchanged.

For simplicial complexes, one such method is based on discrete Morse theory [179], which was adapted to filtrations of simplicial complexes in [157]. The basic idea of the

Table 2.1: We summarise several types of complexes that are used for PH. We indicate the theoretical guarantees and the worst-case sizes of the complexes as functions of the cardinality N of the vertex set. For the witness complexes (see Section 2.5.2.4), L denotes the set of landmark points, while Q denotes the subsample set for the graph-induced complex (see Section 2.5.2.5).

Complex K	Size of K	Theoretical guarantee
Čech	$2^{\mathcal{O}(N)}$	Nerve theorem
Vietoris–Rips (VR)	$2^{\mathcal{O}(N)}$	Approximates Čech complex
Alpha	$N^{\mathcal{O}(\lceil d/2 \rceil)}$ (N points in $\mathbb{R}^d$)	Nerve theorem
Witness	$2^{\mathcal{O}(\lvert L \rvert)}$	For curves and surfaces in Euclidean space
Graph-induced complex	$2^{\mathcal{O}(\lvert Q \rvert)}$	Approximates VR complex
Sparsified Čech	$\mathcal{O}(N)$	Approximates Čech complex
Sparsified VR	$\mathcal{O}(N)$	Approximates VR complex

algorithm developed in [157] is that one can compute a partial matching of the simplices in a filtered simplicial complex so that (i) pairs occur only between simplices that enter the filtration at the same step, (ii) unpaired simplices determine the homology, and (iii) one can remove paired simplices from the filtered complex without altering the total PH. Such deletions are examples of the elementary simplicial collapses that we mentioned in Section 2.3.1. Unfortunately, the problem of finding an optimal partial matching was shown to be NP complete [109], and one thus relies on heuristics to find partial matchings to reduce the size of the complex.

One particular family of elementary collapses, called **strong collapses**, was introduced in [11]. Strong collapses preserve cycles of shortest length in the representative class of a generator of a hole [211]; this feature makes strong collapses useful for finding holes in networks [211]. A distributed version of the algorithm introduced in [211] was introduced in [210] and adapted for the computation of PH in [209].

A method for the reduction of the size of a complex for clique complexes, such as the VR complex, was introduced in [222] and is called the **tidy-set method**. Using maximal cliques, this method extracts a minimal representation of the graph that determines the clique complex. Although the tidy-set method cannot be extended to filtered complexes, it can be used for the computation of zigzag PH (see Section 2.6) [223]. The tidy-set method is a heuristic, because it does not give a guarantee to minimize the size of the output complex.

2.5.3 From a Filtered Simplicial Complex to Barcodes

To compute the PH of a filtered simplicial complex K and obtain a barcode like the one illustrated in Fig. 2.5(c), we need to associate to it a matrix — the so-called **boundary matrix** — that stores information about the faces of every simplex. To do this, we place a total ordering on the simplices of the complex that is compatible with the filtration in the following sense:

- a face of a simplex precedes the simplex;
- a simplex in the ith complex K_i precedes simplices in K_j for $j > i$, which are not in K_i.

Let n denote the total number of simplices in the complex, and let $\sigma_1, \ldots, \sigma_n$ denote the simplices with respect to this ordering. We construct a square matrix δ of dimension $n \times n$ by storing a 1 in $\delta(i, j)$ if the simplex σ_i is a face of simplex σ_j of codimension 1; otherwise, we store a 0 in $\delta(i, j)$.

Once one has constructed the boundary matrix, one has to reduce it using Gaussian elimination.[7] In the following subsections, we discuss several algorithms for reducing the boundary matrix.

2.5.3.1 Standard algorithm

The so-called `standard algorithm` for the computation of PH was introduced for the field $\mathbb{F}_2$ in [76] and for general fields in [224]. For every $j \in \{1, \ldots, n\}$, we define $\text{low}(j)$ to be the largest index value i such that $\delta(i, j)$ is different from 0.[8] If column j only contains 0 entries, then the value of $\text{low}(j)$ is undefined. We say that the boundary matrix is **reduced** if the map low is injective on its domain of definition. In Alg. 1, we illustrate the standard algorithm for reducing the boundary matrix. Because this algorithm operates on columns of the matrix from left to right, it is also sometimes called the "column algorithm." In the worst case, the complexity of the standard algorithm is cubic in the number of simplices.

[7] As we mentioned in Section 4.9, for the reduction of the boundary matrix and thus the computation of PH, it is crucial that one uses simplicial homology with coefficients in a field; see [224] for details.

[8] This map is called "low" in the literature, because one can think of it as indicating the index of the "lowest" row — the one that is nearest to the bottom of the page on which one writes the boundary matrix — that contains a 1 in column j.

Algorithm 1 The standard algorithm for the reduction of the boundary matrix to barcodes.

```
for j = 1 to n do
    while there exists i < j with low(i) =
    low(j) do
        add column i to column j
    end while
end for
```

2.5.3.2 Reading off the intervals

Once the boundary matrix is reduced, one can read off the intervals of the barcode by pairing the simplices in the following way:

- If $\text{low}(j) = i$, then the simplex σ_j is paired with σ_i, and the entrance of σ_i in the filtration causes the birth of a feature that dies with the entrance of σ_j.
- If $\text{low}(j)$ is undefined, then the entrance of the simplex σ_j in the filtration causes the birth of a feature. It there exists k such that $\text{low}(k) = j$, then σ_j is paired with the simplex σ_k, whose entrance in the filtration causes the death of the feature. If no such k exists, then σ_j is unpaired.

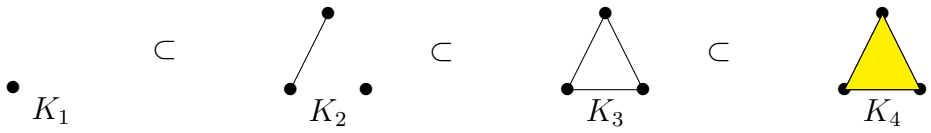
A pair (σ_i, σ_j) gives the half-open interval $[\text{dg}(\sigma_i), \text{dg}(\sigma_j))$ in the barcode, where for a simplex $\sigma \in K$ we define $\text{dg}(\sigma)$ to be the smallest number l such that $\sigma \in K_l$. An unpaired simplex σ_k gives the infinite interval $[\text{dg}(\sigma_k), \infty)$. We give an example of PH computation in Fig. 2.7.

2.5.3.3 Other algorithms

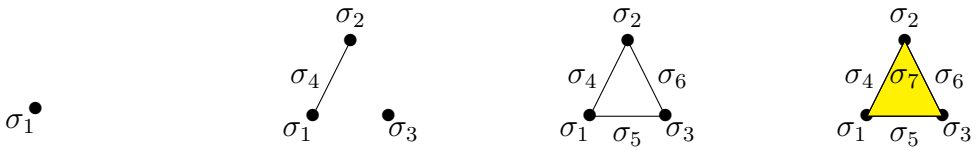
After the introduction of the standard algorithm, several new algorithms were developed. Each of these algorithms gives the same output for the computation of PH, so we only give a brief overview and references to these algorithms, as one does not need to know them to compute PH with one of the publicly-available software packages. In Section 2.7.2, we indicate which implementation of these libraries is best suited to which data set.

As we mentioned in Section 2.5.3.1, in the worst case, the standard algorithm has cubic complexity in the number of simplices. This bound is sharp, as Morozov gave an example of a complex with cubic complexity in [159]. Note that in cases such as when matrices are sparse, complexity is less than cubic. Milosavljević, Morozov, and Skraba [156] introduced an algorithm for the reduction of the boundary matrix in $\mathcal{O}(n^\omega)$, where ω is the matrix-multiplication coefficient (i.e., $\mathcal{O}(n^\omega)$ is the complexity of the multiplication of two square matrices of size n). At present, the best bound for ω is 2.376 [49]. Many other algorithms

(a) A filtered simplicial complex:



(b) We put a total order on the simplices that is compatible with the filtration:



where σ_i denotes the ith simplex in this order.

(c) (Left) The boundary matrix B for the filtered simplicial complex in (a) with respect to order on simplices in (b), and (right) its reduction $\overline{B}$ given by applying Alg. 1 (one first adds column 5 to column 6, and then column 4 to column 6):

$$B = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \overline{B} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

(d) We read off the following intervals from the matrix $\overline{B}$ in (c):

- σ_1 is positive, unpaired; this gives the interval $[1, \infty)$ in H_0.
- σ_2 is positive, paired with σ_4; this gives no interval, because σ_2 and σ_4 enter at the same time in the filtration.
- σ_3 is positive, paired with σ_5: this gives the interval $[2, 3)$ in H_0.
- σ_6 is positive, paired with σ_7: this gives the interval $[3, 4)$ in H_1.

Figure 2.7: Example of PH computation with the `standard algorithm` (see Alg. 1).

have been proposed for the reduction of the boundary matrix. These algorithms give a heuristic speed-up for many data sets and complexes (see the benchmarkings in the forthcoming references), but they still have cubic complexity in the number of simplices. Sequential algorithms include the `twist algorithm` [44] and the `dual algorithm` [61,62]. (Note that the dual algorithm is known to give a speed-up when one computes PH with the VR complex, but not necessarily for other types of complexes (see also the results of our benchmarking for the **vertebra** data set in Appendix B).) Parallel algorithms in a shared setting include the `spectral-sequence algorithm` [74, Section VII.4] and the `chunk algorithm` [14], and parallel algorithms in a distributed setting include the `distributed algorithm` [15]. The `multifield algorithm` is a sequential algorithm that allows the simultaneous computation of PH over several fields [26].

2.5.4 Statistical interpretation of topological summaries

Once one has obtained barcodes, one needs to interpret the results of computations. In applications, one often wants to compare the output of a computation for a certain data set with the output for a null model. Alternatively, one may be studying data sets from the output of a generative model (e.g., many realisations from a model of random networks), and it is then necessary to average results over multiple realisations. In the first instance, one needs both a way to compare the two different outputs and a way to evaluate the significance of the result for the original data set. In the second case, one needs a way to calculate appropriate averages (e.g., summary statistics) of the result of the computations.

From a statistical perspective, one can interpret a barcode as an unknown quantity that one tries to estimate by computing PH. If one wants to use PH in applications, one thus needs a reliable way to apply statistical methods to the output of the computation of PH. To our knowledge, statistical methods for PH were addressed for the first time in the paper [31]. Roughly speaking, there are three current approaches to the problem of statistical analysis of barcodes. In the first approach, researchers study topological properties of random simplicial complexes (see, e.g., [7,51]) and the review papers [6,115]. One can view random simplicial complexes as null models to compare with empirical data when studying PH. In the second approach, one studies properties of a metric space whose points are persistence diagrams. In the third approach, one studies "features" of persistence diagrams. We will provide a bit more detail about the second and third approaches.

In the second approach, one considers an appropriately defined "space of persistence diagrams," defines a distance function on it, studies geometric properties of this space, and does standard statistical calculations (means, medians, statistical tests, and so on).

Recall that a persistence diagram (see Fig. 2.5 for an example) is a multiset of points in $\overline{\mathbb{R}}^2$ and that it gives the same information as a barcode. We now give the following precise definition of a persistence diagram.

Definition 8. A **persistence diagram** is a multiset that is the union of a finite multiset of points in $\overline{\mathbb{R}}^2$ with the multiset of points on the diagonal $\Delta = \{(x, y) \in \mathbb{R}^2 \mid x = y\}$, where each point on the diagonal has infinite multiplicity.

In this definition, we include all of the points on the diagonal in $\mathbb{R}^2$ with infinite multiplicity. We include the diagonal points for technical reasons; roughly, it is desirable to be able to compare persistence diagrams by studying bijections between their points, and persistence diagrams must thus be sets with the same cardinality.

Given two persistence diagrams X and Y, we consider the following general definition of distance between X and Y.

Definition 9. Let $p \in [1, \infty]$. The ***p*th Wasserstein distance** between X and Y is defined as

$$W_p[\mathrm{d}](X, Y) := \inf_{\phi\colon X \to Y} \left[\sum_{x \in X} \mathrm{d}[x, \phi(x)]^p \right]^{1/p}$$

for $p \in [1, \infty)$ and as

$$W_\infty[\mathrm{d}](X, Y) := \inf_{\phi\colon X \to Y} \sup_{x \in X} \mathrm{d}[x, \phi(x)]$$

for $p = \infty$, where d is a metric on $\mathbb{R}^2$ and ϕ ranges over all bijections from X to Y.

Usually, one takes $\mathrm{d} = L_q$ for $q \in [1, \infty]$. One of the most commonly employed distance functions is the **bottleneck distance** $W_\infty[L_\infty]$.

The development of statistical analysis on the space of persistence diagrams is an area of ongoing research, and presently there are few tools that can be used in applications. See [153, 160, 197] for research in this direction. Until recently, the library Dionysus [158] was the only library to implement computation of the bottleneck and Wasserstein distances (for $d = L_\infty$); the library hera [121] implements a new algorithm [122] for the computation of the bottleneck and Wasserstein distances that significantly outperforms the implementation in Dionysus. The library TDA Package [87] (see [86] for the accompanying tutorial) implements the computation of confidence sets for persistence diagrams that was developed in [88], distance functions that are robust to noise and outliers [43], and many more tools for interpreting barcodes.

The third approach for the development of statistical tools for PH consists of mapping the space of persistence diagrams to spaces (e.g., Banach spaces) that are amenable to statistical analysis and machine-learning techniques. Such methods include persistence

landscapes [28], using the space of algebraic functions [4], persistence images [2], and kernelisation techniques [23, 130, 175, 219]. See the papers [125, 192] for applications of persistence landscapes. The package Persistence Landscape Toolbox [69] (see [30] for the accompanying tutorial) implements the computation of persistence landscapes, as well as many statistical tools that one can apply to persistence landscapes, such as mean, ANOVA, hypothesis tests, and many more.

2.5.5 Stability

As we mentioned in Section 2.1, PH is useful for applications because it is stable with respect to small perturbations in the input data.

The first stability theorem for PH, proven in [48], asserts that, under favorable conditions, step (2) in the pipeline in Fig. 2.6 is 1-Lipschitz with respect to suitable distance functions on filtered complexes and the bottleneck distance for barcodes (see Section 2.5.4). This result was generalised in the papers [29, 32, 42]. Stability for PH is an active area of research; for an overview of stability results, their history and recent developments, see [166, Chapter 3].

2.6 Excursus: Generalised persistence

One can use the algorithms that we described in Section 2.5 to compute PH when one has a sequence of complexes with inclusion maps that are all going in the same direction, as in the following diagram:

$$\cdots \to K_{i-1} \to K_i \to K_{i+1} \to \dots .$$

An algorithm, called the `zigzag algorithm`, for the computation of PH for inclusion maps that do not all go in the same direction, as, e.g., in the diagram

$$\cdots \to K_{i-1} \to K_i \leftarrow K_{i+1} \to \dots$$

was introduced in [35]. In the more general setting in which maps are not inclusions, one can still compute PH using the `simplicial map algorithm` [67].

One may also wish to vary two or more parameters instead of one. This yields multi-

filtered simplicial complexes, as, e.g., in the following diagram:

$$\begin{array}{ccccccccc}
 & & \vdots & & \vdots & & \vdots & & \\
 & & \uparrow & & \uparrow & & \uparrow & & \\
\dots & \to & K_{j+1,i-1} & \to & K_{j+1,i} & \to & K_{j+1,i+1} & \to & \dots \\
 & & \uparrow & & \uparrow & & \uparrow & & \\
\dots & \to & K_{j,i-1} & \to & K_{j,i} & \to & K_{j,i+1} & \to & \dots \,. \\
 & & \uparrow & & \uparrow & & \uparrow & & \\
\dots & \to & K_{j-1,i-1} & \to & K_{j-1,i} & \to & K_{j-1,i+1} & \to & \dots \\
 & & \uparrow & & \uparrow & & \uparrow & & \\
 & & \vdots & & \vdots & & \vdots & &
\end{array}$$

In this case, one speaks of **multiparameter persistent homology**. Unfortunately, the Fundamental Theorem of Persistent Homology is no longer valid if one filters with more than one parameter, and there is no such thing as a "generalised interval." The topic of multi-parameter persistence is under active research, and several approaches are being studied to extract topological information from multi-filtered simplicial complexes. See [38,166] for the theory of multi-parameter persistent homology, and see [141] (and [140] for its companion paper) for upcoming software for the visualisation of 2-parameter persistent homology.

2.7 Software

There are several publicly-available implementations for the computation of PH. We give an overview of the libraries with accompanying peer-reviewed publication and summarise their properties in Table 2.2.

The software package javaPlex [193], which was developed by the computational topology group at Stanford University, is based on the Plex library [170], which to our knowledge is the first piece of software to implement the computation of PH. Perseus [161] was developed to implement Morse-theoretic reductions [157] (see Section 2.5.2.6). jHoles [21] is a Java library for computing the weight rank clique filtration for weighted undirected networks [171]. Dionysus [158] is the first software package to implement the `dual algorithm` [61,62]. PHAT [16] is a library that implements several algorithms and data structures for the fast computation of barcodes, takes a boundary matrix as input, and is the first software to implement a matrix-reduction algorithm that can be executed in parallel. DIPHA [13], a spin-off of PHAT, implements a distributed computation of the matrix-reduction algorithm. Gudhi [149] implements new data structures for simplicial complexes and the boundary matrix. It also implements the `multi-field algorithm`, which allows simultaneous computation of PH over several fields [26]. This

library is currently under intense development, and a python interface was just released in the most recent version of the library (namely, Version 2.0.0, whereas the version that we study in our tests is Version 1.3.1). The library RIPSER [12], the most recently developed software of the set that we examine, uses several optimisations and shortcuts to speed up the computation of PH with the VR complex. This library does not have an accompanying peer-reviewed publication. However, because it is currently the best-performing (both in terms of memory usage and in terms of wall-time seconds[9]) library for the computation of PH with the VR complex, we include it in our study. The library SIMPPERS [112] implements the `simplicial map algorithm`. Libraries that implement techniques for the statistical interpretation of barcodes include the TDA PACKAGE [87] and the PERSISTENCE LANDSCAPE TOOLBOX [69]. RIVET, a package for visualizing 2-parameter persistent homology, is slated to be released soon [141]. We summarise the properties of the libraries that we mentioned in this paragraph in Table 2.2, and we discuss the performance for a selection of them in Section 2.7.1.3 and in Appendix B. For a list of programs, see `https://github.com/n-otter/PH-roadmap`.

2.7.1 Benchmarking

We benchmark a subset of the currently available open-source libraries with peer-reviewed publication for the computation of PH. To our knowledge, the published open-source libraries are JHOLES, JAVAPLEX, PERSEUS, DIONYSUS, PHAT, DIPHA, SIMPPERS, and GUDHI. To these, we add the library RIPSER, which is currently the best-performing library to compute PH with the VR complex. To study the performance of the packages, we restrict our attention to the algorithms that are implemented by the largest number of libraries. These are the VR complex and the standard and dual algorithms for the reduction of the boundary matrix. PHAT only takes a boundary matrix as input, so it is not possible to conduct a direct comparison of it with the other implementations. However, the fast data structures and algorithms implemented in PHAT are also implemented in its spin-off software DIPHA, which we include in the benchmarking. The software JHOLES computes PH using the WRCF for weighted undirected networks, and SIMPPERS takes a map of simplicial complexes as input, so these two libraries cannot be compared directly to the other libraries. In Appendix B, we report benchmarking of some additional features that are implemented by some of the six libraries (i.e., JAVAPLEX, PERSEUS, DIONYSUS, DIPHA, GUDHI, and RIPSER) that we test. Specifically, we report results for the computation of PH with cubical complexes for image data sets and the computation of PH with witness, alpha, and Čech complexes.

[9] "Wall time" is the amount of elapsed time perceived by a human.

Table 2.2: Overview of existing software for the computation of PH that have an accompanying peer-reviewed publication (and also RIPSER [12], because of its performance). The symbol "–" signifies that the associated feature is not implemented. For each software package, we indicate the following items. (a) The language in which it is implemented. (b) The implemented algorithms for the computation of barcodes from the boundary matrix. (c) The coefficient fields for which PH is computed, where the letter p denotes any prime number in the coefficient field $\mathbb{F}_p$. (d) The type of homology computed. (e) The filtered complexes that are computed, where VR stands for Vietoris–Rips complex, W stands for the weak witness complex, W_ν stands for parametrised witness complexes, WRCF stands for the weight rank clique filtration, α stands for the alpha complex, and Č for the Čech complex. PERSEUS, DIPHA, and GUDHI implement the computation of the lower-star filtration [78] of a weighted cubical complex; one inputs data in the form of a d-dimensional array; the data is then interpreted as a d-dimensional cubical complex, and its lower-star filtration is computed. (See the Tutorial in Appendix A for more details.) Note that DIPHA and GUDHI use the efficient representation of cubical complexes presented in [202], so the size of the cubical complex that is computed by these libraries is smaller than the size of the resulting complex with PERSEUS. (f) The filtered complexes that one can give as input. JAVAPLEX supports the input of a filtered CW complex for the computation of cellular homology [103]; in contrast with simplicial complexes, there do not currently exist algorithms to assign a cell complex to point-cloud data. (g) Additional features implemented by the library. JAVAPLEX supports the computation of some constructions from homological algebra (see [193] for details), and PERSEUS implements the computation of PH with the VR for points with different "birth times" (see Section 2.5.1.3). The library DIONYSUS implements the computation of vineyards [35] and circle-valued functions [62]. Both JAVAPLEX and DIONYSUS support the output of representatives of homology classes for the intervals in a barcode. (i) Whether visualisation of the output is provided.

Software	JAVAPLEX	PERSEUS	JHOLES	DIONYSUS	PHAT	DIPHA	GUDHI	SIMPPERS	RIPSER
(a) Language	Java	C++	Java	C++	C++	C++	C++	C++	C++
(b) Algorithms for PH	standard, dual, zigzag	Morse reductions, standard	standard (uses JAVAPLEX)	standard, dual, zigzag	standard, dual, twist, chunk, spectral seq.	twist, dual, distributed	dual, multifield	simplicial map	twist, dual
(c) Coeff. field	$\mathbb{Q}$, $\mathbb{F}_p$	$\mathbb{F}_2$	$\mathbb{F}_2$	$\mathbb{F}_2$ (standard, zigzag), $\mathbb{F}_p$ (dual)	$\mathbb{F}_2$	$\mathbb{F}_2$	$\mathbb{F}_p$	$\mathbb{F}_2$	$\mathbb{F}_p$
(d) Homology	simplicial, cellular	simplicial, cubical	simplicial	simplicial	simplicial, cubical	simplicial, cubical	simplicial, cubical	simplicial	simplicial
(e) Filtrations computed	VR, W, W_ν	VR, lower star of cubical complex	WRCF	VR, α, **u**C	–	VR, lower star of cubical complex	VR, α, W lower star of cubical complex	–	VR
(f) Filtrations as input	simplicial complex, zigzag, CW	simplicial complex, cubical complex	–	simplicial complex, zigzag	boundary matrix of simpl. complex	boundary matrix of simpl. complex	–	map of simpl. complexes	–
(g) Additional features	Computes some hom. alg. constructions, homology generators	weighted points for VR	–	vineyards, circle-valued functions, homology generators	–	–	–	–	–
(h) Visualisation	barcodes	persistence diagram	–	–	–	persistence diagram	–	–	–

We study the software packages javaPlex, Perseus, Dionysus, DIPHA, Gudhi, and ripser using both synthetic and real-world data from three different perspectives:

1. Performance measured in CPU seconds and wall-time (i.e., elapsed time) seconds.
2. Memory required by the process.
3. Maximum size of simplicial complex allowed by the software.

2.7.1.1 Data sets

In this subsection, we describe the data sets that we use for our benchmarking. We use data sets from a variety of different mathematical and scientific areas and applications. In each case, when possible, we use data sets that have already been studied using PH. Our list of data sets is far from complete; we view this list as an initial step towards building a comprehensive collection of benchmarking data sets for PH.

Data sets (1)–(4) are synthetic: these arise from topology (1), stochastic topology (2), dynamical systems (3), and from an area at the intersection of network theory and fractal geometry (4) and which was first used to study connection patterns of the cerebral cortex (see below for details). Data sets (5)–(12) are from empirical experiments and measurements: they arise from phylogenetics (5)–(6), genomics (9), neuroscience (8), image analysis (7), medical imaging (10), political science (11), and scientometrics (12).

In each case, these data sets are of one of the following three types: point clouds, weighted undirected networks, and grey-scale digital images. To obtain a point cloud from a real-world weighted undirected network, we compute shortest paths using the inverse of the nonzero weights on edges as distances between nodes (except for the US Congress networks; see below). For the synthetic networks, the values assigned to edges are interpreted as distances between nodes, and we therefore use these values to compute shortest paths. We make all processed versions of the data sets that we use in the benchmarking available at `https://github.com/n-otter/PH-roadmap/tree/master/data_sets`. We provide the scripts that we used to produce the synthetic data sets at `https://github.com/n-otter/PH-roadmap/tree/master/matlab/synthetic_data_sets_scripts`.

We now describe all data sets in detail:

(1) Klein bottle. The Klein bottle is a one-sided nonorientable surface (see Fig. 2.8). We linearly sample points from the Klein bottle using its "figure-8" immersion in $\mathbb{R}^3$ and size sample of 400 points. We denote this data set by **Klein**. Note that the image of the immersion of the Klein bottle does not have the same homotopy type

as the original Klein bottle, but they do have the same singular homology[10] with $\mathbb{F}_2$ coefficients. We have $H_0(B) = \mathbb{F}_2$, $H_1(B) = \mathbb{F}_2 \oplus \mathbb{F}_2$, and $H_2(B) = \mathbb{F}_2$, where B denotes the Klein bottle and $H_i(B)$ is the ith singular homology group with $\mathbb{F}_2$ coefficients.

(2) Random VR complexes (uniform distribution) [114]. The parameters for this model are positive integers N and d; the random VR complex for parameters N and d is the VR complex $\mathrm{VR}_\epsilon(X)$, where X is a set of N points sampled from $\mathbb{R}^d$. (Equivalently, the random VR complex is the clique complex on the random geometric graph $G(N, \epsilon)$ [168].) We sample N points uniformly at random from $[0, 1]^d$. We choose $(N, d) = (50, 16)$ and we denote this data set by **random**. The homology of random VR complexes was studied in [114].

(3) Vicsek biological aggregation model. This model was first introduced in [200] and was studied using PH in [196]. We implement the model in the form in which it appears in [196]. The model describes the motion of a collection of particles that interact in a square with periodic boundary conditions. The parameters for the model are the length l of the side of the square, the initial angle θ_0, the fixed absolute value for the velocity v_0, the number of particles N, a noise parameter η, and the number T of time steps. The output of the model is a point cloud in 3-dimensional Euclidean space in which each point is specified by its position in the 2-dimensional box and its velocity angle. We run three simulations of the model using the parameter values used in [196]. For each simulation, we choose two point clouds that correspond to two different time frames. See [196] for further details. We denote this data set by **Vicsek**.

(4) Fractal networks. These are self-similar networks introduced in [191] to investigate whether connection patterns of the cerebral cortex are arranged in self-similar patterns. The parameters for this model are natural numbers b, k, and n. To generate a fractal network, one starts with a fully-connected network on 2^b nodes. Two copies of this network are connected to each other so that the "connection density" between them is k^{-1}, where the connection density is the number of edges between the two copies divided by the number of total possible edges between them. Two copies of the resulting network are connected with connection density k^{-2}. One repeats this type of connection process until the network has size 2^n, but with a decrease in the connnection density by a factor of $1/k$ at each step.

[10] **Singular homology** is a method that assigns to every topological space homology groups encoding invariants of the space, in an analogous way as simplicial homology assigns homology groups to simplicial complexes. See [103] for an account of singular homology.

We define distances between nodes in two different ways: (1) uniformly at random, and (2) with linear weight–degree correlations. In the latter, the distance between nodes i and j is distributed as $k_i k_j X$, where k_i is the degree of node i and X is a random variable uniformly distributed on the unit interval. We use the parameters $b = 5$, $n = 9$, and $k = 2$; and we compute PH for the weighted network and for the network in which all adjacent nodes have distance 1. We denote this data set by **fract** and distinguish between the two ways of defining distances between weights using the abbreviations "r" for random, and "l" for linear.

(5) Genomic sequences of the HIV virus. We construct a finite metric space using the independent and concatenated sequences of the three largest genes — `gag`, `pol`, and `env` — of the HIV genome. We take 1088 different genomic sequences and compute distances between them by using the Hamming distance. We use the aligned sequences studied using PH in [41]. (The authors of that paper retrieved the sequences from [131].) We denote this data set by **HIV**.

(6) Genomic sequences of H3N2. These are 1000 different genomic sequences of H3N2 influenza. We compute the Hamming distance between sequences. We use the aligned sequences studied using PH in [41]. We denote this data set by **H3N2**.

(7) Stanford Dragon graphic. We sample points uniformly at random from 3-dimensional scans of the dragon [132], whose reconstruction we show in Fig. 2.8. The sample sizes contain 1000 and 2000 points. We denote these data sets by **drag 1** and **drag 2**, respectively.

(8) *C. elegans* neuronal network. This is a weighted, undirected network in which each node is a neuron and edges represent synapses or gaps junctions. We use the network studied using PH in [171]. (The authors of the paper used the data set studied in [204], which first appeared in [208].) Recall that for this example, and also for the other real-world weighted networks (except for the human genome network and for the US Congress networks), we convert each nonzero edge weight to a distance by taking its inverse. We denote this data set by **eleg**.

(9) Human genome. A weighted, undirected network representing a sample of the human genome. We use the network studied using PH in [171]. (The authors of that paper created the sample using data retrieved from [57].) Each node represents a gene, and weighted edges between nodes represent the correlation of the expression

level of the corresponding genes. We define the weight of an edge as the inverse of the correlation.[11] We denote this data set by **genome**.

(10) Grey-scale image: 3-dimensional rotational angiography scan of a head with an aneurysm. This data set was used in the benchmarking in [15]. This data set is given by a 3-dimensional array of size $512 \times 512 \times 512$, with each entry storing an integer that represents the grey value for the corresponding voxel. We retrieved the data set from the repository [1]. We denote this data set by **vertebra**.

(11) US Congress roll-call voting networks. These two networks (the Senate and House of Representatives from the 104th United States Congress) are constructed using the procedure in [205] from data compiled by [173]. In each network, a node is a legislator (Senators in one data set and Representatives in the other), and there is a weighted edge between legislators i and j, where the weight $w_{i,j}$ is a number in $[0, 1]$ (it is equal to 0 if and only if legislators i and j never voted the same way on any bill) given by the number of times the two legislators voted in the same way divided by the total number of bills on which they both voted. See [205] for further details. We denote the networks from the Senate and House by **senate** and **house**, respectively. The network **senate** has 103 nodes, and the network **house** has 445 nodes. To compute shortest paths, we define the distance between two nodes i and j to be $1 - w_{i,j}$. In the 104th Congress, no two politicians voted in the same way on every bill, so we do not have distinct nodes with 0 distance between them. (This is important, for example, if one wants to apply multidimensional scaling.)

(12) Network of network scientists. This is a weighted undirected network representing the largest connected component of a collaboration network of network scientists [163]. Nodes represent authors and edges represent collaborations, where weights indicate the number of joint papers. The largest connected components consists of 379 nodes. We denote this data set by **netw-sc**.

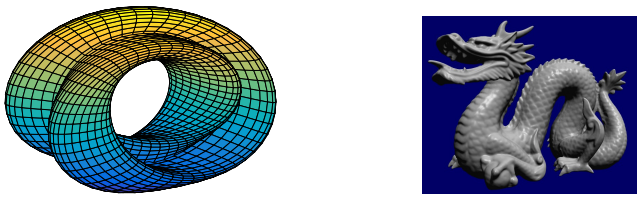


Figure 2.8: Two well-known examples. Plot of the image of the figure-8 immersion of the Klein bottle and the reconstruction of the Stanford Dragon (retrieved from [132]).

[11] We note that the weight should be the correlation; this issue came to our attention when the paper was in press.

2.7.1.2 Machines and compilers

We tested the libraries on both a cluster and a shared-memory system. The cluster is a Dell Sandybridge cluster, it has 1728 (i.e., 108×16) cores of 2.0GHz Xeon SandyBridge, RAM of 64 GiB in 80 nodes and RAM of 128 GiB in 4 nodes, and a scratch disk of 20 TB. It runs the operating system (OS) Red Hat Enterprise Linux 6. The shared-memory system is an IBM System x3550 M4 server with 16 (i.e., 2×8) cores of 3.3GHz, RAM of 768 GB, and storage of 3 TB. It runs the OS Ubuntu 14.04.01.[12] The major difference in running shared algorithms on the shared-memory system versus the distributed-memory system is that each node in the former has much more available RAM than in the latter. (See also the difference in performance between computations on cluster and shared memory system in Tables 2.3 and 2.4.) To compile Gudhi, DIPHA, Perseus and Dionysus we used the compiler `gcc` 4.8.2 on the cluster, and `gcc` 4.8.4 on the shared-memory system; for both machines we used the (default) optimisation `-O3`. Additionally, we used `openmpi` 1.8.3 for DIPHA.

2.7.1.3 Tests and results

We now report the details and results of the tests that we performed. We have made the data sets, header file to measure memory, and other information related to the tests available at `https://github.com/n-otter/PH-roadmap`. Of the six software packages that we study, four implement the computation of the dual algorithm, and four implement the standard algorithm. It is reported in [193] that javaPlex implements the dual algorithm, but the implementation of the algorithm has a bug and gives a wrong output. To our knowledge, this bug has not yet been fixed (at the time of writing), and we therefore test only the standard algorithm.

For the computations on the cluster, we compare the libraries running both the dual algorithm and the standard algorithm. The package DIPHA is the only one to implement a distributed computation. As a default, we run the software on one node and 16 cores; we only increase the number of nodes and cores employed when the machine runs out of memory. However, augmenting the number of nodes can make the computations faster (in terms of CPU seconds) for complexes of all sizes.[13] We see this in our experiments, and it is also discussed in [15]. For the other packages, we run the computations on a single node with one core.

[12]Note that we performed the computations for Gudhi and ripser at a different point in time, during which the shared-memory system was running the OS Ubuntu 16.04.01.

[13]Based on the results of our tests, we think of small, medium, and large complexes, respectively, as complexes with a size of order of magnitude of up to 10 million simplices, between 10 million and 100 million simplices, and between 100 million and a billion simplices.

For computations on the shared-memory system, we compare the libraries using only the dual algorithm if they implement it, and we otherwise use the standard algorithm. For the shared-memory system, we run all packages (including DIPHA) on a single core.

In our benchmarking, we report mean computation times and memory measurements. In Table 2.3, we give the computation times for the different software packages. We measure elapsed and CPU time by using the `time` function in Linux. We report computation times with a precision of one second; if a computation took only fractions of a second, we report "one second" as the computation time. For space reasons, we report results for a subset of the computations. (In Appendix B, we tabulate the rest of our computations.) In Table 2.4, we report the memory used by the processes in terms of maximum resident set size (RSS); in other words, we give the maximum amount of real RAM a program has used during its execution. We measure the maximum RSS using the `getrusage` function in Linux. The header file that we use to measure memory is available at `https://github.com/n-otter/PH-roadmap`. In DIPHA, the measurement of memory is already implemented by the authors of the software. They also use the `getrusage` function in Linux. The package javaPlex is written in Java, and we thus cannot measure its memory as we do for the other packages. However, one can infer memory requirements for this software package using the value of the maximal heap size necessary to perform the computations; we report this value in Table 2.4. In Table 2.5, we give the maximum size of the simplicial complex for which we were able to compute PH with each software package in our benchmarkings.

2.7.2 Conclusions from our benchmarking

Our tests suggest that ripser is the best-performing library currently available for the computation of PH with the Vietoris–Rips complex, and in order of decreasing performance, that Gudhi and DIPHA are the next-best implementations. For the computation of PH with cubical complexes, Gudhi outperforms DIPHA by a factor of 3 to 4 in terms of memory usage, and DIPHA outperforms Gudhi in terms of wall-time seconds by a factor of 1 to 2 (when running on one core on a shared-memory system). Both DIPHA and Gudhi significantly outperform the implementation in Perseus. For the computation of PH with the alpha complex, we did not observe any significant differences in performance between the libraries Gudhi and Dionysus. Because the alpha complex has fewer simplices than the other complexes that we used in our tests, further tests with larger data sets may be appropriate in future benchmarkings.

There is a huge disparity between implementations of the dual and standard algorithms. In our benchmarking, the dual implementations outperformed standard ones

Table 2.3: Performance of the software packages measured in wall-time (i.e., elapsed time), and CPU seconds (for the computations running on the cluster). For each data set, we indicate the size of the simplicial complex and the maximum dimension up to which we construct the VR complex. For all data sets, we construct the filtered VR complex up to the maximum distance between any two points. We indicate the implementation of the standard algorithm using the abbreviation "st" following the name of the package, and we indicate the implementation of the dual algorithm using the abbreviation "d." The symbol "-" signifies that we were unable to finish computations for this data set, because the machine ran out of memory. PERSEUS implements only the standard algorithm, and GUDHI and RIPSER implement only the dual algorithm. (a,b) We run DIPHA on one node and 16 cores for the data sets **eleg**, **Klein**, and **genome**; on 2 nodes of 16 cores for the **HIV** data set; on 2 and 3 nodes of 16 cores for the dual and standard implementations, respectively, for **drag 2**; and on 8 nodes of 16 cores for **random**. (The maximum number of processes that we could use at any one time was 128.) (c) We run DIPHA on a single core.

(a) Computations on cluster: wall-time seconds

Data set	**eleg**	**Klein**	**HIV**	**drag 2**	**random**	**genome**
Size of complex	4.4×10^6	1.1×10^7	2.1×10^8	1.3×10^9	3.1×10^9	4.5×10^8
Max. dim.	2	2	2	2	8	2
JAVAPLEX (st)	84	747	-	-	-	-
DIONYSUS (st)	474	1830	-	-	-	-
DIPHA (st)	6	90	1631	142559	-	9110
PERSEUS	543	1978	-	-	-	-
DIONYSUS (d)	513	145	-	-	-	-
DIPHA (d)	4	6	81	2358	5096	232
GUDHI	36	89	1798	14368	-	4753
RIPSER	1	1	2	6	349	3

(b) Computations on cluster: CPU seconds

Data set	**eleg**	**Klein**	**HIV**	**drag 2**	**random**	**genome**
Size of complex	4.4×10^6	1.1×10^7	2.1×10^8	1.3×10^9	3.1×10^9	4.5×10^8
Max. dim.	2	2	2	2	8	2
JAVAPLEX (st)	284	1031	-	-	-	-
DIONYSUS (st)	473	1824	-	-	-	-
DIPHA (st)	68	1360	25950	1489615	-	130972
PERSEUS	542	1974	-	-	-	-
DIONYSUS (d)	513	145	-	-	-	-
DIPHA (d)	39	73	1276	37572	79691	3622
GUDHI	36	88	1794	14351	-	4764
RIPSER	1	1	2	5	348	2

(c) Computations on shared-memory system: wall-time seconds

Data set	**eleg**	**Klein**	**HIV**	**drag 2**	**genome**	**fract r**
Size of complex	3.2×10^8	1.1×10^7	2.1×10^8	1.3×10^9	4.5×10^8	2.8×10^9
Max. dim.	3	2	2	2	2	3
JAVAPLEX (st)	13607	1358	43861	-	28064	-
PERSEUS	-	1271	-	-	-	-
DIONYSUS (d)	-	100	142055	35366	-	572764
DIPHA (d)	926	13	773	4482	1775	3923
GUDHI	381	6	177	1518	442	4590
RIPSER	2	1	2	5	3	1517

Table 2.4: Memory usage in GB for the computations summarised in Table 2.3. For javaPlex, we indicate the value of the maximum heap size that was sufficient to perform the computation. The value that we give is an upper bound to the memory usage. For DIPHA, we indicate the maximum memory used by a single core (considering all cores). See Table 2.3 for details on the number of cores used.

(a) Computations on cluster

Data set	**eleg**	**Klein**	**HIV**	**drag 2**	**random**	**genome**
Size of complex	4.4×10^6	1.1×10^7	2.1×10^8	1.3×10^9	3.1×10^9	4.5×10^8
Max. dim.	2	2	2	2	8	2
javaPlex (st)	< 5	< 15	> 64	> 64	> 64	> 64
Dionysus (st)	1.3	11.6	-	-	-	-
DIPHA (st)	0.1	0.2	2.7	4.9	-	4.8
Perseus	5.1	12.7	-	-	-	-
Dionysus (d)	0.5	1.1	-	-	-	-
DIPHA (d)	0.1	0.2	1.8	13.8	9.6	6.3
Gudhi	0.2	0.5	8.5	62.8	-	21.5
ripser	0.007	0.02	0.06	0.2	24.7	0.07

(b) Computations on shared-memory system

Data set	**eleg**	**Klein**	**HIV**	**drag 2**	**genome**	**fract r**
Size of complex	3.2×10^8	1.1×10^7	2.1×10^8	1.3×10^9	4.5×10^8	2.8×10^9
Max. dim.	3	2	2	2	2	3
javaPlex (st)	< 600	< 15	< 700	> 700	< 700	> 700
Perseus	-	11.7	-	-	-	-
Dionysus (d)	-	1.1	16.8	134.2	-	268.5
DIPHA (d)	31.2	0.9	17.7	109.5	37.3	276.1
Gudhi	15.4	0.5	10.2	62.8	21.4	134.8
ripser	0.2	0.03	0.07	0.2	0.07	155

Table 2.5: For each software package in (a), we indicate in (b) the maximal size of the simplicial complex for which we were able to compute PH thus far in our tests, the limitation being the physical amount of memory of the machines on which we performed the computations.

(a)	javaPlex	Dionysus		DIPHA		Perseus	Gudhi	ripser
	st	st	d	st	d	st	d	d
(b)	$4.5 \cdot 10^8$	$1.1 \cdot 10^7$	2.8×10^9	$1.3 \cdot 10^9$	$3.4 \cdot 10^9$	$1 \cdot 10^7$	$3.4 \cdot 10^9$	$3.4 \cdot 10^9$

both in terms of computation time (with respect to both CPU and wall-time seconds) and in terms of the amount of memory used. This significant difference in performance and memory usage was also revealed for the software package DIONYSUS in [61].

To conclude, in our benchmarking, the fastest software packages were RIPSER, GUDHI, and DIPHA. For small complexes, the software packages PERSEUS and JAVAPLEX are good choices, because they are the easiest ones to use. (They are the only libraries that need only to be downloaded and are then "plug-and-play," and they have user-friendly interfaces.) Because the library JAVAPLEX implements the computation of a variety of complexes and algorithms, we feel that it is the best software for an initial foray into PH computation.

We now give guidelines for the computation of PH based on our benchmarking. We list several types of data sets in Table 2.6 and indicate which software or algorithm that we feel is best-suited to each one. These guidelines are based on the findings of our benchmarking. Note that one can transform networks into distance matrices, and distance matrices can yield points in Euclidean space using a method such as multi-dimensional scaling. Naturally, given a finite set of points in Euclidean space, we can compute their distance matrix. As we discussed in Section 2.5.1, image data can also be considered as a finite metric space, so the indications for distance matrices and points in Euclidean space also apply to image data.

Table 2.6: Guidelines for which implementation is best-suited for which data set, based on our benchmarking. Recall that we indicate the implementation of the dual algorithm using the abbreviation "d" following the name of a package, and similarly we indicate the implementation of the standard algorithm by "st". Note that for smaller data sets one can also use JAVAPLEX to compute PH with VR complexes from points in Euclidean space, and PERSEUS to compute PH with cubical complexes for image data, and with VR complexes for distance matrices. The library JHOLES can only handle networks with density much less than 1.

Data type	Complexes	Suggested software
networks	WRCF	JHOLES
image data	cubical	GUDHI or DIPHA (st)
distance matrix	VR	RIPSER
distance matrix	W	JAVAPLEX
points in Euclidean space	VR	GUDHI
points in Euclidean space	Č	DIONYSUS
points in Euclidean space	α (only in dim 2 and 3)	DIONYSUS ((st) in dim 2, (d) in dim 3) or GUDHI

2.8 Future directions

We conclude by discussing some future directions for the computation of PH. As we saw in Section 2.5, much work has been done on step 2 (i.e., going from filtered complexes to barcodes) of the PH pipeline of Fig. 2.6, and there exist implementations of many fast algorithms for the reduction of the boundary matrix. Step 1 (i.e., going from data to a filtered complex) of the PH pipeline is an active area of research, but many sparsification techniques (see, e.g., [123, 184]) for complexes have yet to be implemented, and more research needs to be done on steps 1 and 3 (i.e., interpreting barcodes; see. e.g., [28, 31, 197]) of the PH pipeline. In particular, it is important to develop approaches for statistical analysis of persistent homology.

We believe that there needs to be a community-wide effort to build a library that implements the algorithms and data structures for the computation of PH, and that it should be done in a way that new algorithms and methods can be implemented easily in this framework. This would parallel similar community-wide efforts in fields such as computational algebra and computational geometry, and libraries such as Macaulay2 [99], Sage [64], and CGAL [195].

We also believe that there is a need to create guidelines and benchmark data sets for the test of new algorithms and data structures. The methods and collection of data sets that we used in our benchmarking provide an initial step towards establishing such guidelines and a list of test problems.

2.9 Availability of data and materials

The processed version of the data sets used in the benchmarking and the scripts written for the tutorial are available at `https://github.com/n-otter/PH-roadmap`. The open-source libraries for the computation of PH studied in this paper are available at the references indicated in the associated citations.

2.10 Additional acknowledgements

We thank the Rabadan Lab at Columbia University for providing the HIV and H3N2 sequences used in [41] and Giovanni Petri for sharing the data sets used in [171]. We thank Krishna Bhogaonker, Adrian Clough, Patrizio Frosini, Florian Klimm, Vitaliy Kurlin, Robert MacKay, James Meiss, Dane Taylor, Leo Speidel, Parker Edwards, and Bernadette Stolz for helpful comments on a draft of this paper. We also thank the anonymous referees for their many helpful comments. We thank Ulrich Bauer, Michael Lesnick,

Hubert Wagner, and Matthew Wright for helpful discussions, and Florian Klimm, Vidit Nanda, and Bernadette Stolz for precious advice. We would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility (`http://dx.doi.org/10.5281/zenodo.22558`) in carrying out some of the computations performed in this work. We thank the support team at the ARC for their assistance. We are grateful for support from the EPSRC grant EP/G065802/1 (The Digital Economy HORIZON Hub), by The Alan Turing Institute through EPSRC grant EP/N510129/1, and by the EPSRC institutional grant D4D01270 BKA1.01.

Chapter 3

Stratifying multiparameter persistent homology[1]

A fundamental tool in topological data analysis is persistent homology, which allows extraction of information from complex datasets in a robust way. Persistent homology assigns a module over a principal ideal domain to a one-parameter family of spaces obtained from the data. In applications data often depend on several parameters, and in this case one is interested in studying the persistent homology of a multiparameter family of spaces associated to the data. While the theory of persistent homology for one-parameter families is well-understood, the situation for multiparameter families is more delicate. Following Carlsson and Zomorodian we recast the problem in the setting of multigraded algebra, and we propose multigraded Hilbert series, multigraded associated primes and local cohomology as invariants for studying multiparameter persistent homology. Multigraded associated primes provide a stratification of the region where a multigraded module does not vanish, while multigraded Hilbert series and local cohomology give a measure of the size of components of the module supported on different strata. These invariants generalise in a suitable sense the invariant for the one-parameter case.

3.1 Introduction

In [38], Carlsson and Zomorodian introduced multiparameter persistent homology as a way of extending persistent homology to the setting of filtrations depending on more than one parameter. Near the end of their paper, they write

> Our study of multigraded objects shows that no complete discrete invariant exists for multidimensional persistence. We still desire a discriminating invari-

[1]This is the reproduction of the paper Stratifying multiparameter persistent homology, H. Harrington, N. Otter, U. Tillmann, H. Schenck, revising for *Journal of Applied and Computational Topology*, Springer.

ant that captures persistent information, that is, homology classes with large persistence.

We propose several such discriminating invariants by investigating the multiparameter persistent homology (MPH) of a multifiltered simplicial complex from the standpoint of multigraded commutative algebra. Our invariants distinguish between elements of the module that live forever, which we call "fully persistent components" and correspond to free elements in the module, elements that live along multiple but not all directions, which we call "partially persistent components" and correspond to free elements in certain submodules of the module, and elements that die in all directions, which we call "transient components".

Table 3.1: A brief overview of the dictionary between terms used in persistent homology and commutative algebra proposed in this paper. We note that in the one-parameter case the notion of death of a homogeneous element corresponds to a correction of the standard definition that uses the so-called "elder rule", see [165, Remark 5] for more details.

	r-parameter PH		commutative algebra
	$r = 1$	$r > 1$	
free	number of infinite intervals	minimal number of generators of submodule generated by fully persistent components	rank
torsion	—	minimal number of generators of submodule generated by partially persistent components living along $c_{\mathfrak{p}}$	$c_{\mathfrak{p}}$-rank
torsion	elements corresponding to finite intervals	transient components	$\bigcup_n \bigcap_{i=1}^{r} \ker(x_i^n)$
a homogeneous element a is born at $\mathbf{u} \in \mathbb{N}^r$			$\deg(a) = \mathbf{u}$ and a is not in the image of $\sum_{\mathbf{v}} x^{\mathbf{v}}$ for any $\mathbf{v} \prec \mathbf{u}$
a homogeneous element a dies in degrees $D \subset \mathbb{N}^r$			$\mathrm{Ann}(a) \neq (0)$, and $D \subset \mathbb{N}^r$ is obtained from the set of degrees of the set of minimal generators of $\mathrm{Ann}(a)$ by adding to each degree the degree of a
a homogeneous element a lives forever			$\mathrm{Ann}(a) = (0)$

The key objects that we use are the Hilbert function, which gives a measure of the size of the fully persistent components, the associated primes, which provide a stratification of the support of MPH modules in transient, partially and fully persistent components, and local cohomology, which gives a measure of the size of the partially persistent components. Though standard in commutative algebra, their use in TDA is new.

In addition to addressing the question posed by Carlsson and Zomorodian, a further aim of this paper is to provide an introduction to methods of commutative algebra to the TDA community, and to highlight their utility in TDA. To this end, we include background, definitions, and examples of the key objects, including

1. Graded and multigraded algebra, which is the setting for MPH,
2. The Hilbert function, which captures the rank of a multigraded MPH module, yielding a measure of the size of the fully persistent components,
3. The associated primes, which provide a stratification of the support of MPH modules into transient, partially and fully persistent components, and
4. Local cohomology, which gives a measure of the size of the partially persistent components.

We provide an overview of the dictionary between notions from persistent homology and commutative algebra proposed in this paper in Table 3.1.

We note that what we call multiparameter persistent homology is often also called multidimensional persistent homology in the literature. We believe that the adjective "multiparameter" is more appropriate. Furthermore, the term "dimension" is often used to denote the degree of (persistent) homology, and its use could thus cause unnecessary confusion.

3.1.1 Persistent homology

Persistent homology was introduced about 20 years ago, and has become a standard tool in the analysis of datasets with complicated structure; for overviews see [34], [73], [92], [207], and [165] for an introduction to computations. Persistent homology (PH) has been successful in attacking many problems, ranging from analysis of activity in the visual cortex [188] to understanding viral evolution [41] to modeling shapes and surfaces [198].

PH is appealing for applications because it provides a robust and coordinate-independent method to study qualitative features of data across different values of a parameter. One can think of the different parameter values as scales of resolution, and PH provides a summary of how long individual qualitative features persist across the different scales of resolution. Roughly, persistent homology is the homology of a nested sequence of spaces $X_1 \subset \cdots \subset X_n = X$ associated with a suitable data set. The homology modules of such a filtered space X come with additional structure as finitely generated graded modules over the polynomial ring $\mathbb{K}[x]$ in one variable with coefficients in a field $\mathbb{K}$. The grading gives information about the position in the filtration, while the action of x gives a shift in

the filtration by one position. Since the ring $S = \mathbb{K}[x]$ is a principal ideal domain (PID), the classical structure theorem for a finitely generated module M over a PID S applies:

$$M \simeq S^n \oplus \bigoplus_{i=1}^{m} S/p_i,$$

where the p_i are non-trivial principal ideals; the components S/p_i are torsion. Since in the setting of persistent homology the module additionally has a grading, one obtains the following decomposition:

$$M = \bigoplus_{i=1}^{h} x^{\alpha_i} S \oplus \bigoplus_{j=1}^{l} x^{\beta_j} S/(x^{\beta_j+\gamma_j}) \,. \qquad (\star)$$

One can read off from the decomposition in Equation ($\star$) a finite collection of infinite and finite intervals, which collectively are called a "barcode", and completely characterises the isomorphism class of the module: the torsion part gives l finite intervals $[\beta_j, \beta_j + \gamma_j)$, for $j = 1, \ldots, l$, that correspond to features that only exist for a finite number of filtration steps, while the free summands give h infinite intervals $[\alpha_i, \infty)$, for $i = 1, \ldots, h$, corresponding to features that are hardier and survive (persist) forever.

In many applications, data depend not only on one, but several parameters, and to apply PH to such data one therefore needs to study the evolution of qualitative features across several parameters. The homology of a multifiltered space has the structure of a multigraded module over a polynomial ring $S = \mathbb{K}[x_1, \ldots, x_r]$ in r variables, where r is the number of parameters. Unfortunately, in this case we no longer have a decomposition such as the one for the one-parameter case, and the problem of finding a complete characterisation of isomorphism classes analogous to the one-parameter case has been shown to be hopeless [38].

On the other hand, for applications one does not need a complete classification of modules; rather, what one needs are invariants that are computable, suitably robust with respect to perturbations in input data, and amenable to statistical interpretation, in the sense that one can assess the quality or uncertainty of the resulting invariant using statistical techniques.

3.1.2 Related work

Several different research approaches have been taken to study multiparameter persistent homology:

- Rank invariant: in [38] the authors propose the rank invariant as an invariant for multiparameter modules. They show that this invariant is equivalent to the barcode

in the one-parameter case. We briefly discuss the rank invariant in Section 3.4.5 and relate it to our work.

- Efficient computation of presentations of modules: in [37] the authors propose a polynomial-time algorithm to compute a presentation of homology of "one-critical" multifiltered simplicial complexes, which roughly are multifiltered simplicial complexes in which each simplex enters the complex at exactly one filtration value. In [40] a polynomial-time algorithm for the computation of the presentation of the homology of an arbitrary multifiltered simplicial complex is proposed.

- Restriction to a line: in [20] the authors study the collection of one-parameter modules associated to a multiparameter module by restricting it to lines with positive real slopes. They show that such a collection of one-parameter modules gives the same information as the rank invariant. Building on this, in [140] the authors introduce a tool for the visualisation of barcodes of a multiparameter module restricted to a line. We discuss several ways of associating an $\mathbb{N}$-graded module to an $\mathbb{N}^r$-graded module in Section 3.5.

- Noise: in [183] a general notion of noise for persistence modules is proposed, and related invariants associated to multiparameter modules are studied. The proposed invariants are however not computable, and therefore they are more of theoretical interest.

- Tor modules [124]: invariants that refine the discrete invariants proposed in [38].

- Fringe presentations [154]: this is the work that is most closely related to our work. It studies births and deaths of generators in the more general setting of modules over the monoid ring $\mathbb{K}[\mathbb{R}^r]$.

3.1.3 Structure of the chapter

The chapter is structured as follows:

- In Section 3.2, we review multiparameter persistent homology and multigraded algebra, and we present a proof of the well-known fact that every multigraded module is the homology of a multifiltered simplicial complex.

- In Section 3.3, we discuss invariants associated with the free resolution of a module: the Hilbert function, multigraded Hilbert series and Hilbert polynomial. We show that the Hilbert function and Hilbert series are invariants encoding key properties of MPH. Specifically, for a module M over an integral domain, the natural measure

of size is the rank $\mathrm{rk}(M)$, which can be read off from its Hilbert function and multigraded Hilbert series. Furthermore, we show that one can reduce the computation of the rank of a module to the computation of (ordinary) homology of a simplicial complex.

- In Section 3.4, we introduce a finer invariant for modules, which is given by the associated primes; we discuss how minimal associated primes give information on the coordinate subspaces on which the module does not vanish, while non-minimal (so-called embedded) associated primes give a stratification of these subspaces (see Fig. 3.5 for an example of two modules that have the same Hilbert series, but different stratifications). We then use local cohomology to give a measure of the size of the submodule generated by elements that only live along some direction, relate our work to the rank invariant introduced in [38], and show that the associated primes may be computed by only computing the cokernels of the maps in an appropriate chain complex.

- In Section 3.5 we investigate different ways to associate an $\mathbb{N}$-graded module over a polynomial ring in one variable to an $\mathbb{N}^r$-graded module over $\mathbb{K}[x_1, \ldots, x_r]$. First, we adapt the restriction to a line studied in [20, 140] for $\mathbb{R}^r$-graded modules to our $\mathbb{N}^r$-graded setting. We then discuss how one can restrict a module to the diagonal using methods from commutative algebra, and show that one can read off the rank of a module from the rank of a module restricted to the diagonal.

- In Section 3.6, we summarise our results and discuss future work, and we provide `Macaulay2` code that we used for the computations in this paper in Appendix C.

3.2 Multiparameter persistence and multigraded algebra

It is well known that studying multiparameter persistent homology amounts to the study of a module over a multivariate polynomial ring. In this section we first review standard but useful facts about multiparameter persistent homology; we then give a brief overview of multigraded modules and recall the Hilbert Syzygy Theorem. Finally, we present a proof of the well-known fact that any multigraded module is the homology of a multifiltered space.

3.2.1 Multiparameter persistence

Definition 3.2.1. Denote by $\mathbb{N}^r$ the set of r-tuples of natural numbers, and define the following partial order on $\mathbb{N}^r$: for any pair of elements $\mathbf{u}, \mathbf{v} \in \mathbb{N}^r$ we define $\mathbf{u} \preccurlyeq \mathbf{v}$ iff $u_i \leq v_i$ for all $i = 1, \dots r$, where we write $\mathbf{u} = (u_1, \dots, u_r)$. Given a collection of simplicial complexes $\{K_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}$ indexed by $\mathbb{N}^r$, we say that $\{K_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}$ is an ***r*-filtration** if whenever $\mathbf{u} \preccurlyeq \mathbf{v}$ we have that $K_\mathbf{u} \subseteq K_\mathbf{v}$. If there exists $\mathbf{u}' \in \mathbb{N}^r$ such that $K_\mathbf{u} = K_{\mathbf{u}'}$ for all $\mathbf{u}' \preccurlyeq \mathbf{u}$, then we say that the r-filtration **stabilises**. A **multifiltration** is an r-filtration for some r.

An ***r*-filtered simplicial complex** is a simplicial complex K together with a multifiltration $\{K_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}$ that stabilises and such that $K = \cup_{\mathbf{u}\in\mathbb{N}^r} K_\mathbf{u}$. An r-filtered simplicial complex $(K, \{K_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r})$ is **finite** if K is finite. A **multifiltered simplicial complex** is an r-filtered simplicial complex for some $r \geq 1$.

Given a multifiltered simplicial complex $(K, \{K_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r})$, for each $x \in K$ we call the minimal elements $\mathbf{u} \in \mathbb{N}^r$ (with respect to the partial order $\preccurlyeq$) at which it enters the filtration its **entry degrees**. If every $x \in K$ has exactly one entry degree we call the multifiltered simplicial complex **one-critical**.

Let K be a multifiltered simplicial complex, and let $i = 0, 1, 2, \dots$. For any $\mathbf{u} \in \mathbb{N}^r$ denote by $C_i(K_\mathbf{u})$ the $\mathbb{K}$-vector space with basis given by the i-simplices of $K_\mathbf{u}$, and similarly by $H_i(K_\mathbf{u})$ the ith simplicial homology with coefficients in $\mathbb{K}$. Whenever $\mathbf{u} \preccurlyeq \mathbf{v}$ we have that the inclusion maps $K_\mathbf{u} \to K_\mathbf{v}$ induce $\mathbb{K}$-linear maps $\psi_{\mathbf{u},\mathbf{v}} \colon C_i(K_\mathbf{u}) \to C_i(K_\mathbf{v})$ and $\phi_{\mathbf{u},\mathbf{v}} \colon H_i(K_\mathbf{u}) \to H_i(K_\mathbf{v})$ such that whenever $\mathbf{u} \preccurlyeq \mathbf{w} \preccurlyeq \mathbf{v}$ we have that $\psi_{\mathbf{w},\mathbf{v}} \circ \psi_{\mathbf{u},\mathbf{w}} = \psi_{\mathbf{u},\mathbf{v}}$, and similarly $\phi_{\mathbf{w},\mathbf{v}} \circ \phi_{\mathbf{u},\mathbf{w}} = \phi_{\mathbf{u},\mathbf{v}}$. We thus give the following definition:

Definition 3.2.2. Let K be a multifiltered simplicial complex. The ***i*th chain module** of K over $\mathbb{K}$ is the tuple $(\{C_i(K_\mathbf{u})\}_{\mathbf{u}\in\mathbb{N}^r}, \{\psi_{\mathbf{u},\mathbf{v}} \colon C_i(K_\mathbf{u}) \to C_i(K_\mathbf{v})\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$.

Similarly, the **simplicial homology** with coefficients in $\mathbb{K}$ of K is the tuple $(\{H_i(K_\mathbf{u})\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi_{\mathbf{u},\mathbf{v}} \colon H_i(K_\mathbf{u}) \to H_i(K_\mathbf{v})\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$, where the maps $\psi_{\mathbf{u},\mathbf{v}}$ and $\phi_{\mathbf{u},\mathbf{v}}$ are those induced by the inclusions.

The ith chain module and homology of a multifiltered simplicial complex are a "multiparameter" example of what is usually called a persistence module in the persistent homology literature:

Definition 3.2.3. An ***r*-parameter persistence module** is given by a tuple $(\{M_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi_{\mathbf{u},\mathbf{v}} \colon M_\mathbf{u} \to M_\mathbf{v}\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$ where $M_\mathbf{u}$ is a $\mathbb{K}$-module for each $\mathbf{u}$ and $\phi_{\mathbf{u},\mathbf{v}}$ is a $\mathbb{K}$-linear map, such that whenever $\mathbf{u} \preccurlyeq \mathbf{w} \preccurlyeq \mathbf{v}$ we have $\phi_{\mathbf{w},\mathbf{v}} \circ \phi_{\mathbf{u},\mathbf{w}} = \phi_{\mathbf{u},\mathbf{v}}$. A **multiparameter persistence module** is an r-parameter persistence module for some $r \geq 1$.

A **morphism** of multiparameter persistence modules

$$f\colon (\{M_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}}) \to \left(\{M'_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi'_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}}\right)$$

is a collection of $\mathbb{K}$-linear maps $\{f_{\mathbf{u}}\colon M_{\mathbf{u}} \to M'_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{N}^r}$ such that $f_{\mathbf{v}} \circ \phi_{\mathbf{u},\mathbf{v}} = \phi'_{\mathbf{u},\mathbf{v}} \circ f_{\mathbf{u}}$ for all $\mathbf{u} \preccurlyeq \mathbf{v}$.

Let K be a multifiltered simplicial complex. An example of a morphism of multiparameter persistence modules is given by the differentials of the simplicial chain complex $(C_\bullet(K_{\mathbf{u}}), d_\bullet\colon C_\bullet(K_{\mathbf{u}}) \to C_{\bullet-1}(K_{\mathbf{u}}))$, for each $\mathbf{u} \in \mathbb{N}^r$, which induce morphisms of multiparameter persistence modules

$$(\{C_i(K_{\mathbf{u}})\}_{\mathbf{u}\in\mathbb{N}^r}, \{\psi_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}}) \to (\{C_{i-1}(K_{\mathbf{u}})\}_{\mathbf{u}\in\mathbb{N}^r}, \{\psi_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}}) \tag{3.1}$$

for any $i > 0$.

Remark 3.2.4. One could equivalently define an r-parameter persistence module as being a functor from the poset category $\mathbb{N}^r$ to the category with objects $\mathbb{K}$-vector spaces and morphisms $\mathbb{K}$-linear maps, but we have opted to give a more hands-on definition in this paper.

3.2.2 Multigraded algebra

The name multiparameter persistence module is justified by the fact that a multiparameter persistence module is a module over the polynomial ring $\mathbb{K}[x_1, \ldots, x_r]$. In the following we recall some notions about multigraded rings and modules. In this chapter a ring will always mean a commutative ring with identity.

Definition 3.2.5. Let R be a ring. We say that R is **graded by** $\mathbb{N}^r$ (or $\mathbb{N}^r$**-graded**) if there is a collection of abelian groups $\{R_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{N}^r}$ such the underlying abelian group of R can be written as $\oplus_{\mathbf{u}\in\mathbb{N}^r} R_{\mathbf{u}}$, and such that this decomposition is compatible with the ring structure: for all $\mathbf{u}, \mathbf{v} \in \mathbb{N}^r$ and all $r_{\mathbf{u}} \in R_{\mathbf{u}}$ and $r_{\mathbf{v}} \in R_{\mathbf{v}}$ we have that $r_{\mathbf{u}} r_{\mathbf{v}} \in R_{\mathbf{u}+\mathbf{v}}$. The **homogeneous elements** of the ring R are the elements $r \in R$ such that there exists $\mathbf{u} \in \mathbb{N}^r$ with $r \in R_{\mathbf{u}}$, and $\mathbf{u}$ is called the **degree** of r and denoted by $\deg(r)$.

Given a module M over an $\mathbb{N}^r$-graded ring R, we say that M is **graded by** $\mathbb{N}^r$ (or $\mathbb{N}^r$**-graded**) if there exists a collection $\{M_{\mathbf{u}}\}_{\mathbf{u}\in\mathbb{N}^r}$ of abelian groups such that the underlying abelian group of M can be written as $\oplus_{\mathbf{u}\in\mathbb{N}^r} M_{\mathbf{u}}$ and the decomposition is compatible with the R-module structure: for all $\mathbf{u}, \mathbf{v} \in \mathbb{N}^r$ we have that $rm \in M_{\mathbf{u}+\mathbf{v}}$ for all $r \in R_{\mathbf{u}}$ and all $m \in M_{\mathbf{v}}$. Similarly as for graded rings, an element m of the module which is contained in one of the direct summands $M_{\mathbf{u}}$ is called **homogeneous**, $\mathbf{u}$ is called the **degree** of m, and denoted by $\deg(m)$.

A **homomorphism** of $\mathbb{N}^r$-graded modules is a homomorphism $f\colon M \to N$ of $\mathbb{N}^r$-graded modules such that for all $\mathbf{u} \in \mathbb{N}^r$ we have $f(M_\mathbf{u}) \subset N_\mathbf{u}$.

Now consider the polynomial ring $S = \mathbb{K}[x_1, \ldots, x_r]$. This ring is naturally graded by $\mathbb{N}^r$:

$$S = \bigoplus_{\mathbf{u}\in\mathbb{N}^r} \mathbb{K}\mathbf{x}^\mathbf{u}$$

where we denote by $\mathbf{u}$ the r-tuple $(u_1, \ldots, u_r)$ and the notation $\mathbf{x}^\mathbf{u}$ stands for $x_1^{u_1} \ldots x_r^{u_r}$.

Given a multiparameter persistence module $(\{M_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$ let M denote the direct sum $\bigoplus_{\mathbf{u}\in\mathbb{N}^r} M_\mathbf{u}$. Then M is an S-module graded by $\mathbb{N}^r$: the action of $\mathbb{K}$ is given componentwise on each direct summand, and similarly for any $\mathbf{v} \in \mathbb{N}^r$ the action of $\mathbf{x}^\mathbf{u}$ on $M_\mathbf{v}$ is given by the linear map $\phi_{\mathbf{v},\mathbf{v}+\mathbf{u}}$.

Conversely, given an S-module M graded by $\mathbb{N}^r$, we obtain a multiparameter persistence module $(\{M_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\mathbf{x}^\mathbf{u}\colon M_{\mathbf{v}-\mathbf{u}} \to M_\mathbf{v}\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$. More precisely, we have the following:

Theorem 3.2.6. [38] There is an isomorphism of categories between the category with objects given by r-parameter persistence modules of finitely generated type[2] and morphisms given by morphisms of r-parameter persistence modules, and the category with objects given by finitely generated $\mathbb{N}^r$-graded modules over $\mathbb{K}[x_1, \ldots x_r]$ and morphisms homomorphisms of $\mathbb{N}^r$-graded modules.

Theorem 3.2.6 is usually called Correspondence Theorem in the persistent homology literature. Thanks to this theorem we can approach the problem of studying multiparameter persistent homology by reformulating it in the language of multigraded algebra. When dealing with applications, one is often interested in finitely generated $\mathbb{N}^r$-graded modules over the polynomial ring $S = \mathbb{K}[x_1, \ldots, x_r]$, where we adopt the notation common in commutative algebra. (See [154] for an application in which one needs to consider modules over the monoid ring $\mathbb{K}[\mathbb{R}^r]$.) Every S-module has a free resolution:

Definition 3.2.7. A free resolution for an S-module M is an exact sequence

$$F_\bullet : \cdots \longrightarrow F_i \xrightarrow{d_i} F_{i-1} \longrightarrow \cdots \longrightarrow F_0 \longrightarrow M \longrightarrow 0,$$

where the F_i are free S-modules. A free resolution is **$\mathbb{N}^r$-graded** if the S-modules F_i are $\mathbb{N}^r$-graded for each i and the homomorphisms d_i are homomorphisms of $\mathbb{N}^r$-graded modules. We say that a resolution has **length n** if $F_{n'} = 0$ for all $n' > n$.

[2] A persistence module $(\{M_\mathbf{u}\}_{\mathbf{u}\in\mathbb{N}^r}, \{\phi_{\mathbf{u},\mathbf{v}}\}_{\mathbf{u}\preccurlyeq\mathbf{v}})$ is said to be of **finitely generated type** if $M_\mathbf{u}$ is a finite-dimensional vector space for all $\mathbf{u}$, and furthermore there exists $\mathbf{u}_0$ such that for all $\mathbf{u}_0 \preccurlyeq \mathbf{v}$ the map $\phi_{\mathbf{u}_0,\mathbf{v}}$ is an isomorphism. We note that in [38] such modules are called finite.

By the Hilbert Syzygy Theorem the free resolution of a finitely generated module has finite length:

Theorem 3.2.8. [155] Every finitely generated $\mathbb{N}^r$-graded S-module M has a free $\mathbb{N}^r$-graded resolution of length at most r such that the free modules in the resolution are finitely generated.

In particular, for $r = 1$ we obtain that the free resolution of a finitely generated S-module has length 1, and therefore there is a finite set of n generators, as well as a finite set of m relations such that we can write the module M as the cokernel of a diagram of free $\mathbb{N}^r$-graded S-modules of the following form:

$$\bigoplus_{j=1}^{m} S(-\mathbf{u}_j) \longrightarrow \bigoplus_{i=1}^{n} S(-\mathbf{v}_i)\,,$$

where we use the convention that a free S-module of rank one with homogeneous generator of degree $\mathbf{u}$ is denoted by $S(-\mathbf{u})$, so that $S(-\mathbf{u})_{\mathbf{v}} = S_{-\mathbf{u}+\mathbf{v}}$, and thus in particular the homogeneous elements of degree $\mathbf{u}$ of $S(-\mathbf{u})$ are the homogeneous elements of degree 0 of S. We also say that $S(-\mathbf{u})$ is S **shifted by u**.

One can then write the module M uniquely as a direct sum [206]

$$M = \bigoplus_{i=1}^{h} S(-\alpha_i) \oplus \bigoplus_{j=1}^{l} S(-\beta_j)/\gamma_j\,, \tag{3.2}$$

where the α_i, β_j and γ_j are positive integers.

When $r > 1$ one in general no longer has a decomposition such as the one in Equation (3.2) because of the presence of non-trivial relations between relations, which are called **2nd syzygies**, as well as syzygies of higher order.

A convenient reference for the theory of multigraded modules is [155].

Remark 3.2.9. Associated to a simplicial complex Δ on v vertices is the Stanley-Reisner ring $\mathbb{K}[\Delta]$, which is $\mathbb{N}^v$ graded (see [155, §1.1]), with vertex i having degree $\mathbf{e}_i$, edge v_iv_j of degree $\mathbf{e_i} + \mathbf{e_j}$, and so on. This differs from the grading of multiparameter persistence;for example in Example 3.3.22, vertices c and e occur in degree $(0, 0)$, while edge ce occurs in degree $(0, 1)$.

Remark 3.2.10. Free resolutions depend on the characteristic of the underlying field, even for monomial ideals: a classical example of Reisner [176] shows that the Stanley–Reisner ideal for the standard six vertex triangulation of $\mathbb{RP}^2$ has different free resolution over $\mathbb{Q}$ and over $\mathbb{Z}/2$, reflecting the difference in simplicial homology with these choices of coefficients.

3.2.3 Multigraded modules as multiparameter persistence modules

We have seen that the homology of a multifiltered space is an example of a multiparameter persistence module. The converse is also true, as stated first in [38] without proof. For completeness we provide a proof of this statement.

Theorem 3.2.11. If the coefficient ring $\mathbb{K}$ is a prime field $\mathbb{F}_p$, $\mathbb{Q}$, or $\mathbb{Z}$, then any finitely generated, $\mathbb{N}^r$-graded S-module M can be realised as the homology in degree i, for $i > 0$, of a finite multifiltered simplicial complex.

Proof. By assumption, because M is $\mathbb{N}^r$-graded, it is generated by a finite set $\mathcal{G}$ of homogeneous elements. Let $F_S(\mathcal{G})$ denote the free S-module generated by $\mathcal{G}$ and let N denote the kernel of the canonical surjective homomorphisms $F_S(\mathcal{G}) \to M$. By Hilbert's Syzygy Theorem (Theorem 3.2.8), N itself is generated by a finite set $\mathcal{R}$ of homogeneous elements. Each $r \in \mathcal{R}$ can be written as a finite linear combination

$$r = \sum_{g \in \mathcal{G}} c(r,g)\mathbf{x}^{\deg(r)-\deg(g)}g$$

with $c(r,g) \in \mathbb{K}$.

When $\mathbb{K}$ is the prime field $\mathbb{F}_p = \{0, \ldots, p-1\}$ or the integers $\mathbb{Z}$, then we can interpret $c(r,g)$ as an integer. When $\mathbb{K} = \mathbb{Q}$, we clear the denominators in the above equation by multiplying by n_r, a multiple of the divisors of the non-zero $c(r,g)$ as $g \in \mathcal{G}$ varies. Replacing the relation r by $n_r r$, we may now assume that the coefficients $c(r,g)$ are integers.

We will first build a finite, multifiltered cell complex X with $\mathbb{N}^r$-graded i-th homology isomorphic to M. Let

$$Z = \bigvee_{g \in \mathcal{G}} S^i_g$$

be a bouquet of i-dimensional spheres, one for each generator g of M. More precisely, Z is the cell complex composed of a base point $*$ (of multidegree $\mathbf{0}$) and i-cells e^i_g in degree $\deg(g)$, one for each generator g and attached to $*$ by the constant map. For each relation $r \in \mathcal{R}$, we attach an $i+1$-cell e^{i+1}_r in degree $\deg(r)$ by a map $\phi_r : \partial e^{i+1}_r = S^i_r \to Z$ satisfying the following property. Define the projection map $P_g : Z \to S^i_g$ to be the map that is the identity on S^i_g and collapses all other spheres onto the base point. Then ϕ_r is chosen such that for all g and r, the composition $P_g \circ \phi_r : S^i_r \to S^i_g$ has degree $c(r,g)$. Let X be the resulting cell complex. It has a natural multifiltration

$$X_{\mathbf{v}} = \bigcup_{\deg(g) \preccurlyeq \mathbf{v}} e^i_g \cup \bigcup_{\deg(r) \preccurlyeq \mathbf{v}} e^{i+1}_r.$$

By construction the cellular $\mathbb{N}^r$-graded i-th homology of X is M. Indeed, the cellular chains of X are given in dimension i by $C_i(X) = S[e_g^i \,|\, g \in \mathcal{G}]$ and in dimension $i+1$ by $C_{i+1}(X) = S[e_r^{i+1} \,|\, r \in \mathcal{R}]$ with the ith boundary map constant zero and the $i+1$-th boundary given by

$$\partial_{i+1}(e_r^i) = \Sigma_{g \in \mathcal{G}}\ c(r,g)\, e_g^i.$$

Finally we note that the cell complex X can be replaced by a homotopy equivalent simplicial complex K. To do so, we replace Z by a homeomorphic simplicial complex where each sphere S_g^i is identified with $\partial \triangle_g^{i+1}$, a copy of the boundary of the standard $i+1$-simplex, all glued together at the first 0-simplex. Similarly, each $i+1$ cell e_r^{i+1} is identified with a copy of the standard $i+1$ simplex $\triangle_r^{i+1}$. By repeated application of the simplicial approximation theorem and taking subdivisions of the source, each of the attaching maps ϕ_r can be replaced by a homotopic simplicial map $\tilde{\phi}_r : \partial \triangle_r^{i+1} \to \partial \triangle_g^{i+1}$. Define K to be the union of the simplicial mapping cylinders for each $\tilde{\phi}_r$, then K has the same $\mathbb{N}^r$-graded i-th homology as X. □

A more general statement than the one in Theorem 3.2.11 is true:

Theorem 3.2.12. If in the setting of Theorem 3.2.11 one more generally assumes that the module M is tame, in the sense that each homogeneous part is finite-dimensional as a $\mathbb{K}$-module, then there exists a multifiltered simplicial complex K such that M is the simplicial homology in degree $i > 0$ of K.

Proof. As in the proof of Theorem 3.2.11, let $\mathcal{G}$ be a set of homogeneous generators for M. For $k = 0, 1, 2, \ldots$ denote by M^k the submodule of M generated by the elements of $\mathcal{G}$ that have degree smaller or equal to $(k, \ldots, k)$. By the assumption of tameness we have that M^k is finitely generated as a $\mathbb{K}[x_1, \ldots, x_r]$-module. Furthermore, for all $k \leq k'$ we have that M^k is a submodule of $M^{k'}$, and M is the colimit of the direct system given by the modules M^k together with the inclusion maps. Given an inclusion $M^k \hookrightarrow M^{k'}$ let $g_1^k, \ldots, g_m^k$ be the elements of $\mathcal{G}$ that generate M^k and $r_1^k, \ldots, r_n^k$ a set of relations for M^k. Then, the set of generators for $M^{k'}$ includes the generators for M^k, and we can inductively enlarge the set of relations for $M^{k'}$ so that it contains all relations $r_1^k, \ldots, r_n^k$, for any $k < k'$. Since M^k is finitely generated, we can follow the construction in the proof of Theorem 3.2.11 and consider the CW complex X^k with multifiltration

$$X_{\mathbf{v}}^k = \bigcup_{\deg(g) \preccurlyeq \mathbf{v}} e_g^i \cup \bigcup_{\deg(r) \preccurlyeq \mathbf{v}} e_r^{i+1}$$

such that its cellular homology in degree i is M^k, where g runs over the generators $g_1^k, \ldots, g_m^k$, and r runs over the relations $r_1^k, \ldots, r_n^k$. Similarly we consider a CW complex

$X^{k'}$ such that its homology in degree i is $M^{k'}$. By construction and our choice of generators and relations we have that X^k is contained in $X^{k'}$, and the inclusion map induces the inclusion $M^k \hookrightarrow M^{k'}$. Let X be the colimit of the direct system given by the multifiltered cell complexes X^k together with the inclusion maps. The cell complex X has multifiltration

$$X_{\mathbf{v}} = \bigcup_{\deg(g) \preccurlyeq \mathbf{v}} e_g^i \cup \bigcup_{\deg(r) \preccurlyeq \mathbf{v}} e_r^{i+1}$$

where g runs over the elements of $\mathcal{G}$ and r over the union of the sets of relations for the modules M^k, which is a set of relations for M. Since homology commutes with filtered colimits we have that M is the cellular homology in degree i of X. Similarly to the construction in the proof of Theorem 3.2.11, X can be replaced with a homotopy equivalent multifiltered simplicial complex K.

□

Remark 3.2.13. We note that the assumption that $i > 0$ is necessary. For, already in the ungraded case, that is in ordinary homology, there is a restriction in that no torsion can occur in zero-dimensional homology. This is because the only degrees that can occur when attaching a 1-cell to a 0-cell are $-1, 0$, or 1.

Remark 3.2.14. The reader may wonder whether the theorem is true for all fields $\mathbb{K}$. It is clearly true for non-graded modules as a vector spaces of dimension n is the i-th (reduced) homology of a wedge of n spheres of dimension i for all $i \geq 0$. It also holds for $r = 1$ as in that case $S = \mathbb{K}[x]$ is a principal ideal domain, and generators and relations can be found such that every relation is a multiple of one generator: $r = c(r, g)x^{\deg(r)-\deg(g)}g$, and on replacing g by $g' := c(r, g)g$, we have that the attaching degree is 1. However, when $r > 1$, several non-zero coefficients may occur in front of g which cannot be converted simultaneously into integers by changing the generators.

In light of Theorem 3.2.11 we will study general multigraded S-modules.

3.3 The rank of a module and its Hilbert function

A free resolution of a module is not an invariant, as a module in general has many non-isomorphic free resolutions. In this section we discuss invariants associated to the free resolution of a module, namely the Hilbert series, the Hilbert function and the Hilbert polynomial, and how one can read off the rank of a module from these invariants. The rank of a module can in a certain sense be seen as the most coarse invariant of a module; it is the number of minimal generators for a maximal free submodule of the module. In the one-parameter case this corresponds to the number of infinite intervals in the barcode.

Definition 3.3.15. [81, p. 261] For a module M defined over an integral domain R with field of fractions Q, the **rank** of M is

$$\mathrm{rk}_R(M) = \dim_Q M \otimes_R Q\,.$$

In particular, the rank of a free R-module R^n is n.

While Definition 3.3.15 is mathematically precise, it does not give any insight of how to actually compute $\mathrm{rk}_R(M)$, which is important for applications. When the integral domain R is the polynomial ring $S = \mathbb{K}[x_1, \ldots, x_r]$, and the module M is $\mathbb{N}^r$-graded and finitely generated, we know by Theorem 3.2.11 that there exists a finite multifiltered simplicial complex K such that $M \cong H_i(K)$. Since the multifiltered complex K stabilises, say in degree $\mathbf{s}$, $\mathrm{rk}_R(H_i(K))$ will equal the rank of the ordinary simplicial homology of the complex $K_{\mathbf{s}}$. In the following we introduce the Hilbert function, and make the previous statement precise. We then briefly discuss the Hilbert series, which gives a compact way to encode the information of the Hilbert function, as well as the Hilbert polynomial, which in the one-parameter case encodes the information of the Hilbert function for high enough degrees. Finally, we conclude the section with a series of examples.

3.3.1 The Hilbert function

The Hilbert function of a module M encodes the dimensions of all vector spaces $M_{\mathbf{u}}$:

Definition 3.3.16. Let M be an $\mathbb{N}^r$-graded module over $S = \mathbb{K}[x_1, \ldots, x_r]$. The **Hilbert function** of M is the function $HF(M, \mathbf{u}) = \dim_{\mathbb{K}} M_{\mathbf{u}}$.

An easy way to visualize the Hilbert function is as an r-dimensional array, with the $\mathbf{u} = (u_1, \ldots, u_r)$ entry equal to the dimension of $M_{\mathbf{u}}$. Hence the Hilbert function of S is an r-dimensional array with a one in every position which has all indices non-negative, and zeroes elsewhere, and $S(-\mathbf{u})$ is the same array, but with the origin translated to position $\mathbf{u}$.

Lemma 3.3.17. For a finitely generated graded $S = \mathbb{K}[x_1, \ldots, x_r]$-module M with free resolution $F_\bullet$ as in Definition 3.2.7,

(i) $HF(M, \mathbf{u}) = \sum_{i \geq 0} (-1)^i HF(F_i, \mathbf{u})$

(ii) $\mathrm{rk}_S(M) = \sum_{i \geq 0} (-1)^i \,\mathrm{rk}_S(F_i)$.

Proof. For any degree $\mathbf{u}$, the degree $\mathbf{u}$ components of $F_\bullet$ are an exact sequence of vector spaces over $\mathbb{K}$. The equality in (i) now follows from the Rank-Nullity Theorem from linear algebra by induction.

Let Q be the field of fraction of S. Applying $_ \otimes_S Q$ is the same as localizing at the zero ideal. Since localisation preserves exactness, the equality in (ii) now follows similarly as for (i) from an Euler characteristic argument. □

Lemma 3.3.18. For a finitely generated $\mathbb{N}^r$-graded S-module M,

$$\mathrm{rk}_S(M) = HF(M, \mathbf{u}) \text{ for } \mathbf{u} \gg \mathbf{0}\,.$$

Proof. A finitely generated, $\mathbb{N}^r$-graded free module F can be written as a finite sum

$$F = \bigoplus_{j=1}^{n} S(-\mathbf{u}_j), \text{ with } \mathbf{u}_j \in \mathbb{N}^r.$$

The multigraded module S is one-dimensional in each multidegree. Hence $HF(S(-\mathbf{u}_j), \mathbf{v}) = 1$ whenever $\mathbf{v} \geq \mathbf{u}_j$. So when $\mathbf{v} \geq \mathbf{u}_j$ for $j = 1, \ldots, n$,

$$HF(F, \mathbf{v}) = n = \mathrm{rk}_S(F).$$

We can now apply Lemma 3.3.17 to prove the statement for any finitely generated S-module M. Let $F_\bullet$ be a free resolution of M as in Definition 3.2.7. Then, for large enough $\mathbf{v}$,

$$HF(M, \mathbf{v}) = \sum_{i \geq 0} (-1)^i HF(F_i, \mathbf{v}) \; = \sum_{i \geq 0} (-1)^i \,\mathrm{rk}_S(F_i) = \mathrm{rk}_S(M),$$

where the first equality holds by item (i) in Lemma 3.3.17, and the third holds by item (ii) in that lemma. □

We thus have that $\mathrm{rk}_S(M) = \dim_{\mathbb{K}}(M)_{\mathbf{u}}$ for high enough degree $\mathbf{u}$. In particular, by Theorem 3.2.11 we know that for any $i > 0$ there exists a multifiltered simplicial complex K that stabilises at some degree $\mathbf{s}$ and such that $M \cong H_i(K)$; hence we have $\mathrm{rk}_S(M) = \dim_{\mathbb{K}}(H_i(K_{\mathbf{s}}))$. One can thus reduce the computation of the rank of a multigraded persistence module of a finite multifiltered simplicial complex K to the computation of the simplicial homology of the (unfiltered) complex K.

3.3.2 The Hilbert polynomial and Hilbert series

In the one-parameter case, or more generally when all variables have degree 1, one has that in degree high enough the Hilbert function equals a polynomial:

Lemma 3.3.19. [82, Corollary 1.3] Let M be a finitely generated $\mathbb{N}$-graded module over $S = \mathbb{K}[x_1, \ldots, x_r]$, with free resolution $F_\bullet$ as in Definition 3.2.7, where each module F_i is $\mathbb{N}$-graded. Then if we write $F_i = \oplus_j S(-a_{ij})$, we have that there exists a polynomial $HP(M, u)$ of degree at most $r-1$ in the variable u such that $HP(M, u) = HF(M, u)$ for all $u \geq \max_j\{a_{ij}\} - (r-1)$.

Definition 3.3.20. The polynomial $HP(M,t)$ is called **Hilbert polynomial** of M.

Given an $\mathbb{N}^r$-graded module M we can make it into an $\mathbb{N}$-graded module $\mathbf{M}$ by setting the degree of every variable to 1, and we can then consider the Hilbert polynomial of this module. The Hilbert polynomial of a module with non-zero rank has degree $\leq r-1$, and if M is a finitely generated $\mathbb{N}$-graded $\mathbb{K}[x]$-module , then [182]

$$HP(M,t) = \frac{rk(M)t^{r-1}}{(r-1)!} + \text{terms of lower degree.}$$

Similarly, the rank of an $\mathbb{N}^r$-graded module M is obtained from the leading coefficient of the Hilbert polynomial of $\mathbf{M}$, see [182, Page 41]. We note that the existence of the Hilbert polynomial relies on the variables having degree 1; see [82, Exercise 1C.2] for an example of how the statement of Lemma 3.3.19 is no longer valid if one changes the grading.

One can encode the information given by the Hilbert function in a compact way using a power series, called Hilbert series:

Definition 3.3.21. Let M be a finitely generated $\mathbb{N}^r$-graded module over $S = \mathbb{K}[x_1, \ldots, x_r]$. The **multigraded Hilbert series** of M is the formal power series in $\mathbb{Z}[[t_1, \ldots, t_r]]$ defined as follows:

$$HS(M,\mathbf{t}) = \sum_{\mathbf{u}\in\mathbb{N}^r} HF(M,\mathbf{u})\mathbf{t}^{\mathbf{u}}\,.$$

An easy calculation shows that

$$HS(S(-\mathbf{u}),\mathbf{t}) = \frac{\mathbf{t}^{\mathbf{u}}}{\prod_{i=1}^{r}(1-t_i)}\,. \tag{3.3}$$

By the Hilbert Syzygy Theorem a finitely generated, $\mathbb{N}^r$-graded S-module M has a free resolution of length at most r, and all the free modules F_i which appear in the resolution are graded and of finite rank. Since the Hilbert series is additive [81, Exercise 19.15], one can compute the multigraded Hilbert series of M from a free resolution by induction. We obtain:

$$HS(M,\mathbf{t}) = \sum_{i=0}^{r}(-1)^i HS(F_i,\mathbf{t})\,. \tag{3.4}$$

3.3.3 Examples

We illustrate our work with several examples. We provide `Macaulay2` code that we used to do these computations in Section C.2.

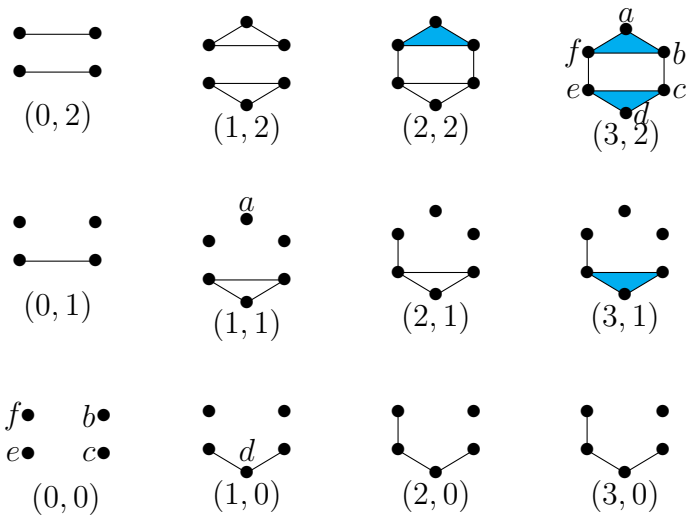


Figure 3.1: A one-critical bifiltration, given as example in [37].

Example 3.3.22. In [37], Carlsson, Singh and Zomorodian analyse the simplicial homology of the one-critical bifiltration in Fig. 3.1.

If the multifiltration is one-critical the modules in the simplicial chain complex are free, and thus one can write the differentials in the chain complex using monomial matrices (see [155, Section 1.4]). In this example, the differentials in the simplicial chain complex are given by

$$d_1 = \begin{bmatrix} x_2 & x_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -x_1x_2^2 & 0 & x_1^2x_2^2 & x_2^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -x_1^2x_2^2 & 0 & x_1 & x_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -x_2 & -x_1 & x_1^2 \\ 0 & -x_1x_2^2 & 0 & -x_2^2 & 0 & 0 & 0 & -x_1^2 \end{bmatrix}, \text{ and}$$

$$d_2 = \begin{bmatrix} -x_1 & 0 \\ x_1 & 0 \\ 0 & 0 \\ -x_1^2 & 0 \\ 0 & -x_1^2x_2 \\ 0 & x_1^3 \\ 0 & -x_1^2x_2 \\ 0 & 0 \end{bmatrix},$$

where the bases of 0, 1 and 2-simplices are ordered lexicographically. Using `Macaulay2` (see Section C.1), we compute the following minimal presentation of the first homology

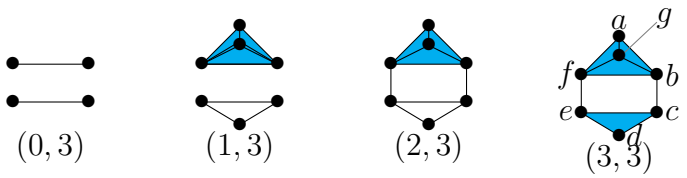


Figure 3.2: A one-critical bifiltration obtained from that in Fig. 3.1 by adding an interior vertex g to abf in degree $(1, 3)$ and triangulating. We note that the 2-simplex abf is not present in degree $(1, 3)$, but it is present in degrees $(2, 3)$ and $(3, 3)$. Here we only show the simplices in degrees greater or equal to $(0, 3)$, since those in degrees smaller or equal to $(3, 2)$ are the same as those for the bifiltration in Fig. 3.1.

of K:

$$S(-3,-1) \oplus S(-2,-2) \xrightarrow{\begin{bmatrix} x_1^2 & 0 \\ 0 & x_1 \\ 0 & 0 \end{bmatrix}} S(-1,-1) \oplus S(-1,-2) \oplus S(-2,-2)\,.$$

In particular, we see that $H_1(K)$ has three generators, and there are two relations, which do not interact with each other, that is, there is no non-trivial 2nd syzygy between them. Thus

$$H_1(K) \cong S(-2,-2) \oplus S(-1,-1)/x_1^2 \oplus S(-1,-2)/x_1\,.$$

Example 3.3.23. We next illustrate several concepts introduced in this section. We modify the bifiltration K in Example 3.3.22, by adding an interior vertex g to abf in degree $(1, 3)$ and triangulating, so that we obtain the bifiltered complex K' which in degree smaller than $(3, 2)$ is as K, and for degrees greater or equal to $(0, 3)$ is illustrated in Fig. 3.2.

The filtration K' is still one-critical, and we have that

$$H_1(K') \cong S(-2,-2) \oplus S(-1,-1)/x_1^2 \oplus S(-1,-2)/\langle x_1, x_2\rangle\,.$$

We note that here we have an example of a module with a generator that "dies" in two different degrees. We will introduce birth and death for elements of multiparameter persistence modules in Definition 3.4.38. Furthermore, this gives an example of a 1-cycle that is the boundary of two different 2-chains; see also Example 3.3.25 for another example along these lines. Note also that, in contrast to the bifiltered complex K from Example 3.3.22, the bifiltered complex K' has non-trivial homology in degree 2.

From the exact sequences

$$0 \longrightarrow S(-2,0) \xrightarrow{\cdot x_1^2} S \longrightarrow S/x_1^2 \longrightarrow 0$$

$$0 \longrightarrow S(-1,-1) \longrightarrow S(-1,0) \oplus S(0,-1) \xrightarrow{[x_1,x_2]} S \longrightarrow S/\langle x_1, x_2\rangle \longrightarrow 0$$

we see that

$$\begin{aligned} HS(S(-2,-2),\mathbf{t}) &= \frac{t_1^2 t_2^2}{(1-t_1)(1-t_2)} \\ HS(S(-1,-1)/x_1^2,\mathbf{t}) &= \frac{t_1 t_2(1-t_1^2)}{(1-t_1)(1-t_2)} \\ HS(S(-1,-2)/\langle x_1,x_2\rangle,\mathbf{t}) &= \frac{t_1 t_2^2(1-t_1-t_2+t_1t_2)}{(1-t_1)(1-t_2)}. \end{aligned}$$

Summing these yields the multigraded Hilbert series

$$\frac{t_1^2 t_2^2}{(1-t_1)(1-t_2)} + \frac{t_1 t_2(1+t_1)}{(1-t_2)} + t_1 t_2^2\,.$$

Example 3.3.24. Let $S = \mathbb{K}[x]$, and let $M = S(-2) \oplus S(-3) \oplus S(-1)/x^2 \oplus S/x$. We have:

$$HS(M,t) = \frac{1+t^2}{1-t}\,.$$

Example 3.3.25. We give another example of a one-critical bifiltration in which a 1-cycle is the boundary of two different 2-chains with non-comparable (with respect to the partial order on $\mathbb{N}^2$) entry degrees.

Consider the bifiltration of a triangulation of the 2-sphere, such that the equator (obtained by gluing together the one-simplices $\{a,b\}$, $\{b,c\}$ and $\{a,c\}$) has entry degree $(0,0)$, the southern hemisphere (the 2-simplex $\{a,b,c\}$) has entry degree $(2,0)$, the northern hemisphere (obtained by gluing together the 2-simplices $\{a,b,d\}$, $\{b,c,d\}$ and $\{a,c,d\}$) is completed at $(1,2)$, and the whole sphere is completed at $(2,2)$, as illustrated in Fig. 3.3.

Here we have that the generator ω of $\mathrm{H}_1(K)$ is in degree $(0,0)$, while there are γ and γ' in $C_2(K)$ in degrees $(2,0)$ and $(1,2)$, respectively, with $d_2(\gamma) = x_1^2\omega$ and $d_2(\gamma') = x_1x_2^2\omega$. So $H_1(K) = S/\langle x_1^2, x_1x_2^2\rangle$, which has a free resolution

$$0 \to S(-2,-2) \longrightarrow S(-2,0) \oplus S(-1,-2) \longrightarrow S \longrightarrow H_1(K) \longrightarrow 0,$$

so the Hilbert series is

$$\frac{1-t_1^2-t_1t_2^2+t_1^2t_2^2}{(1-t_1)(1-t_2)}\,.$$

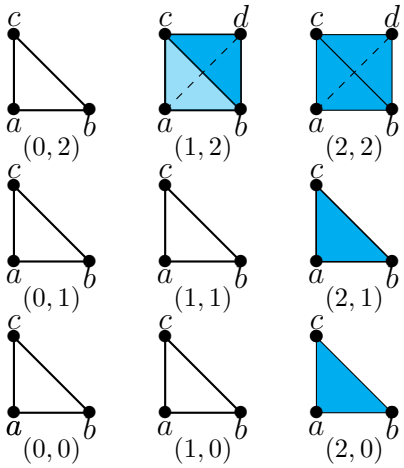


Figure 3.3: A one-critical bifiltration with a one-cycle that is the boundary of two different 2-chains with non-comparable entry degrees.

Example 3.3.26. In Fig. 3.4 we give two examples of 2-filtered simplicial complexes which are not one-critical. We compute the minimal presentation of the homology in degree 1 of these multifiltered simplicial complexes using the procedure given in [40]. The first homology M_A of the 2-filtration in Fig. 3.4(A) has minimal presentation

$$0 \to S(-1,0) \oplus S(0,-1) \to S$$

while the first homology M_B of the 2-filtration in Fig. 3.4(B) has minimal presentation:

$$0 \to S(-1,-1) \oplus S(-1,-1) \to S(-1,0) \oplus S(0,-1)\,.$$

From these we compute the multigraded Hilbert series: we have $M_A = S/\langle x_1, x_2\rangle$, and its multigraded Hilbert series is 1. We have $M_B = S(-1,0)/x_2 \oplus S(0,-1)/x_1$, and the multigraded Hilbert series of M_B is

$$\frac{t_1 + t_2 - 2t_1t_2}{(1-t_1)(1-t_2)}\,.$$

Example 3.3.27. Here we give an example of two non-isomorphic modules that have the same multigraded Hilbert series. Consider the $\mathbb{N}^2$-graded $S = \mathbb{K}[x_1, x_2]$ modules

$$\begin{aligned} M &= S(-1,-1) \oplus S(-2,-2) \\ N &= S(-2,-2) \oplus S(-2,-2) \oplus S(-2,-1)/x_2 \oplus S(-1,-2)/x_1 \oplus S(-1,-1)/\langle x_1, x_2\rangle \end{aligned}$$

Note that M and N are not isomorphic because one has torsion and the other doesn't. The Hilbert series of the modules agree, and are as follows:

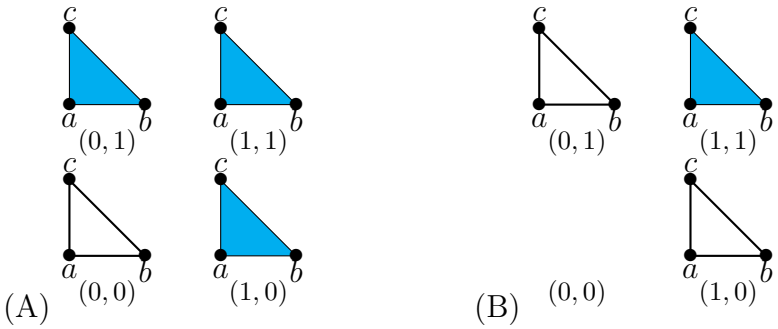


Figure 3.4: Two filtrations which are not one-critical.

$$\frac{t_1t_2 + t_1^2t_2^2}{(1-t_1)(1-t_2)} = \frac{2t_1^2t_2^2}{(1-t_1)(1-t_2)} + \frac{t_1}{(1-t_2)} + \frac{t_2}{(1-t_1)} - t_1 - t_2 - t_1t_2. \tag{3.5}$$

We note that already for 1-parameter PH the Hilbert series may not distinguish between different modules.

3.4 Associated primes and stratification

Our goal in this section is to extract finer information from multiparameter persistent homology. To do this, we study the associated primes of an $\mathbb{N}^r$-graded S-module M. From a geometric perspective, the minimal associated primes of M give information about the points in the grid $\mathbb{N}^r$ where the module does not vanish, while the embedded (i.e., non minimal) associated primes provide a way to stratify such support into subsets sorted by the dimension of the prime. Using this geometric intuition, in Definition 3.4.38 we give a generalisation of the definition of death and birth of elements of an S-module.

3.4.1 Associated primes of multigraded modules

In this subsection we give a brief overview of the theory of associated primes of $\mathbb{N}^r$-graded modules. We outline the main notions and give proofs for convenience, since the theory in the multigraded case is less well-known than the theory in the $\mathbb{N}$-graded case (i.e., when all variables have degree 1). We make no claim of originality.

Definition 3.4.28. Let R be a commutative ring, and let M be an R-module. Let U be a non-empty subset of M. Define the **annihilator of U** as follows:

$$\mathrm{Ann}(U) = \{r \in R \mid \forall u \in U : ru = 0\}.$$

We say that a prime ideal $\mathfrak{p} \subset R$ is **associated to M** if $\mathfrak{p}$ is the annihilator of an element of M. We denote by $\mathrm{Ass}(M)$ the poset of all primes associated to M, with partial order

given by inclusion. The minimal elements of $\mathrm{Ass}(M)$ are called **minimal**. Associated primes that are not minimal are **embedded**.

Associated primes of graded modules satisfy an additional property, namely they are homogeneous.

Definition 3.4.29. Let R be an $\mathbb{N}^r$-graded ring, and let $\mathfrak{p} \subset R$ be an ideal. We say that $\mathfrak{p}$ is **homogeneous** if it is generated by homogeneous elements.

Let a be an element of an $\mathbb{N}^r$-graded ring R or $\mathbb{N}^r$-graded module M. Then we can write

$$a = \sum_{\mathbf{u} \in \mathbb{N}^r} h_{\mathbf{u}}, \tag{3.6}$$

where for all $\mathbf{u}$ the element $h_{\mathbf{u}}$ is homogeneous of degree $\mathbf{u}$. Clearly the elements $h_{\mathbf{u}}$ are uniquely determined by a.

Definition 3.4.30. Let a be an element of an $\mathbb{N}^r$-graded ring R or $\mathbb{N}^r$-graded module M. The non-zero homogeneous elements $h_{\mathbf{u}_1}, \ldots, h_{\mathbf{u}_n}$ in (3.6) such that $a = \sum_{i=1}^n h_{\mathbf{u}_i}$ are called **homogeneous components of a**.

There is a very useful characterisation of homogeneous ideals in terms of homogeneous components:

Lemma 3.4.31. An ideal of an $\mathbb{N}^r$-graded ring is homogeneous if and only if it contains the homogeneous components of its elements.

Proof. Let R be an $\mathbb{N}^r$-graded ring, and let $I \subset R$ be a homogeneous ideal generated by a set of homogeneous elements $\{h_j\}_{j \in J}$. Let $a \in I$, and let $a = \sum_{i=1}^n r_i$ where $r_i \in R$ are the homogeneous components of a. Since a is in $I = \langle h_j \rangle_{j \in J}$, we have $a = \sum_{j \in J' \subset J} c_j h_j$ for some $c_j \in R$. Since a can be written uniquely as a sum of homogeneous elements of R, we must have that r_i is the sum of some of the homogeneous summands of c_j times h_j for some of the $j \in J$. Therefore we have $r_i \in I$ for all i.

For the converse, suppose that I is an ideal in R which contains the homogeneous components of its elements. Then I is generated by the homogeneous components of its elements, and is therefore homogeneous. □

We can now state the following result about associated primes for modules over an arbitrary commutative ring R [27, IV.3].

Proposition 3.4.32. Let M be an $\mathbb{N}^r$-graded module over an $\mathbb{N}^r$-graded ring R. Any associated prime $\mathfrak{p}$ of M is homogeneous.

For modules over $S = \mathbb{K}[x_1, \ldots, x_r]$ the $\mathbb{N}^r$-grading imposes a strong condition on associated primes, which turn out to have a very simple form:

Lemma 3.4.33. For a finitely generated $\mathbb{N}^r$-graded S-module M, any homogeneous prime $\mathfrak{p}$ of M is of the form $\mathfrak{p} = \langle x_{i_1}, \ldots, x_{i_k} \rangle$.

Proof. Let $\mathfrak{p} \subset S$ be an associated prime of M; by Proposition 3.4.32 $\mathfrak{p}$ is homogeneous. Now let $p \in \mathfrak{p}$, and write $p = \sum_{i=1}^{n} p_{\mathbf{u}_i}$ where the $p_{\mathbf{u}_i}$ are the homogeneous components of p. Since $\mathfrak{p}$ is homogeneous, we know that $p_{\mathbf{u}_i} \in \mathfrak{p}$ for all i. Furthermore, since S is $\mathbb{N}^r$-graded, we have that $p_{\mathbf{u}_i} = a\mathbf{x}^{\mathbf{u}_i}$ for some $a \in \mathbb{K}$, and therefore $\mathfrak{p} \subseteq \langle x_{i_1}, \ldots, x_{i_k} \rangle$, where the indices $i_1, \ldots, i_k$ are determined by the non-zero entries in the degrees of the homogeneous components of the elements of $\mathfrak{p}$. Furthermore, since $\mathfrak{p}$ is prime, we have that $x_j \in \mathfrak{p}$ whenever the jth entry u_{ij} of $\mathbf{u}_i$ is non-zero. Therefore $\langle x_{i_1}, \ldots, x_{i_r} \rangle \subseteq \mathfrak{p}$. □

Corollary 3.4.34. For a finitely generated $\mathbb{N}^r$-graded S-module M, any associated prime $\mathfrak{p}$ of M is of the form $\mathfrak{p} = \langle x_{i_1}, \ldots, x_{i_k} \rangle$.

Proof. Any associated prime is homogeneous by Proposition 3.4.32, and homogeneous primes are generated by a subset of variables by Lemma 3.4.33. □

3.4.2 Minimal associated primes and support shape

We now use the fact that associated primes of the modules that we are interested in have the form of Corollary 3.4.34 to define an invariant which only depends on the minimal associated primes.

Definition 3.4.35. Let M be a finitely generated $\mathbb{N}^r$-graded S-module, and let $\mathfrak{p} = \langle x_{i_1}, \ldots, x_{i_k} \rangle$ be an associated prime of M. The **complement support** of $\mathfrak{p}$ is defined as follows:

$$c_{\mathfrak{p}} = \{(u_1, \ldots, u_r) \in \mathbb{N}^r \mid u_i = 0 \text{ for all } i \in \{i_1, \ldots, i_k\}\} .$$

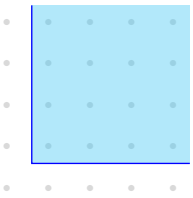
The **support shape** of M is the subset $ss(M)$ of $\mathbb{N}^r$ defined as follows:

$$ss(M) = \bigcup_{\mathfrak{p} \in \mathrm{Ass}(M)} c_{\mathfrak{p}}.$$

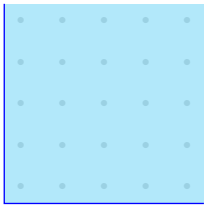
We note that the support shape of the module is completely determined by the minimal elements of $\mathrm{Ass}(M)$. We give examples of support shapes for the modules in Example 3.3.27 in Fig. 3.5(A).

The support shape of a module M encodes the points $\mathbf{u}$ of $\mathbb{N}^r$ such that $M_{\mathbf{u}} \neq 0$ only up to translation (i.e., it forgets the degree of the generators), thickness (i.e., it forgets the

(A)

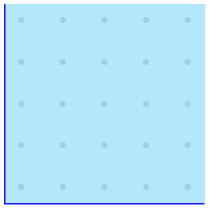


(a) The points of $\mathbb{N}^r$ at which the module M, as well as N, does not vanish.



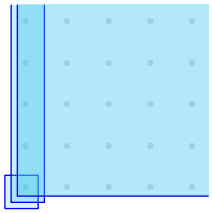
(b) The support shape of M, as well as N.

(B)

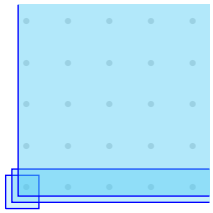


(c) The stratification of $ss(M)$.

(C)



(d) The chain in the stratification of $ss(N)$ corresponding to the increasing sequence of associated primes $(0) \subset (x_1) \subset (x_1, x_2)$.



(e) The chain in the stratification of $ss(N)$ corresponding to the increasing sequence of associated primes $(0) \subset (x_2) \subset (x_1, x_2)$.

Figure 3.5: Examples of stratifications of modules. We use grids with gray dots to represent the poset $\mathbb{N}^2$ and we represent subsets of $\mathbb{N}^2$ by cyan regions. We illustrate examples of support shapes and their stratifications for the two modules $M = S(-1,-1) \oplus S(-2,-2)$ and

$$N = S(-2,-2) \oplus S(-2,-2) \oplus \tfrac{S(-2,-1)}{x_2} \oplus \tfrac{S(-1,-2)}{x_1} \oplus \tfrac{S(-1,-1)}{\langle x_1, x_2 \rangle}$$

from Example 3.3.27. We have $\mathrm{Ass}(M) = \{(0)\}$ and $\mathrm{Ass}(N) = \{(0), (x_1), (x_2), (x_1, x_2)\}$. In (A) we illustrate the points in $\mathbb{N}^2$ at which the modules do not vanish as well as their support shapes. We note that the support shape is the same for M and N. In (B) and (C) we illustrate the stratifications for the modules. Since M has only one associated prime, the stratification of the support shape is the support shape itself.

degrees of the generators of the annihilators) and multiplicity (i.e., it forgets the number of generators of the module). We illustrate these notions with some examples in Fig. 3.6. In other words, the support shape gives information about what we get by projecting the points $\mathbf{u}$ of $\mathbb{N}^r$ such that $M_{\mathbf{u}} \neq 0$ to coordinate subspaces; a $c_{\mathfrak{p}}$ is in the support shape if the projection is fully dimensional. Thus, the support shape encodes in which directions elements of the module live, without giving more refined information about the degrees of the elements or the multiplicity. With local cohomology we can catch a shadow of this information, as we explain in Section 3.4.4.

3.4.3 Embedded associated primes and stratification of the support shape

We now use embedded (i.e., non minimal) associated primes to give a stratification of the support shape of the module. Namely, by Corollary 3.4.34 we know that the associated primes of a module over the polynomial ring $\mathbb{K}[x_1, \ldots, x_r]$ are of the form $\langle x_{i_1}, \ldots, x_{i_k} \rangle$. The **dimension** of such an ideal is defined to be $r - k$ [81, Chapter 9]. Therefore, given a nested sequence of associated primes

$$\mathfrak{p}_0 \subseteq \mathfrak{p}_1 \subseteq \cdots \subseteq \mathfrak{p}_m$$

of dimension $d_0 \geq d_1 \geq \cdots \geq d_m$, we obtain the following nested sequence of subsets of the support shape of M:

$$c_{\mathfrak{p}_m} \subseteq \cdots \subseteq c_{\mathfrak{p}_1} \subseteq c_{\mathfrak{p}_0} \subseteq ss(M)\,. \tag{3.7}$$

This motivates the following definition:

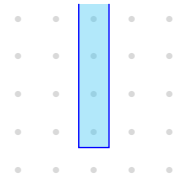
Definition 3.4.36. We call the poset $\mathrm{Ass}(M)$ the **stratification** of the support shape of M.

In Fig. 3.5(B)–(C) we give examples of stratifications for support shapes.

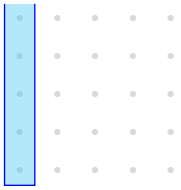
Remark 3.4.37. In algebraic geometry, given an ideal $I \subset S$, one defines the vanishing set of I as the subset $V(I) \subset \mathbb{K}^r$ given by the points of $\mathbb{K}^r$ at which the evaluation of all polynomials in I vanishes. The sets $V(I)$ are closed in the Zariski topology. For example, for $r = 2$ we have that $V((x_1)) = \{(0, a) \in \mathbb{K}^2 \mid a \in \mathbb{K}\}$. In our case we are not interested in the whole affine space $\mathbb{K}^r$, but rather in the non-negative grid $\mathbb{N}^r$, and we therefore give the definition of complement of a prime ideal in Definition 3.4.35, which is motivated by vanishing sets.

When working in the affine space $\mathbb{K}^r$, one could define the support shape of a finitely generated module M over S as the union over the vanishing sets of the associated primes

(A)

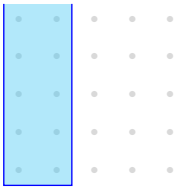


(a) The points of $\mathbb{N}^2$ at which the module $M_1 = S(-1,-2)/x_1$ does not vanish.

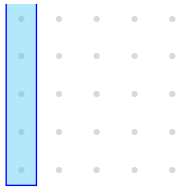


(b) The support shape of M_1.

(B)

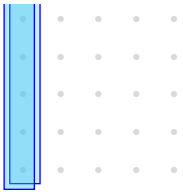


(c) The points of $\mathbb{N}^2$ at which the module $M_2 = S/(x_1^2)$ does not vanish.

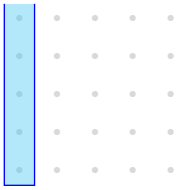


(d) The support shape of M_2.

(C)



(e) The points of $\mathbb{N}^2$ at which the module $M_3 = S/x_1 \oplus S/x_1$ does not vanish.



(f) The support shape of M_3.

Figure 3.6: Example of information forgotten by the associated primes. We illustrate how the support shape of an $S = \mathbb{K}[x_1, x_2]$-module M encodes the points $\mathbf{u}$ of $\mathbb{N}^r$ such that $M_\mathbf{u} \neq 0$ only up to (A) translation (i.e., it forgets the degrees of the generators), (B) thickness (i.e., it forgets the degrees of the minimal generators of the annihilators) and (C) multiplicity (i.e., it forgets the number of generators of the module).

of the module. The set $ss(M)$ is then a closed subset of affine space. Further, given a nested sequence of associated primes $\mathfrak{p}_0 \subseteq \mathfrak{p}_1 \subseteq \cdots \subseteq \mathfrak{p}_m$ of M, one obtains a nested sequence of affine algebraic varieties

$$V(\mathfrak{p}_m) \subseteq \cdots \subseteq V(\mathfrak{p}_1) \subseteq V(\mathfrak{p}_0) \subseteq ss(M)\,.$$

This is the motivation behind our use of the term "stratification" in Definition 3.4.36.

We next generalise the definition of birth and death of generators as can, e.g., be found in [165]:

Definition 3.4.38. Let M be a finitely generated $\mathbb{N}^r$-graded module over $\mathbb{K}[x_1, \ldots, x_r]$. We say that a homogeneous element a of M is **born at $(u_1, \ldots, u_r) \in \mathbb{N}^r$** if the degree of a is $(u_1, \ldots, u_r)$ and a is not in the image of any sum of maps $\sum_{\mathbf{v}} x^{\mathbf{v}}$ for any $\mathbf{v} \prec \mathbf{u}$. If $\mathrm{Ann}(a) \neq (0)$ let $D \subset \mathbb{N}^r$ be the subset of $\mathbb{N}^r$ obtained from the set of degrees of the set of minimal generators of $\mathrm{Ann}(a)$ by adding to each degree the degree of a. Then we say that a **dies in degrees D**. If $\mathrm{Ann}(a) = (0)$ we say that a **lives forever**.

If $\sqrt{\mathrm{Ann}(a)} = \langle x_{i_1}, \ldots, x_{i_k} \rangle = \mathfrak{p}$, we say that **$a$ lives along $c_{\mathfrak{p}} \subset \mathbb{N}^r$**. In this case we say that a has **support dimension $r - k$**.

Note that an element has support dimension r if and only if it lives forever. Similarly as for the support shape, if a lives along $c_{\mathfrak{p}}$, then $c_{\mathfrak{p}}$ encodes information on where a is non-zero in the module M only up to translation and thickness.

Using Definition 3.4.38 we can read off from the stratification of the support shape whether there are elements of a certain support dimension in the module, and we make the following definition:

Definition 3.4.39. We call elements of support dimension 0 **transient components**, elements of support dimension $1 \leq d < r$ **partially persistent components**, and elements of support dimension r **fully persistent components**. A **persistent component** is either a partially persistent or a fully persistent component.

The rank of a module gives the minimal number of generators for the submodule of the module which is generated by the fully persistent components. On the other hand, we will give a measure of the size of the module of the partially persistent components living along $c_{\mathfrak{p}}$, for $\mathfrak{p}$ an associated prime of the module, in Section 3.4.4.

If M is an $\mathbb{N}$-graded module over $\mathbb{K}[x]$ then for every $m \in M$ its annihilator $\mathrm{Ann}(m)$ is either (0) or (x^n) for some $n > 0$. We thus have that Definition 3.4.38 is an equivalent formulation of the definition of birth and death of generators as is, e.g., given in [165], which is a correction of the standard definition that uses the so-called "elder rule", see [165, Remark 5] for more details.

Example 3.4.40. For the modules in Example 3.3.27 we obtain the following: the module M has two generators of support dimension 2, of which one is born at $(1, 1)$, while the other is born at $(2, 2)$. On the other hand, the module N has two generators of support dimension 2, both born at (2,2); it has two of dimension 1, of which one is born at $(2, 1)$ and dies at $(2, 2)$, while the other is born at $(1, 2)$ and dies at $(2, 2)$; finally, it has one

generator of support dimension 0, born at $(1,1)$, and which dies at $(1,2)$ and $(2,1)$. The generator of N which is born at $(1,1)$ lives along the subspace spanned by x_1, while the generator of dimension 1 born at $(1,2)$ lives along the subspace spanned by x_2.

Example 3.4.41. For $r > 1$ an element might die at more than one degree, as Example 3.3.25 shows: the generator ω of $H_1(K)$ is born at $(0,0)$, and its annihilator is $(x_1^2, x_1x_2^2)$, which has radical (x_1). Therefore ω dies at degrees $\{(2,0),(1,2)\}$, lives along c_{x_2} and has support dimension 1.

3.4.4 Persistent but not fully persistent components: the rank of M along a coordinate subspace

A first measure of the size of a module M is the rank, which gives a measure of the number of fully persistent components. In this section we analyse how to compute the size of the components of M which are persistent but not fully persistent.

First, we recall some additional algebra:

Definition 3.4.42. [81, Chapter 3.6] Let $S = \mathbb{K}[x_1, \ldots, x_r]$. For a finitely generated S-module M and I an ideal of S different from the zero ideal, the **zeroth local cohomology of M with respect to I** is the subset of M defined as follows:

$$H_I^0(M) = \{m \in M \mid I^n \cdot m = 0 \text{ for all } n \gg 0\}. \tag{3.8}$$

When M is a finitely generated $\mathbb{N}^r$-graded $S = \mathbb{K}[x_1, \ldots, x_r]$ module, Corollary 3.4.34 shows that any associated prime of M is generated by a subset of the variables.

Notation 3.4.43. To simplify the notation and without loss of generality, we assume for the remainder of this section that $\mathfrak{p} = \langle x_1, \ldots, x_k\rangle \subseteq S = \mathbb{K}[x_1, \ldots, x_r]$, and $S_{c_\mathfrak{p}} = \mathbb{K}[x_{k+1}, \ldots, x_r]$. Thanks to the inclusion $S_{c_\mathfrak{p}} \hookrightarrow S$, an S-module also has the structure of an $S_{c_\mathfrak{p}}$-module.

We first note the connection between Definition 3.4.42 and Definition 3.4.38.

Lemma 3.4.44. For a finitely generated $\mathbb{N}^r$-graded S-module M and an associated prime $\mathfrak{p}$ of M we have:

$$\bigcup_{\substack{\mathfrak{p}' \supseteq \mathfrak{p} \\ \mathfrak{p}' \in \mathrm{Ass}(M)}} \{a \in M \mid a \text{ lives along } c_{\mathfrak{p}'}\} \subseteq H_\mathfrak{p}^0(M).$$

Proof. Let $\mathfrak{p}'$ be an associated prime of M that contains $\mathfrak{p}$. By Definition 3.4.38, a lives along $c_{\mathfrak{p}'}$ if and only if $\sqrt{Ann(a)} = \mathfrak{p}'$. This implies that $\mathfrak{p}'^n \cdot a = 0$ for some n, and hence in particular that $\mathfrak{p}^n \cdot a = 0$, which is equivalent to $a \in H_\mathfrak{p}^0(M)$. □

The set $H^0_I(M)$ is easily seen to be a submodule of M. When M is finitely generated as an S-module, then so is $H^0_I(M)$. Furthermore we have:

Lemma 3.4.45. Let M be a finitely generated $\mathbb{N}^r$-graded S-module and let $\mathfrak{p}$ be an associated prime. Then $H^0_{\mathfrak{p}}(M)$ is a finitely generated $S_{c_{\mathfrak{p}}}$-module.

Proof. Since M is finitely generated over S and S is Noetherian, the submodule $H^0_{\mathfrak{p}}(M)$ is also finitely generated as an S-module [81, Proposition 1.4]. Let $\{a_1, \ldots, a_m\}$ be homogeneous generators of $H^0_{\mathfrak{p}}(M)$ over S. By definition of local cohomology, for each i, a_i is annihilated by $\mathfrak{p}^{n_i}$ for some $n_i \in \mathbb{N}$. Therefore for each $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, k\}$ there exist $d_{ij} \in \mathbb{N}$ such that $x_j^{d_{ij}} \cdot a_i = 0$. So the S-module generated by the a_i is the same (as a vector space) as the $S_{c_{\mathfrak{p}}}$-module generated by all $\mathbf{x}^{\boldsymbol{\alpha}} \cdot a_i$ such that $\alpha_j \leq d_{ij}$ for all $j \in \{1, \ldots, k\}$. □

It follows from Lemma 3.4.45 that $H^0_{\mathfrak{p}}(M)$ has finite rank as an $S_{c_{\mathfrak{p}}}$-module, and by Lemma 3.3.18 we can extract this rank from the Hilbert function of $H^0_{\mathfrak{p}}(M)$.

Definition 3.4.46. For a finitely generated $\mathbb{N}^r$-graded S-module M and an associated prime ideal $\mathfrak{p} = \langle x_1, \ldots, x_k \rangle$ of M we define the **$c_{\mathfrak{p}}$-rank** of M to be the rank of $H^0_{\mathfrak{p}}(M)$ as an $S_{c_{\mathfrak{p}}}$-module.

Intuitively, the $c_{\mathfrak{p}}$-rank of an $\mathbb{N}^r$-graded module M gives information about the size of the persistent components of M that live along $c_{\mathfrak{p}}$; it is thus a count of the ways that the module goes to infinity in the directions orthogonal to the coordinate subspace spanned by $x_1, \ldots, x_k$. In the following we make the previous statement precise:

Proposition 3.4.47. Let $\mathfrak{p}$ be an associated prime of M with dimension contained strictly between 0 and r. The module M has the structure of an $S_{c_{\mathfrak{p}}}$-module by restriction of scalars, and for a subset $U \subset M$ we denote by $\langle U \rangle$ the $S_{c_{\mathfrak{p}}}$-submodule of M generated by U. We have:

$$\mathrm{rk}_{S_{c_{\mathfrak{p}}}} \left(\left\langle \{a \in M \mid a \text{ lives along } c_{\mathfrak{p}}\} \right\rangle \right) = \mathrm{rk}_{S_{c_{\mathfrak{p}}}}(H^0_{\mathfrak{p}}(M)) \,.$$

Proof. By Lemma 3.4.44 we have that

$$\mathrm{rk}_{S_{c_{\mathfrak{p}}}} \left(\left\langle \{a \in M \mid a \text{ lives along } c_{\mathfrak{p}}\} \right\rangle \right) \leq \mathrm{rk}_{S_{c_{\mathfrak{p}}}}(H^0_{\mathfrak{p}}(M)) \,,$$

hence the claim follows once we show that the elements of $H^0_{\mathfrak{p}}(M)$ that do not live along $c_{\mathfrak{p}}$ do not contribute to the rank of $H^0_{\mathfrak{p}}(M)$ as an $S_{c_{\mathfrak{p}}}$-module.

Let $a \in H^0_{\mathfrak{p}}(M)$ be such that $\sqrt{\mathrm{Ann}(a)} \neq \mathfrak{p}$. Note that $a \in H^0_{\mathfrak{p}}(M)$ if and only if $\mathfrak{p}^n \cdot a = 0$, and this is equivalent to $\mathfrak{p} \subseteq \sqrt{\mathrm{Ann}(a)}$. If $\sqrt{\mathrm{Ann}(a)}$ is a prime, say $\mathfrak{p}'$, then

a lives along $c_{\mathfrak{p}'}$ with $\mathfrak{p}' \supset \mathfrak{p}$, and thus a does not contribute to the rank of $H^0_{\mathfrak{p}}(M)$ as an $S_{c_{\mathfrak{p}}}$-module, as it lives along a coordinate subspace of dimension strictly less than k. Now assume that $\sqrt{\mathrm{Ann}(a)}$ is not a prime. Then we can write $\sqrt{\mathrm{Ann}(a)} = \mathfrak{q}_1 \cap \cdots \cap \mathfrak{q}_s$, with $\mathfrak{q}_1, \ldots, \mathfrak{q}_s$ associated primes of $S/\sqrt{\mathrm{Ann}(a)}$. We must have that $\mathfrak{q}_i \neq \mathfrak{p}$, as otherwise $\sqrt{\mathrm{Ann}(a)} = \mathfrak{p}$. Hence $\mathfrak{p} \subset \mathfrak{q}_i$ for all $i = 1, \ldots, s$. Thus the support of a is a union of coordinate subspaces that have all dimension strictly less than that of $c_{\mathfrak{p}}$, and therefore a cannot contribute to the rank of $H^0_{\mathfrak{p}}(M)$ as an $S_{c_{\mathfrak{p}}}$-module. □

Thus, the $c_{\mathfrak{p}}$-rank of M gives the minimal number of generators of the $S_{c_{\mathfrak{p}}}$-submodule of M generated by the elements that live along $c_{\mathfrak{p}}$.

Example 3.4.48. Consider the modules $M_1 = S(-1,-2)/x_1$, $M_2 = S/x_1^2$, and $M_3 = S/x_1 \oplus S/x_1$ from Fig. 3.6. We have

$$\begin{aligned}
\mathrm{rk}_{S_{c_{\langle x_1 \rangle}}}(H^0_{\langle x_1 \rangle}(M_1)) &= 1 \\
\mathrm{rk}_{S_{c_{\langle x_1 \rangle}}}(H^0_{\langle x_1 \rangle}(M_2)) &= 2 \\
\mathrm{rk}_{S_{c_{\langle x_1 \rangle}}}(H^0_{\langle x_1 \rangle}(M_3)) &= 2\,.
\end{aligned}$$

3.4.5 Syzygies and the rank invariant

The invariants that we proposed in Section 3.4.3 give information about non-trivial syzygies: if a module has an associated prime of dimension $r - c$, then it also has non-trivial syzygies of order c (see Theorem 8.3.2 in [182]). However, the converse is not true in general, in Example 3.4.50 we have a module with non-trivial syzygies of order 1 that does not have associated primes of the corresponding dimension. Another invariant that fails to distinguish between the modules of Example 3.4.50 is the rank invariant, an invariant for multiparameter persistence modules introduced by Carlsson and Zomorodian in [38], which is equivalent to the barcode in the one-parameter case [38, Theorem 12]:

Definition 3.4.49. Let M be a multigraded module over $\mathbb{K}[x_1, \ldots, x_r]$. For a pair $\mathbf{u} \preccurlyeq \mathbf{v} \in \mathbb{N}^r$, the **rank invariant** $\rho_M(\mathbf{u}, \mathbf{v})$ is the rank of the map $M_{\mathbf{u}} \xrightarrow{\cdot x^{\mathbf{v}-\mathbf{u}}} M_{\mathbf{v}}$, so

$$\rho_M(\mathbf{u}, \mathbf{v}) = \dim_{\mathbb{K}} M_{\mathbf{u}} - \dim_{\mathbb{K}} \ker(\cdot x^{\mathbf{v}-\mathbf{u}})_{\mathbf{u}}\,.$$

In [140] Lesnick and Wright give an example of two non-isomorphic modules that have the same rank invariant:

Example 3.4.50. [140, Example 2.2] The $\mathbb{N}^2$-graded $S = \mathbb{K}[x_1, x_2]$ modules

$$\begin{aligned}
N &= S(-1,0) \oplus S(0,-1) \\
M &= (S(-1,0) \oplus S(0,-1) \oplus S(-1,-1))/\operatorname{im}(x_2, -x_1, 0)^t)
\end{aligned}$$

In M we have glued the two copies of S in N together where they overlap and added a free copy of S. These modules have the same rank invariant; they also have the same associated primes, namely just the zero ideal.

In the following we relate the rank invariant $\rho_M(\mathbf{u}, \mathbf{v})$ to the rank of a module:

Corollary 3.4.51. For $\mathbf{u} \gg \mathbf{0}$, the rank invariant $\rho_M(\mathbf{u}, \mathbf{v})$ is $\mathrm{rk}(M)$.

Proof. By Lemma 3.3.18 we have that $\mathrm{rk}(M) = \dim_{\mathbb{K}} M_{\mathbf{a}}$ for $\mathbf{a} \gg \mathbf{0}$; once $\mathbf{u}$ is in this range, the map from $M_{\mathbf{u}}$ to $M_{\mathbf{v}}$ is a square matrix with $x^{\mathbf{v}-\mathbf{u}}$ on the diagonal, so has full rank equal to $\dim_{\mathbb{K}} M_{\mathbf{u}}$. □

Proposition 3.4.52. If $\ker(\cdot x^{\mathbf{v}-\mathbf{u}})_{\mathbf{u}} \neq 0$ then for some x_i dividing $x^{\mathbf{v}-\mathbf{u}}$, $x_i \in P$ for some $P \in Ass(M)$.

Proof.

$$\begin{aligned} \ker(\cdot x^{\mathbf{v}-\mathbf{u}})_{\mathbf{u}} \neq 0 \quad &\Leftrightarrow \quad \exists 0 \neq m \in M_{\mathbf{u}} \text{ with } x^{\mathbf{v}-\mathbf{u}} \cdot m = 0 \\ &\Leftrightarrow \quad x^{\mathbf{v}-\mathbf{u}} \in \mathrm{Ann}(m) \\ &\Rightarrow \quad x^{\mathbf{v}-\mathbf{u}} \in P \in \mathrm{Ass}(M) \text{ for some } P \\ &\Leftrightarrow \quad x_i | x^{\mathbf{v}-\mathbf{u}} \text{ for some } x_i \in P \in \mathrm{Ass}(M). \end{aligned}$$

□

3.4.6 Computation of associated primes

Many of the results of this paper rely on the multigraded nature of multiparameter persistence modules. By making use of the fact that any finitely generated multiparameter persistence module $H_i(K)$ can be realised as the homology of a graded chain complex associated to the filtered complex K, we can also simplify the computation of the associated primes. In particular, it is possible to determine the associated primes of $H_i(K)$ without ever computing the kernel of the i-th differential d_i, which is the homomorphism of $\mathbb{N}^r$-graded modules introduced in Equation (3.1).

In the following we recall two facts about the associated primes: they behave well on short exact sequences, and they are closely related to the annihilator of M (see Theorem 3.1 and Lemma 3.6 of [81]):

Proposition 3.4.53. For a short exact sequence of finitely generated S-modules

$$0 \longrightarrow N \longrightarrow M \longrightarrow P \longrightarrow 0, \text{ we have}$$

1. $\mathrm{Ass}(N) \subseteq \mathrm{Ass}(M) \subseteq \mathrm{Ass}(N) \bigcup \mathrm{Ass}(P)$.
2. $\bigcup_{\mathfrak{p} \in \mathrm{Ass}(M)} \mathfrak{p} = \{s \in S \mid s \text{ is a zero divisor on M}\}$.

3. $\mathrm{Ass}(M)$ is a finite nonempty set of primes, each containing $\mathrm{Ann}(M)$, and includes all primes minimal over $\mathrm{Ann}(M)$.

Proposition 3.4.54. We have

$$\mathrm{Ass}(\mathrm{coker}(d_{i+1})) = \begin{cases} \mathrm{Ass}(H_i(K)) \cup \{(0)\}\,, & \text{if } H_i(K) \text{ has no torsion-free submodule} \\ & \text{but } \mathrm{coker}(d_{i+1}) \text{ does.} \\ \mathrm{Ass}(H_i(K))\,, & \text{otherwise.} \end{cases}$$

Proof. First, note that for any multifiltered simplicial complex there exists a one-critical multifiltered simplicial complex such that the respective persistent homology modules are isomorphic [37]. Given a one-critical multifiltration, the modules in its simplicial chain complex are free, and we thus make this assumption in the following. We have the short exact sequence of multigraded S-modules

$$0 \longrightarrow H_i(K) \longrightarrow C_i(K)/\operatorname{im}(d_{i+1}) \longrightarrow C_i(K)/\ker(d_i) \longrightarrow 0. \tag{3.9}$$

Furthermore, by the first isomorphism theorem for S-modules we have

$$C_i(K)/\ker(d_i) \cong \operatorname{im}(d_i) \subseteq C_{i-1}(K)\,,$$

with $C_{i-1}(K)$ a free module over the integral domain S, so by Proposition 3.4.53(1), the only associated prime for $\operatorname{im}(d_i)$ is the zero ideal; applying again Proposition 3.4.53(1) to Equation 3.9 yields the result. □

Thanks to Proposition 3.4.54, we can compute the associated primes of the ith homology $H_i(K) = \ker d_i/\operatorname{im} d_{i+1}$ by computing the associated primes of the cokernel of d_{i+1}.

3.5 From MPH to PH: restriction to a line and intersection with the diagonal

A natural line of approach to extract information from multiparameter persistent homology is to associate an $\mathbb{N}$-graded module over a polynomial ring in one variable to an $\mathbb{N}^r$-graded module M over $S = \mathbb{K}[x_1, \ldots, x_r]$, and then to study the barcode of the $\mathbb{N}$-graded module.

To our knowledge, the only approach to assign an $\mathbb{N}$-graded module to an $\mathbb{N}^r$-graded module that has been studied until now is the restriction of a module to a line; this approach was studied in [20, 140], and furthermore in [140] the authors introduce a tool for the visualisation of barcodes along such restrictions. To define such restrictions to

lines with non-negative real slope, it is necessary to work in the more general setting of $\mathbb{R}^r$-graded modules (see [140] for details). In this section we explain how to adapt this construction to the $\mathbb{N}^r$-grading, and we show that the rank of a module can be computed as the rank of the module restricted to a suitable line.

Finally, we discuss the algebraic version of restricting a module to a line, and show that one can read off the rank of a module from its intersection with the diagonal. We conclude the section with an example of an MPH module for which the restriction to the diagonal line yields a module that is not isomorphic to the module obtained by intersecting with the diagonal.

3.5.1 Restriction to a line

The restriction of a module to a line with non-negative slope was studied in [20, 140]. We adapt this to our setting, by associating an $\mathbb{N}$-graded module to an $\mathbb{N}^r$-graded module as follows. Given a tuple $(\mathbf{u}, \mathbf{u}_0) \in \mathbb{N}^r \times \mathbb{N}^r$, we define the following order-preserving map

$$l \colon \mathbb{N} \to \mathbb{N}^r \colon i \mapsto i\mathbf{u} + \mathbf{u_0}\,.$$

Therefore, thinking of the module M as a functor $M \colon \mathbb{N}^r \to \mathrm{Vect}_{\mathbb{K}}$ (see Remark 3.2.4), we can define $M_l \colon \mathbb{N} \to \mathrm{Vect}_{\mathbb{K}}$ as the following composition of functors:

$$M_l = M \circ l\,.$$

This associates an $\mathbb{N}$-graded module M_l on the polynomial ring $\mathbb{K}[x]$ to any $\mathbb{N}^r$-graded module M on $\mathbb{K}[x_1, \ldots, x_r]$, where the action of x on M_l is given by the action of $\mathbf{x}^{\mathbf{u}}$ on M.

Proposition 3.5.55. Let M be a finitely generated $\mathbb{N}^r$-graded S-module. Let $(\mathbf{u}, \mathbf{u}_0) \in \mathbb{N}^r \times \mathbb{N}^r$, and let l be the induced order-preserving map $\mathbb{N} \to \mathbb{N}^r$. If $u_i > 0$ for all i, we have:

$$\mathrm{rk}_S(M) = \mathrm{rk}_{\mathbb{K}[x]}(M \circ l)$$

where $x = \mathbf{x}^{\mathbf{u}}$.

Proof. Whenever $j >> 0$ we have that $\mathrm{rk}_{\mathbb{K}[x]}(M \circ l) = HF(M \circ l, j)$ by Lemma 3.3.18. Furthermore, $HF(M \circ l, j) = \dim_{\mathbb{K}}(M \circ l)_j$ by the definition of Hilbert function, and this last term equals $\dim_{\mathbb{K}}(M)_{\mathbf{u}j+\mathbf{u_0}}$. If $u_i > 0$ for all i, then $\dim_{\mathbb{K}}(M)_{\mathbf{u}j+\mathbf{u_0}} = \dim_{\mathbb{K}}(M)_{(v_1,\ldots,v_r)}$ with $v_i >> 0$. Again by Lemma 3.3.18, this last term equals $\mathrm{rk}_S(M)$.

□

More generally, given any order-preserving map $f \colon \mathbb{N} \to \mathbb{N}^r$, one can associate an $\mathbb{N}$-graded module $M_f := M \circ f$ to an $\mathbb{N}^r$-graded module M; thus one can probe the S-module M with any monotonically increasing path in $\mathbb{N}^r$ and associate to it a barcode.

3.5.2 Intersecting with the diagonal

In this subsection we discuss how one can intersect an $\mathbb{N}^r$-graded module with a specific line using methods from commutative algebra, and we show that the rank of the resulting module equals the rank of the original module.

The method of restricting a module that we briefly discussed in Section 3.5.1 reflects the intuitive method of probing the structure of a topological space $X \subseteq \mathbb{K}^r$ by intersecting X with a low dimensional linear space; for example, with a line. The algebraic version of intersecting with a geometric object $L = V(\mathfrak{p})$ is the tensor product

$$M|_L = M \otimes_S S/\mathfrak{p}\,.$$

We intersect an $\mathbb{N}^r$-graded S-module M with a line L through the origin in $\mathbb{K}^r$. For such a line, the defining ideal $\mathfrak{p} = \langle l_1, \ldots, l_{r-1}\rangle$ is generated by $r-1$ homogeneous (in the $\mathbb{N}$-grading) elements. Since

$$S/\mathfrak{p} \simeq \mathbb{K}[x],$$

this means that $M \otimes_S S/\mathfrak{p}$ is a $\mathbb{K}[x]$-module, so $M \otimes_S S/\mathfrak{p}$ has a barcode. We will now show that in case that the line L is the diagonal, the number of infinite intervals in the barcode of $M \otimes_S S/\mathfrak{p}$ is equal to $\mathrm{rk}_S(M)$.

Theorem 3.5.56. If M is a finitely generated $\mathbb{N}^r$-graded S-module with $r \geq 2$ and $\mathfrak{p} = \langle x_1 - x_2, \ldots, x_{r-1} - x_r\rangle$, then

$$\mathrm{rk}_S(M) = \mathrm{rk}_{S/\mathfrak{p}}(M \otimes_S S/\mathfrak{p}).$$

Proof. We proceed by induction on r, starting with the case $r = 2$, so that $\mathfrak{p} = \langle x_1 - x_2\rangle$ is principal. Tensoring the short exact sequence

$$0 \longrightarrow S(-1) \xrightarrow{x_1 - x_2} S \longrightarrow S/\mathfrak{p} \longrightarrow 0 \tag{3.10}$$

with M yields the exact sequence

$$0 \longrightarrow \ker(x_1 - x_2) \longrightarrow M(-1) \xrightarrow{x_1 - x_2} M \longrightarrow M \otimes S/\mathfrak{p} \longrightarrow 0 \tag{3.11}$$

Suppose $m \in \ker(x_1 - x_2)$, so that $x_1 m - x_2 m = 0$. If m has degree (a, b) then $x_1 m$ has degree $(a+1, b)$ and $x_2 m$ has degree $(a, b+1)$ so we must have $x_1 m = 0 = x_2 m$, in other words $\ker(x_1 - x_2)$ is annihilated by $\langle x_1, x_2\rangle$. In particular, it follows that $HP(\ker(x_1 - x_2), j) = 0$ for $j >> 0$.

Denote by $\mathbf{M}$ the $\mathbb{N}$-graded module obtained from M by setting the degree of every variable to be 1. By the discussion after Definition 3.3.20 we know that $HP(\mathbf{M}, i) = ci + d$ for i large enough, and with $c = \mathrm{rk}(M)$.

As the Hilbert polynomial is additive on exact sequences, we obtain, for $i \gg 0$:

$$\begin{aligned} HP(M \otimes S/\mathfrak{p}, i) &= HP(\mathbf{M}, i) - HP(\mathbf{M}, i-1) + HP(\ker(\cdot x_1 - x_2), i-1) \\ &= HP(\mathbf{M}, i) - HP(\mathbf{M}, i-1) \\ &= ci + d - (c(i-1) + d) \\ &= c\,. \end{aligned}$$

This establishes the theorem for $r = 2$.

Now suppose the assertion holds for $r = n-1$ and let $r = n$. This is where we use the fact that $\mathfrak{p}$ is generated by very special linear forms: an $\mathbb{N}^r$-graded module M, quotiented by $\langle x_i - x_j \rangle$ becomes an $\mathbb{N}^{r-1}$-graded module. This is not the case for an arbitrary linear form. For our $\mathbb{N}^r$-graded module M, we first proceed as above, tensoring it with the short exact sequence of Equation 3.10 and obtaining the four term exact sequence of Equation 3.11. While $\ker(x_1 - x_2)$ is annihilated by $\langle x_1, x_2 \rangle$, this does not force it to vanish in high degree. However, since it is torsion, it does not contribute to rank, hence arguing as above we find

$$\mathrm{rk}_S(M) = \mathrm{rk}_{S/\langle x_1 - x_2 \rangle}(M \otimes_S S/\langle x_1 - x_2 \rangle)$$

But $M \otimes S/\langle x_1 - x_2 \rangle$ is an $\mathbb{N}^{n-1}$-graded module, we may apply our induction hypothesis with $\mathfrak{p}' = \langle x_2 - x_3, \ldots x_{r-1} - x_r \rangle$ to obtain

$$\mathrm{rk}_{S/\langle x_1 - x_2 \rangle}(M \otimes_S S/\langle x_1 - x_2 \rangle) = \mathrm{rk}_{S/\mathfrak{p}}(M \otimes_S S/\langle x_1 - x_2 \rangle \otimes S/\mathfrak{p}') = \mathrm{rk}_{S/\mathfrak{p}}(M \otimes_S S/\mathfrak{p})\,.$$

□

While the two methods to pass from MPH to PH that we we have introduced in this section give modules that have the same rank as the original module, they do not give isomorphic modules in general:

Example 3.5.57. Let $M = S(-3, 0)/x_2$, and further let

$$\begin{aligned} l \colon \mathbb{N} &\to \mathbb{N}^2 \\ i &\mapsto (i, i)\,. \end{aligned}$$

Then $M \circ l$ is the zero module, while $M \otimes_S S/\langle x_1 - x_2 \rangle$ is isomorphic to the $\mathbb{K}[x]$-module $x^3\mathbb{K}$.

3.6 Conclusion

This paper studies multifiltered simplicial complexes K, and the associated multiparameter persistent homology modules $H_i(K)$ introduced by Carlsson and Zomorodian in [38]. We propose the rank of a module as an invariant for multiparameter persistence modules that captures fully persistent components, namely elements of the module living forever in all directions, and show that the rank of the module is the lead coefficient (suitably interpreted) of the Hilbert series of the MPH module. We show that one can compute the rank by computing the simplicial homology of the simplicial complex at which a multifiltration stabilises, and we furthermore give a geometric interpretation of the rank of the module as the rank of the one-parameter persistence module obtained by restricting the module to a linear subspace. We refine our invariant by studying the associated primes of the module, which give a stratification of the support shape of the module, and capture persistent components, that is, elements of the module living forever along some coordinate direction but not all. We provide a method to compute the size of the submodule generated by the persistent components using local cohomology, and we provide a shortcut to compute associated primes. Finally, we discuss several ways to associate a one-parameter persistence module to an r-parameter persistence module.

We provide the `Macaulay2` code that we have written for our work in Appendix C, with the appropriate documentation. We are currently implementing a code that takes as input an arbitrary multifiltration (not necessarily one-critical) using the presentation given in [40], as well as a way to compute the $c_{\mathfrak{p}}$-rank quickly.

Our work presents several interesting followup questions; a question that we are currently investigating is the stability of the stratification of the support shape of a module.

3.7 Additional acknowledgments

We were supported by The Alan Turing Institute through EPSRC grant EP/N510129/1, and by EPSRC Institutional Sponsorship 2016 award EP/P511377/1. We thank Bei Wang and Michael Lesnick for suggesting an improvement of Definition 3.4.38, and Parker Edwards for helpful comments on a draft of this paper.

Chapter 4

Magnitude meets persistence. Homology theories for filtered simplicial sets

The Euler characteristic is an invariant of a topological space that in a precise sense captures its canonical notion of size, akin to the cardinality of a set. The Euler characteristic is closely related to the homology of a space, as it can be expressed as the alternating sum of its Betti numbers, whenever the sum is well-defined. Thus, one says that homology **categorifies** the Euler characteristic. In his work on the generalisation of cardinality-like invariants, Leinster introduced the **magnitude** of a metric space, a real number that gives the "effective number of points" of the space. Recently, Leinster and Shulman introduced a homology theory for metric spaces, called **magnitude homology**, which categorifies the magnitude of a space. When studying a metric space, one is often only interested in the metric space up to a rescaling of the distance of the points by a non-negative real number. The **magnitude function** describes how the effective number of points changes as one scales the distance, and it is completely encoded in the Euler characteristic of magnitude homology. When studying a finite metric space in topological data analysis using **persistent homology**, one approximates the space through a nested sequence of simplicial complexes so as to recover topological information about the space by studying the homology of this sequence. Here we relate magnitude homology and persistent homology as two different ways of computing homology of filtered simplicial sets. Motivated by this general framework we make conjectures about an Euler characteristic for (multiparameter) persistent homology, and the **persistence function** of a metric space that it would encode.

4.1 Introduction

In a letter to Goldbach written in 1750, Euler [85] noted that for any polyhedron consisting of F regions, E edges and V vertices one obtains $V-E+F=2$. This sum is known as the **Euler characteristic** of the polyhedron. While one usually first encounters the Euler characteristic in relation to topological spaces, one can more generally define the Euler characteristic of an object in any symmetric monoidal category [150], and this can be thought of as its canonical size, a "dimensionless" measure. The irrelevance of topology for the notion of Euler characteristic, and how it should be thought of as an invariant giving a measure of the size or cardinality of an object was made precise among others by Schanuel [181].

In 1973 Lawvere [133] observed that every metric space is a category enriched over the monoidal category $[0,\infty]$ with objects non-negative real numbers, and a morphism $\epsilon \to \epsilon'$ whenever $\epsilon \geq \epsilon'$, with tensor product given by addition. Such enriched categories are called "Lawvere metric spaces", and a Lawvere metric space is the same thing as an extended quasi-pseudo-metric space.

In his work on the generalisation of the Euler characteristic as a cardinality-like invariant, Leinster [136] introduced an invariant for finite categories generalising work done by Rota on posets. The invariant introduced by Leinster generalises both the cardinality of a set, as well as the topological Euler characteristic. In subsequent work [137] Leinster generalised this invariant to enriched categories, calling it **magnitude**. The magnitude of a metric space is a real number that measures the "effective number of points" of the space. The **magnitude function** describes how the effective number of points changes as one scales the distances of the points of the metric space by a non-negative real number.

The Euler characteristic of a topological space X is closely related to the singular homology of a space, as it can be expressed as the alternating sum of its Betti numbers

$$\chi(X) = \sum_{i=0}^{\infty} (-1)^i \beta_i(X)\,,$$

whenever the sum and the summands are finite. One then says that homology **categorifies** the Euler characteristic. Thus, a natural question to ask is whether there is a homology theory for metric spaces that categorifies in an analogous way the magnitude. Leinster and Shulman [139] recently answered this question in the affirmative, by introducing **magnitude homology**, a homology theory for metric spaces as a special case of Hochschild homology for enriched categories. This homology theory encodes the magnitude function of a metric space.

Towards the end of [139] Leinster and Shulman list a series of open problems, the last two being as follows:

9. Magnitude homology only "notices" whether the triangle inequality is a strict equality or not. Is there a "blurred" version that notices "approximate equalities"?

10. Almost everyone who encounters both magnitude homology and persistent homology feels that there should be some relationship between them. What is it?

In this chapter we attempt a first answer to these questions, and show that they are intertwined. In [139] Leinster and Shulman define magnitude homology as the homology of the geometric realisation of a certain two-sided simplicial bar construction. In online discussions [138] they had previously considered a different definition, whose starting point was the generalisation of a nerve of a category to enriched categories. Motivated by the latter definition, we give an alternative definition of magnitude homology (Definition B' in Section 4.8.2), and show that it is equivalent to the definition given by Leinster and Shulman (Proposition 4.8.79). Specifically, let X be a Lawvere metric space. Its enriched nerve $N(X)$ is a functor $[0,\infty]^{\mathrm{op}} \to \mathrm{sSet}$, where for each $\epsilon \in [0,\infty]$ the simplicial set $N(X)(\epsilon)$ has set of n-simplices given by

$$N(X)(\epsilon)_n = \{(x_0, \dots, x_n) : \ d(x_0, x_1) + \cdots + d(x_{n-1}, x_n) \leq \epsilon\}\,.$$

We can then consider the composite functor

$$CN(X) = \left([0,\infty]^{\mathrm{op}} \xrightarrow{N(X)} \mathrm{sSet} \xrightarrow{\mathbb{Z}[\cdot]} \mathrm{sAb} \xrightarrow{U} \mathrm{ch}_{\mathrm{Ab}}\right)\,,$$

where the functor $\mathbb{Z}[\cdot]$ is induced by the free abelian group functor, and the functor U is the functor that sends a simplicial abelian group to its unnormalised chain complex. For any $\epsilon \in [0,\infty]$ we consider the coefficient functor $A_\epsilon \colon [0,\infty] \longrightarrow \mathrm{ch}_{\mathrm{Ab}}$ that sends ϵ to the chain complex with a copy of $\mathbb{Z}$ concentrated in degree zero, and all other numbers to the trivial chain complex. Let

$$CN(X) \otimes_{[0,\infty]} A_- \colon [0,\infty]^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$$

be the functor that sends a number ϵ to the coend $CN(X) \otimes_{[0,\infty]} A_\epsilon$. We can compose this functor with the functor $H_\star \colon \mathrm{ch}_{\mathrm{Ab}} \to \mathrm{Ab}$ that sends a chain complex to its homology. The magnitude homology of X (according to Definition B') is then the composition $H_\star(CN(X) \otimes_{[0,\infty]} A_-)$.

Persistent homology is, in an appropriate sense, the generalisation of simplicial homology of a simplicial set to filtered simplicial sets. Given a metric space (X, d), one

associates to it a filtered simplicial set $S(X)$ such that $S(X)(\epsilon)$ captures in a certain sense the topology of X "below distance ϵ". One then considers the functor

$$CS(X) = \left([0,\infty]^{\mathrm{op}} \xrightarrow{S(X)} \mathrm{sSet} \xrightarrow{\mathbb{Z}[\cdot]} \mathrm{sAb} \xrightarrow{U} ch_{\mathrm{Ab}}\right),$$

where $\mathbb{Z}[\cdot]$ and U are defined as above. Furthermore, one usually considers the functor of coefficients $A\colon [0,\infty]^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$ that sends every number to the chain complex with a copy of $\mathbb{F}$ concentrated in degree zero, where $\mathbb{F}$ is a field. One then considers the componentwise tensor product $CS(X) \boxtimes A$ of the functors $CS(X)$ and A, and defines **persistent homology of $S(X)$ with coefficients in** $\mathbb{F}$ to be the composition $H_\star(CS(X) \boxtimes A)$, where $H_\star\colon \mathrm{ch}_{\mathrm{Ab}} \to \mathrm{Ab}$ is the usual homology functor. More generally, we can consider arbitrary coefficient functors $A\colon [0,\infty]^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$.

Now, to relate persistent homology to magnitude homology we introduce for any $\epsilon \in [0,\infty]$ the coefficient functor

$$\begin{aligned} A_{[0,\epsilon]}\colon [0,\infty]^{\mathrm{op}} &\to \mathrm{Ab} \\ l &\mapsto \begin{cases} \mathbb{Z}, \text{if } l \in [0,\epsilon] \\ 0, \text{otherwise} \end{cases} \end{aligned}$$

which we consider as taking values in $\mathrm{ch}_{\mathrm{Ab}}$ through the canonical inclusion functor $\mathrm{Ab} \hookrightarrow \mathrm{ch}_{\mathrm{Ab}}$ that sends an abelian group to the chain complex with that abelian group concentrated in degree zero. We then define the functor $CN(X) \otimes_{[0,\infty]} A_{[0,-]}\colon [0,\infty]^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$ that sends ϵ to the coend $CN(X) \otimes_{[0,\infty]} A_{[0,\epsilon]}$. The **blurred magnitude homology of X** is the composition $H_\star(CN(X) \otimes_{[0,\infty]} A_{[0,-]})$. We then have (see Theorem 4.10.85):

Theorem. For any metric space X, the blurred magnitude homology of X is the persistent homology of $N(X)$.

It remains to convince the reader that this is indeed a right way to relate magnitude homology to persistent homology. For this, we note that the coefficient functors used in the definition of magnitude homology are such that magnitude homology "forgets" the maps induced on the homology groups by the inclusions of simplicial sets $N(X)(\epsilon) \to NX(\epsilon')$, whenever $\epsilon' \geq \epsilon$. This is made clear by the characterisation of the chain complexes $CN(X) \otimes_{[0,\infty]} A_\epsilon$ in Lemma 4.8.78, together with Proposition 4.8.79. By contrast, the "persistence" in persistent homology is exactly the information given by such maps. Thus, morally, persistent homology and magnitude homology are very different homology theories.

Another way to see that persistent homology and magnitude homology are intrinsically different is given by taking their limit. In what we call "limit homology", one is interested

not in the homology of a functor $CS(X)\colon [0,\infty]^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$, but in its homology as ϵ approaches zero, namely in taking the categorical limit of the sequence of homology groups $\{H_\star(CS(X)(\epsilon))\}$. Such a homology theory was first introduced by Vietoris in [201] to define a homology theory for compact metric spaces. To do this, Vietoris introduced what is now called the Vietoris–Rips complex at scale ϵ: this is the simplicial complex $V(X)$ whose n-simplices are the unordered $(n+1)$-tuples of points $\{x_0,\dots,x_n\}$ of X obeying

$$\forall i,j \quad d(x_i,x_j) \leq \epsilon.$$

Vietoris defined the homology of a compact metric space (X,d) to be the limit

$$\mathcal{H}_\star(X) := \varprojlim_{\epsilon} H_\star(CV(X)(\epsilon))\,.$$

Vietoris's motivation was to prove what is now called the "Vietoris mapping theorem", a result that relates the homology groups of two spaces using properties of a map between them. While there has been some work done on Vietoris homology (see, e.g., [104, 174]), the theory has not been as widely studied as other homology theories. A limit homology theory that plays a fundamental role in algebraic topology is Čech homology: given a space X and a cover $\mathcal{U}$ of X, one considers the simplicial homology $H_\star(CN(\mathcal{U}))$ of the nerve of $\mathcal{U}$. If $\mathcal{V}$ is a cover of X that refines $\mathcal{U}$, then there is a homomorphism $H_\star(CN(\mathcal{V})) \to H_\star(CN(\mathcal{U}))$. The Čech homology of X is the inverse limit of the inverse system obtained by considering all open covers of X.

In Corollary 4.11.89 we show that for finite metric spaces ordinary and blurred version of magnitude homology — which coincides with the persistent homology of the enriched nerve — differ in the limit: while the limit of blurred magnitude homology coincides with Vietoris homology, the limit of ordinary magnitude homology is trivial.

Motivated by the relationship of persistent homology with (blurred) magnitude homology, we conjecture the existence of an Euler characteristic for persistent homology in Section 4.12, which would then give the **persistence function**, an invariant for persistent homology. In turn, by considering generalised Lawvere metric spaces, this invariant could give an invariant for multiparameter persistent homology.

We note that while in limit homology and persistent homology one works with simplicial complexes, the definition of magnitude homology is based on simplicial sets. While simplicial complexes present advantages from the computational point of view, as a simplex can be uniquely specified by listing its vertices, from the theoretical point of view simplicial sets are better suited. In Section 4.2 we will explain how to a given simplicial complex one can assign a simplicial set such that their geometric realisations are homeomorphic. This assignment is not functorial, however this will not be a problem for our purposes.

4.1.1 Structure of the chapter

The chapter is structured as follows:

- We cover preliminaries about simplicial complexes and simplicial sets in Section 4.2; enriched categories and Lawvere metric spaces in Section 4.3; filtered simplicial sets in Section 4.4; and persistent as well as graded objects in Section 4.5.
- In Section 4.7 we recall the definition of magnitude for enriched categories, and briefly discuss the special case of the magnitude of metric spaces as well as the magnitude function.
- In Section 4.8 we give the definition of magnitude homology as a special case of Hochschild homology following [139] (see Definition A in Section 4.8.1), and then introduce an alternative definition (Definition B' in Section 4.8.2), and show that they are equivalent in Proposition 4.8.79. In Section 4.8.3 we briefly summarise how magnitude homology categorifies the magnitude, following [139].
- In Section 4.9 we give a general definition of persistent homology, while in Section 4.10 we introduce the blurred magnitude homology, and show that it is the persistent homology of the enriched nerve. We show how blurred and ordinary magnitude homology differ in the limit in Section 4.11.
- Finally, we conjecture the existence of an Euler characteristic for persistent homology in Section 4.12.

4.2 Simplicial complexes and simplicial sets

Simplicial complexes and simplicial sets can be seen as combinatorial versions of topological spaces. They are related to topological spaces by the geometric realisation. Here we recall the definitions of simplicial complexes and simplicial sets and discuss how one can assign a simplicial set to a simplicial complex in such a way that the corresponding geometric realisations are homeomorphic.

Definition 4.2.58. A **simplicial complex** is a tuple $K = (V, \Sigma)$ where V is a set, and Σ is a set of non-empty finite subsets of V such that:

(i) for all $v \in V$ we have that $\{v\} \in \Sigma$

(ii) Σ is closed with respect to taking subsets.

The elements of Σ with cardinality $n+1$ are called ***n*-simplices** of K. The elements of V are called **vertices** of K. Given two simplicial complexes $K = (V, \Sigma)$ and $K' = (V', \Sigma')$, a **simplicial map** $K \to K'$ is a map $f\colon V \to V'$ such that for all $\sigma \in \Sigma$ we have $f(\sigma) \in \Sigma'$.

Remark 4.2.59. We note that if one wants the 0-simplices to coincide with the vertices of a simplicial complex, then condition (i) in Definition 4.2.58 cannot be dispensed of; while condition (ii) implies that all vertices contained in simplices are in Σ, condition (i) guarantees that these are the only vertices. Often in the literature one finds a definition of simplicial complex as a variant of Definition 4.2.58 in which condition (i) is omitted, and in such a definition one thus allows vertices that are not 0-simplices. One could give a definition equivalent to Definition 4.2.58 by only requiring closure under taking subsets as follows: let Σ be a family of non-empty finite sets closed under taking subsets, and let $V(\Sigma) = \bigcup \Sigma$. Then $(V(\Sigma), \Sigma)$ is a simplicial complex according to Definition 4.2.58.

To define simplicial sets, we first need to introduce the "simplex category" Δ. Consider the category with objects finite non-empty totally ordered sets, and morphisms given by order preserving maps. The skeleton of this category is denoted by Δ and called **simplex category**. In other words, Δ has objects given by a totally ordered set $[n] = \{0, 1, \ldots, n\}$ for every natural number n, and morphisms order-preserving maps.

Definition 4.2.60. A **simplicial set** is a functor $S\colon \Delta^{\mathrm{op}} \to \mathrm{Set}$. The elements of $S(n)$ are called ***n*-simplices**.

Explicitly, one can show that a simplicial set is a collection of sets $\{S_n\}_{n\in\mathbb{N}}$ together with maps

$$d_i\colon S_n \to S_{n-1}$$

and

$$s_i\colon S_n \to S_{n+1}$$

for all $0 \leq i \leq n$, that satisfy certain compatibility conditions, see [54, Def. 1.1].

The geometric realisation functor gives a canonical way to associate a topological space to a simplicial complex or set. For this, one chooses a topological model for n-simplices, namely the standard n-simplex Δ^n, and then proceeds to glue together standard simplices using the face and boundary maps.

Definition 4.2.61. The **standard *n*-simplex** is the subset of Euclidean space

$$\Delta^n = \left\{ (x_0, \ldots, x_n) \in \mathbb{R}^n \mid \sum_{i=0}^{n} x_i = 1\,, \text{ and } 0 \leq x_i \leq 1 \text{ for all } i \right\}.$$

There are **boundary** maps

$$\delta_i \colon \Delta^n \to \Delta^{n-1} \colon (x_0, \ldots, x_n) \mapsto (x_0, \ldots, x_i + x_{i+1}, \ldots, x_n)$$

and **face** maps

$$\sigma_i \colon \Delta^n \to \Delta^{n+1} \colon (x_0, \ldots, x_n) \mapsto (x_0, \ldots, x_{i-1}, 0, x_i, \ldots, x_n)\,.$$

Given a simplicial complex $K = (V, \Sigma)$, its **geometric realisation $|K|$** is the quotient space

$$\bigcup_{\sigma \in \Sigma} \Delta^{|\sigma|-1} \times \{\sigma\} / \sim$$

where $\bigcup_{\sigma\in\Sigma} \Delta^{|\sigma|-1} \times \{\sigma\}$ is endowed with the disjoint union space topology, while the equivalence relation $\sim$ is the transitive closure of the following relation

$$\left\{ \Big((x, \sigma), (\delta_i(x), \tau) \Big) \colon x \in \Delta_{|\sigma|-1}, \text{ and } \sigma \subseteq \tau \text{ with } |\tau| = |\sigma| + 1 \right\}.$$

Similarly, given a simplicial set $S \colon \Delta^{\mathrm{op}} \to \mathrm{Set}$, its **geometric realisation $|S|$** is the quotient space

$$\bigcup_{n \in \mathbb{N}} \Delta^n \times S_n / \sim$$

where the equivalence relation $\sim$ is the transitive closure of the union of the relations

$$\left\{ \Big((x, d_i(\sigma)), (\sigma_i(x), \sigma) \Big) \colon x \in \Delta^n \text{ and } \sigma \in S_{n+1} \right\}, \text{ and}$$

$$\left\{ \Big((x, s_i(\sigma)), (\delta_i(x), \sigma) \Big) \colon x \in \Delta^n \text{ and } \sigma \in S_{n-1} \right\}.$$

Now, given a simplicial complex $K = (V, \Sigma)$, we assign to it a simplicial set so that its geometric realisation is homeomorphic to that of K. First, choose a total order on V. Let $K_n^{\mathrm{sim}} = \{(x_0, \ldots, x_n) \mid \{x_0, \ldots, x_n\} \in \Sigma \text{ and } x_0 \leq \cdots \leq x_n\}$.

For $0 \leq i \leq n$ let

$$d_i \colon K_n^{\mathrm{sim}} \to K_{n-1}^{\mathrm{sim}} \colon (x_0, \ldots, x_n) \mapsto (x_0, \ldots, \hat{x_i}, \ldots, x_n)\,,$$

where $\hat{x_i}$ means that that entry is missing, and let

$$s_i \colon K_{n-1}^{\mathrm{sim}} \to K_n^{\mathrm{sim}} \colon (x_0, \ldots, x_{n-1}) \mapsto (x_0, \ldots, x_i, x_i, \ldots, x_{n-1})\,.$$

It is then easy to show that $\{K_n^{\mathrm{sim}}\}_{n\in\mathbb{N}}$ together with the maps d_i and s_i is a simplicial set. We denote this simplicial set by K^{sim}. Furthermore, we have:

Lemma 4.2.62. The geometric realisations of K^{sim} and K are homeomorphic.

Proof. This is easy to see, since the non-degenerate simplices are in bijection, and all degenerate simplices are in the image of some non-degenerate simplex. See also [54]. □

The assignment $K \mapsto K^{\mathrm{sim}}$ is not functorial, since it depends on the choice of a total order on V. One could assign a simplicial set to a simplicial complex in a functorial way, so that their geometric realisations are homotopy equivalent rather than homeomorphic, however this is at the cost of adding many more simplices. For our purposes the non-functorial assignment $K \mapsto K^{\mathrm{sim}}$ suffices.

4.3 Enriched categories and Lawvere metric spaces

An ordinary (small) category C is given by a set of objects, and for every pair of objects x, y a set of morphisms $C(x, y)$, together with composition maps

$$C(x, y) \times C(y, z) \to C(x, z)$$

and maps assigning to every object x its identity morphism

$$\{\star\} \to C(x, x)\,,$$

such that the composition of morphisms is associative and the identity morphism for every object is the neutral element for this composition. Let $\mathcal{V}$ be a monoidal category with tensor product $\otimes_{\mathcal{V}}$ and unit $1_{\mathcal{V}}$. A (small) **category enriched over $\boldsymbol{\mathcal{V}}$** (or **$\boldsymbol{\mathcal{V}}$-category**) is a generalisation of an ordinary category: we still have a set of objects, for every pair of objects x, y we are given an object $C(x, y)$ in $\mathcal{V}$, together with composition and identity assigning morphisms in $\mathcal{V}$, namely

$$C(x, y) \otimes_{\mathcal{V}} C(y, z) \to C(x, z)$$

and

$$1_{\mathcal{V}} \to C(x, x)\,,$$

which satisfy associativity and unitality conditions. When $\mathcal{V}$ is the category of sets, a category enriched over $\mathcal{V}$ is an ordinary category. We note that while an enriched category is in general not a category, it has an "underlying" category, see [120] for details.

In [133] Lawvere observed that any metric space is an enriched category:

Definition 4.3.63. Let $\mathcal{P}^+$ denote the symmetric monoidal category with objects given by the extended non-negative real numbers (that is, elements of $[0,\infty]$), exactly one morphism $l' \to l$ if $l' \geq l$, tensor product given by addition, and unit by 0. A **Lawvere metric space** is a small category enriched over $\mathcal{P}^+$.

In other words, a Lawvere metric space is given by a set X, together with for all $x, y \in X$ a number $X(x,y) \in [0,\infty]$, and for all $x, y, z \in X$ a morphism

$$X(x,y) + X(y,z) \to X(x,z) \tag{4.1}$$

as well as a morphism

$$0 \to X(x,x)\,. \tag{4.2}$$

Equation (4.1) is the triangle inequality, while Equation (4.2) implies that $X(x,x) = 0$. Thus, a Lawvere metric space is the same thing as an extended (since we are allowing infinite distances) quasi-pseudo-metric space (as distances are not necessarily symmetric, and we allow different elements to have zero distance).

4.4 Filtered simplicial sets

Given a metric space (X,d) we are interested in associating to it a filtered simplicial set, namely a functor $S\colon \mathcal{P}^{+\mathrm{op}} \to \mathrm{sSet}$, in such a way that $S(\epsilon)$ captures the topology and geometry of X "below ϵ", in a suitable sense. Examples of such filtered simplicial sets that we consider in this paper are the enriched nerve and the Vietoris–Rips simplicial set. We next recall their definitions.

Definition 4.4.64. Let (X,d) be a metric space. The **enriched nerve** of X is the filtered simplicial set $N(X)$ such that for any $\epsilon \in [0,\infty]$ the simplicial set $N(X)(\epsilon)$ has set of n-simplices given by

$$N(X)(\epsilon)_n = \left\{ (x_0, \ldots, x_n) \mid x_i \in X, \text{ and } \sum_{i=0}^{n-1} d(x_i, x_{i+1}) \leq \epsilon) \right\}$$

and the obvious degeneracy and face maps. Further, for any $\epsilon' \geq \epsilon$ the simplicial maps $N(X)(\epsilon' \geq \epsilon)\colon N(X)(\epsilon) \to N(X)(\epsilon')$ are the canonical inclusion maps.

Definition 4.4.65. Let (X,d) be a metric space, and put a total order on X. The **length** of an ordered tuple $(x_0, \ldots, x_n)$ of elements of X is $\sum_{i=0}^{n-1} d(x_i, x_{i+1})$.

Definition 4.4.66. Let (X, d) be a metric space. The **Vietoris–Rips simplicial set** of X is the filtered simplicial set $V^{\mathrm{sim}}(X)$ with set of n-simplices given by

$$V^{\mathrm{sim}}(X)(\epsilon)_n = \{(x_0, \dots, x_n) \mid d(x_i, x_j) \leq \epsilon \text{ for all } i, j \in \{0, \dots, n\}\}$$

and the obvious degeneracy and face maps. Furthermore, for any $\epsilon' \geq \epsilon$ the simplicial maps $V^{\mathrm{sim}}(X)(\epsilon' \geq \epsilon) \colon V^{\mathrm{sim}}(X)(\epsilon) \to V^{\mathrm{sim}}(X)(\epsilon')$ are the canonical inclusion maps.

4.5 Persistent vs. graded objects

Our aim is to study the homology of filtered simplicial sets such as those introduced in Section 4.4, and we are thus interested in functors $\mathcal{P}^{+\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$. Since such functors are the central object of study in persistent homology, we introduce the following definition:

Definition 4.5.67. Let C be a small category, and $(P, \leq)$ a poset, which we identify with the category with objects given by the elements of P and exactly one morphism $p' \to p$ if $p' \leq p$. A functor $P \to C$ is called a ***P*-persistent** element of the set of objects of C.

Example 4.5.68. Let $P = \mathbb{N}$ and let $\leq$ be the usual order on the natural numbers, and further let $C = \mathbb{K}\mathrm{Vect}$ be the category of vector spaces over a field $\mathbb{K}$ together with linear maps. There is an isomorphism of categories between the functor category of $\mathbb{N}$-persistent vector spaces over $\mathbb{K}$ and the category of $\mathbb{N}$-graded modules over the polynomial ring $\mathbb{K}[x]$. Implicit in this isomorphism is the fact that we are endowing the poset $(\mathbb{N}, \leq)$ with the usual monoid structure given by addition, which is compatible with the order. Similarly, when we consider the poset $([0, \infty], \leq)$ together with monoid structure given by addition, there is an isomorphism of categories between the functor category of $[0, \infty]$-persistent vector spaces and the category of $[0, \infty]$-graded modules over the monoid ring $\mathbb{K}[([0, \infty], +)]$. Furthermore, finitely generated modules correspond to persistent vector spaces of "finitely presented type" [50]. This is known as the Correspondence Theorem in the persistent homology literature, and $\mathbb{N}$-, as well as $[0, \infty]$-persistent vector spaces are usually called **persistence modules**.

On the other hand, we will see that in magnitude homology one "forgets" the information given by the inclusion maps in the filtration of a simplicial sets, and thus the chain complexes that one ends up with are more properly *graded* objects, rather than persistent objects.

Definition 4.5.69. Let C be a small category, and I a set, which we identify with the discrete category with objects given by the elements of I and no morphisms apart from the identity morphisms. A functor $I \to C$ is called an ***I*-graded** element of the set of objects of C.

If C has all coproducts, then one can characterise such functors as follows:

Proposition 4.5.70. Let C be a small category with all coproducts, and let I be a set. There is an isomorphism of categories between the functor category of I-graded objects of C and the category with objects pairs $(c, \{c_i\}_{i\in I})$ such that c is isomorphic to the coproduct of $\{c_i\}_{i\in I}$, and morphisms $(c, \{c_i\}_{i\in I}) \to (c', \{c'_i\}_{i\in I})$ given by $\{f_i\}_{i\in I}$ where for each $i \in I$ we have that $f_i \colon c_i \to c'_i$ is a morphism in C.

Example 4.5.71. When $I = [0, \infty]$, we have that a $[0, \infty]$-graded chain complex of abelian groups can be identified with a chain complex of $[0, \infty]$-graded abelian groups, because coproducts of chain complexes are computed componentwise.

Thus, while a $[0, \infty]$-graded vector space over $\mathbb{K}$ is simply a vector space V together with a direct sum decomposition $V = \oplus_{l\in[0,\infty]} V_l$, we have that a $[0, \infty]$-persistent vector space over $\mathbb{K}$ is a $[0, \infty]$-graded vector space together with an action of the monoid ring $\mathbb{K}[([0, \infty], +)]$, which corresponds to the information given by the non-trivial maps $l \to l'$ whenever $l \leq l'$.

4.6 Coends

One of the main ingredients in the definition of blurred magnitude homology that we will give in Section 4.10 is the coend, a construction that is ubiquitous in category theory. For ease of reference we briefly recall its definition here.

Intuitively, given a bivariate functor with mixed variance $F \colon D^{\mathrm{op}} \times D \to C$, its coend is an object in C that identifies the "left action" of F with the "right action" of F; for instance, the tensor product of a left module with a right module over a ring is an example of coend, see [145, Section IX.6].

While one can define a coend in this general setting, we will make use of the following characterisation of coends in the case that D is cocomplete and C small.

Definition 4.6.72. Suppose that D is a cocomplete category, and C is a small category. Given a functor $F \colon C^{\mathrm{op}} \times C \to D$, its coend is the coequaliser of the diagram

$$\coprod_{f \colon c \to c'} F(c', c) \rightrightarrows \coprod_{c \in C} F(c, c)\,,$$

where the two parallel morphisms are the unique morphisms induced by the morphisms $F(f, 1_c) \colon F(c', c) \to F(c, c)$, and $F(1_{c'}, f) \colon F(c', c) \to F(c', c')$, respectively.

For more details on coends we refer the reader to [145, Section IX.6], as well as the survey [144].

4.7 Magnitude

We first recall the definition of magnitude for arbitrary enriched categories following [139], and then briefly discuss some properties of the magnitude function of a metric space.

Let $\mathcal{V}$ be a symmetric monoidal category with tensor product $\otimes_{\mathcal{V}}$ and unit $1_{\mathcal{V}}$. We denote the set of objects of $\mathcal{V}$ by $\mathrm{ob}\mathcal{V}$.

Definition 4.7.73. Let $(\mathbb{K}, \cdot, +, 1, 0)$ be a semiring. A **size** is a function $\#\colon \mathrm{ob}(\mathcal{V}) \to \mathbb{K}$ such that

(i) if $x \cong y$ then $\#x = \#y$

(ii) $\#(1_{\mathcal{V}}) = 1$ and $\#(x \otimes_{\mathcal{V}} y) = \#(x) \cdot \#(y)$.

If $\mathcal{V}$ is essentially small (i.e., equivalent to a small category), then there is a size function $\#\colon \mathrm{ob}(\mathcal{V}) \to \mathbb{K}$ that is universal, in the sense that any other size function $\#'\colon \mathrm{ob}(\mathcal{V}) \to \mathbb{K}'$ factors through it. We first describe the semiring $\mathbb{K}$. The set of isomorphism classes of $\mathcal{V}$ has a structure of monoid, where $[x] \cdot [y] = [x \otimes_{\mathcal{V}} y]$, and $1 \cdot [x] = [1_{\mathcal{V}} \otimes_{\mathcal{V}} x]$. Given a monoid $(M, \cdot, 1_M)$, there is a canonical way to associate a semiring $F(M)$ to it, given by the left adjoint functor to the forgetful functor that given a semiring $(R, +, 0, \cdot, 1, +)$ forgets the monoid structure $(R, +, 0)$. The semiring $F(M)$ is thus given by formal $\mathbb{N}$-linear combinations of elements of M, subject to the relation $m^0 \sim 1_M$ for all $m \in M$, and the relations that make the binary operation $\cdot$ of M distributive over taking linear combinations. Thus, we can consider the free semiring associated to the monoid of isomorphism classes of $\mathrm{ob}(\mathcal{V})$, and we denote it by $F(\mathrm{ob}\mathcal{V}/\cong)$.

Definition 4.7.74. Let $\mathcal{V}$ be an essentially small symmetric monoidal category. The **canonical size function** of $\mathcal{V}$ is the function $\#\colon \mathrm{ob}(\mathcal{V}) \to F(\mathrm{ob}\mathcal{V}/\cong)\colon x \mapsto [x]$ that sends every object to its isomorphism class.

In particular, for $\mathcal{V} = \mathcal{P}^+$, the elements of $F(\mathrm{ob}\mathcal{V}/\cong)$ are formal $\mathbb{N}$-linear combinations of numbers in $[0, \infty]$, and we have $[l_1] \cdot [l_2] = [l_1 + l_2]$. This motivates the introduction of the notation $\sum_i a_i q^{l_i}$ for $\sum_i a_i l_i$ and $\mathbb{N}[q^{[0,\infty]}]$ for $F(\mathrm{ob}\mathcal{V}/\cong)$.

Definition 4.7.75. Let X be a finite $\mathcal{V}$-category, together with a size $\#\colon \mathrm{ob}\mathcal{V} \to \mathbb{K}$, and put a total order on the objects of X. The **zeta function** of X is the matrix Z_X indexed by the objects of X with

$$Z_X(x, y) = \#(X(x, y))\,.$$

One says that X has **Möbius inversion with respect to #** if Z_X is invertible over $\mathbb{K}$. The **magnitude of X with respect to #** is the sum of all entries of Z_X^{-1}, and is denoted by $|X|$.

In general there are more inverse matrices over a field than a semiring, so it is natural to complete $\mathbb{K}$ to a field. For this, one first needs to restrict to pseudo-quasi-metric spaces, so to categories enriched over $[0, \infty)$, as otherwise q^∞ is a non-trivial zero divisor. Denote by $\mathbb{Q}(q^{[0,\infty)})$ the field of fractions of $\mathbb{Z}[q^{[0,\infty)}]$. One then has that any finite quasi-metric space has Möbius inversion with respect to the canonical size function [139, Theorem 2.10]. So in particular any quasi-metric space has a magnitude lying in $\mathbb{Q}(q^{[0,\infty)})$.

Evaluating the magnitude of a metric space with respect to the canonical size function at $q = e^{-t}$ for varying values of t gives the magnitude function:

Definition 4.7.76. Let (X, d) be a finite metric space, and let $f(q) \in \mathbb{Q}(q^{[0,\infty)})$ be its magnitude with respect to the canonical size function. The **magnitude function** of (X, d) is the (partially defined) function

$$\begin{aligned} [0, \infty) &\to \mathbb{R} \\ t &\mapsto f(e^{-t})\,. \end{aligned}$$

For arbitrary finite metric spaces the magnitude function can be ill-behaved: it might only be partially defined, take on negative values, or the magnitude of a subspace might be greater than the magnitude of the whole space. However, for a certain class of nice spaces — positive definite spaces — the magnitude function is well-defined and has many of the properties that one would expect a cardinality-like invariant to have [137, Section 2.4]. In particular, all finite subsets of Euclidean space are positive definite spaces. For such spaces one has:

- The magnitude function is well-defined [137, Lemma 2.4.2 (i)].
- $1 \leq f(e^{-t})$ for all t, see [137, Corollary 2.4.5].
- $\lim_{t\to\infty} f(e^{-t}) = \#X$, where $\#X$ here denotes the cardinality of the set X (this holds for arbitrary finite metric spaces) [137, Proposition 2.2.6].

4.8 Magnitude homology

Leinster and Shulman introduce magnitude homology in [139], as the categorification of the magnitude of a metric space. Here we first briefly recall the definition of magnitude homology as given in [139], then we give an alternative equivalent definition that will be the definition that we will use later to relate persistent homology to magnitude homology, and briefly summarise how magnitude homology categorifies the magnitude. In this and the next sections we denote by $\mathcal{P}$ the symmetric monoidal category with set of objects given by the non-negative real numbers $[0, \infty)$, with exactly one morphism $l' \to l$ whenever $l' \geq l$, tensor product given by addition, and unit by 0.

4.8.1 Magnitude homology as Hochschild homology

Let $\mathcal{A}$ denote the category of $[0,\infty)$-graded abelian groups. This category has two different monoidal structures, induced by two different tensor products: the componentwise tensor product $(A \boxtimes B)_l = A_l \otimes_{\mathbb{Z}} B_l$ and the tensor product $\otimes$ given by "convolution", namely $(A \otimes B)_l = \oplus_{i+j=l} A_i \otimes_{\mathbb{Z}} B_j$. These in turn induce two different tensor products on $\mathrm{ch}_{\mathcal{A}}$, by taking as usual the tensor product of chain complexes to be defined by "convolution", namely $(C_\bullet \otimes D_\bullet)_n = \oplus_{i+j=n} C_i \otimes D_j$, and similarly for the product $\boxtimes$. Given a $\mathcal{P}$-category X, and a monoidal functor $\Sigma\colon \mathcal{P} \longrightarrow \mathrm{ch}_{\mathcal{A}}$, this makes X into a $\mathrm{ch}_{\mathcal{A}}$-category, which we denote by ΣX. Now, to define magnitude homology, Leinster and Shulman consider the following **functor of coefficients**

$$\begin{aligned} \Sigma\colon \mathcal{P} &\to \mathcal{A} \\ l &\mapsto \mathbb{Z}_l \end{aligned}$$

where $\mathbb{Z}_l$ is the $[0,\infty)$-graded abelian group with a copy of $\mathbb{Z}$ in degree l and 0 otherwise, and we consider Σ to be a functor taking values in $\mathrm{ch}_{\mathcal{A}}$ through the canonical inclusion functor $\mathcal{A} \hookrightarrow ch_{\mathcal{A}}$ sending an abelian group to the chain complex concentrated in degree zero. The functor Σ is strongly monoidal with respect to the product $\otimes$ on $ch_{\mathcal{A}}$: we have $\Sigma(X(x,y)+X(y,z)) \cong \Sigma(X(x,y)) \otimes \Sigma(X(y,z))$ for all $x, y, z \in [0,\infty)$. Next, Leinster and Shulman consider the two-sided simplicial bar construction [186] with n-simplices given by the $[0,\infty)$-graded chain complex:

$$B_n(\Sigma X) := \bigsqcup_{(x_0,\ldots,x_n)} \Sigma X(x_0,x_0) \otimes \Sigma X(x_0,x_1) \otimes \cdots \otimes \Sigma X(x_{n-1},x_n) \otimes \Sigma X(x_n,x_n)\,.$$

This can be seen as an instance of a two-sided simplicial bar construction by noting that $\Sigma X(x_n,x_n)$ acts on $\Sigma X(x_{n-1},x_n)$ from the left, while $\Sigma X(x_0,x_0)$ acts on $\Sigma X(x_0,x_1)$ from the right. Since Σ is strongly monoidal with respect to $\otimes$, we have that

$$B_n(\Sigma X) \cong \bigsqcup_{(x_0,\ldots,x_n)} \Sigma(X(x_0,x_1) + \cdots + X(x_{n-1},x_n))\,.$$

Definition 4.8.77. [139, p. 9–12] Let X be a $\mathcal{P}$-category. The **magnitude nerve** of X is the simplicial $[0,\infty)$-graded chain complex $B_\bullet(\Sigma X)$.

Magnitude homology (Definition A). The **magnitude homology** of X is the homology of the geometric realisation $B(\Sigma X)$ of $B_\bullet(\Sigma X)$, and denoted by $H_\star^\Sigma(X)$.

The assignment $X \mapsto H_\star^\Sigma(X)$ induces a functor from the category with objects $\mathcal{P}$-categories and morphisms $\mathcal{P}$-functors to the category of $[0,\infty)$-graded abelian groups [139,

Theorem 4.4]. Explicitly, one can describe the chain complex $B(\Sigma X)$ as follows [139, Section 5]: one has

$$B(\Sigma X)_n = \bigoplus_{l\in[0,\infty)} \mathbb{Z}\left[\{(x_0,\dots,x_n) \mid \sum_{i=0}^{n} d(x_i,x_{i+1}) = l\}\right] . \tag{4.3}$$

Thus, in degree n it is the free $[0,\infty)$-graded abelian group, which in degree l is generated by the ordered tuples $(x_0,\dots,x_n)$ of length exactly l. Furthermore, the boundary map $d_n\colon B(\Sigma X)_n \to B(\Sigma X)_{n-1}$ is given by the alternating sum of maps d_n^i, defined as follows for all $1 \le i \le n-1$:

$$d_n^i(x_0,\dots,x_n) = \begin{cases} (x_0,\dots,x_{i-1},x_{i+1},\dots x_n), & \text{if } d(x_{i-1},x_i)+d(x_i,x_{i+1}) = d(x_{i-1},x_{i+1}) \\ 0, & \text{otherwise} \end{cases}$$

while for $i = 0$ we have

$$d_n^0(x_0,\dots,x_n) = \begin{cases} (x_1,x_2,\dots x_n), & \text{if } d(x_0,x_1) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

and similarly for $i = n$.

4.8.2 Magnitude homology: an alternative viewpoint

In online discussions [138] Leinster and Shulman initially gave a different definition of magnitude homology. Here we recall this definition (Definition B), and prove that an adaptation of it (Definition B') agrees with the definition given in the previous section (Definition A). We will use Definition B' of magnitude homology to relate magnitude homology to persistent homology.

Here Ab denotes the category of abelian groups with monoidal structure given by the tensor product, which we denote by $\boxtimes$; this induces a monoidal structure on the category of chain complexes over Ab. Given a $\mathcal{P}$-category X, Leinster and Shulman consider the following functor

$$CN(X) = \left(\mathcal{P}^{\mathrm{op}} \xrightarrow{N(X)} \mathrm{sSet} \xrightarrow{\mathbb{Z}[\cdot]} \mathrm{sAb} \xrightarrow{U} \mathrm{ch}_{\mathrm{Ab}}\right) , \tag{4.5}$$

where the functor $\mathbb{Z}[\cdot]$ is induced by the free abelian group functor, and the functor U is the functor that sends a simplicial abelian group to its unnormalised chain complex. They then introduce a functor of coefficients $A\colon \mathcal{P} \longrightarrow \mathrm{Ab}$, where one views A as taking values in $\mathrm{ch}_{\mathrm{Ab}}$ through the canonical inclusion $\mathrm{Ab} \hookrightarrow \mathrm{ch}_{\mathrm{Ab}}$, and give the following definition:

Magnitude homology (Definition B). The **magnitude homology of X with coefficients in A** is the homology of the chain complex given by the coend $CN(X) \otimes_{\mathcal{P}} A$.

Explicitly, one can describe this chain complex as follows for a particular choice of coefficient functor.

Lemma 4.8.78. Let $A_\epsilon \colon \mathcal{P} \longrightarrow \mathrm{Ab}$, be the following functor:

$$\begin{aligned} A_\epsilon \colon \mathcal{P} &\to \mathrm{Ab} \\ l &\mapsto \begin{cases} \mathbb{Z}, \text{if } l = \epsilon \\ 0, \text{otherwise,} \end{cases} \end{aligned}$$

which we consider as taking values in $\mathrm{ch}_{\mathrm{Ab}}$ through the canonical inclusion $\mathrm{Ab} \hookrightarrow \mathrm{ch}_{\mathrm{Ab}}$. The chain complex $CN(X) \otimes_{\mathcal{P}} A_\epsilon$ is given in degree n by the free abelian group on the tuples $(x_0, \ldots, x_n)$ that have length exactly ϵ. The boundary maps d_n are given by alternating sums of maps d_n^i which can be described as follows, for $0 < i < n$:

$$\begin{aligned} d_n^i \colon (CN(X) \otimes_{\mathcal{P}} A_\epsilon)_n &\to (CN(X) \otimes_{\mathcal{P}} A_\epsilon)_{n-1} \\ (x_0, \ldots, x_n) &\mapsto \begin{cases} (x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n), & \text{if } d(x_{i-1}, x_{i+1}) = d(x_{i-1}, x_i) + d(x_i, x_{i+1}) \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

while for $i = 0$ we have

$$\begin{aligned} d_n^0 \colon (CN(X) \otimes_{\mathcal{P}} A_\epsilon)_n &\to (CN(X) \otimes_{\mathcal{P}} A_\epsilon)_{n-1} \\ (x_0, \ldots, x_n) &\mapsto \begin{cases} (x_1, x_2, \ldots, x_n), & \text{if } d(x_0, x_1) = 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

and similarly for $i = n$.

Proof. [1] The coend $CN(X) \otimes_{\mathcal{P}} A_\epsilon$ is the coequaliser of the following diagram:

$$\bigoplus_{l' \geq l} CN(X)(l) \boxtimes A_\epsilon(l') \underset{\bigoplus_{l' \geq l} 1 \boxtimes A_\epsilon(l' \geq l)}{\overset{\bigoplus_{l' \geq l} CN(X)(l' \geq l) \boxtimes 1}{\rightrightarrows}} \bigoplus_{l \in \mathcal{P}} CN(X)(l) \boxtimes A_\epsilon(l)\,.$$

Thus, it is the coproduct over $l \in \mathcal{P}$ of the chain complexes $CN(X)(l) \boxtimes A_\epsilon(l)$, modulo the relations given by equating the two parallel morphisms on the left hand side. By definition of A_ϵ, the morphisms are both trivial if $l' \neq \epsilon$, thus we assume that $l' = \epsilon$. The two morphisms are identical if $l = \epsilon$, thus we assume that $l \neq \epsilon$, and so we have $\epsilon > l$. Thus, the bottom parallel morphism is zero, while the top parallel morphism is

$$CN(X)(l) \longrightarrow CN(X)(\epsilon)\,.$$

[1] We note that parts of this proof were given by Shulman in an online comment [185].

Furthermore, we have

$$\bigoplus_{l\in\mathcal{P}} CN(X)(l)\boxtimes A_\epsilon(l)\cong CN(X)(\epsilon)$$

since tensoring with $A_\epsilon(l)$ makes all summands vanish, except if $l=\epsilon$. Thus, in degree n the chain complex $CN(X)\otimes_{\mathcal{P}} A_\epsilon$ is the free abelian group on the tuples $(x_0,\dots,x_n)$ that have length exactly ϵ.

Now, denote by $D(\epsilon)$ the subcomplex of $CN(X)(\epsilon)$ whose n-chains are the n-tuples with length strictly less than ϵ, so that $CN(X)\otimes_{\mathcal{P}} A_\epsilon\cong CN(X)(\epsilon)/D(\epsilon)$ by the previous discussion. Note that the boundary map on the quotient chain complex $CN(X)(\epsilon)/D(\epsilon)$ is the alternating sum of maps

$$d_n^i\colon CN(X)(\epsilon)_n/D(\epsilon)_n\to CN(X)(\epsilon)_{n-1}/D(\epsilon)_{n-1}$$

which send $c+D(\epsilon)_n$ to $d_n^{i,C}(c)+D(\epsilon)_{n-1}$, where $d_n^{i,C}\colon CN(X)(\epsilon)_n\to CN(X)(\epsilon)_{n-1}$ is the map induced by the ith face map. Thus $d_n^i(x_0,\dots,x_n)$ is the map induced by the ith face maps if and only if by deleting the ith entry the length of the tuple is unchanged, and is the zero map otherwise.

□

Our aim is to relate Definition B with Definition A. For any $\epsilon'\geq\epsilon$ define the natural transformation $\iota\colon A_\epsilon\Rightarrow A_{\epsilon'}$ where ι_l is the identity if $l=\epsilon=\epsilon'$, and the zero map otherwise. This induces a chain map $CN(X)\otimes_{\mathcal{P}} A_\epsilon\to CN(X)\otimes_{\mathcal{P}} A_{\epsilon'}$, and we thus have a functor $CN(X)\otimes_{\mathcal{P}} A_-\colon\mathcal{P}^{\mathrm{op}}\to \mathrm{ch}_{\mathrm{Ab}}$ that assigns to a number ϵ the chain complex $CN(X)\otimes_{\mathcal{P}} A_\epsilon$. The functor $CN(X)\otimes_{\mathcal{P}} A_-$ factors through the projection functor $\mathcal{P}^{\mathrm{op}}\to[0,\infty)$, where $[0,\infty)$ is the discrete category with objects given by non-negative real numbers. Recall that a chain complex of of $[0,\infty)$-graded abelian groups can be identified with an $[0,\infty)$-graded chain complex (see Example 4.5.71). Thus, in particular, we can identify the chain complex $B(\Sigma X)$ with a functor $\mathcal{P}^{\mathrm{op}}\to\mathrm{ch}_{\mathrm{Ab}}$ that coincides with $B(\Sigma X)$ on the set of objects, and sends every morphism to the trivial chain map. We have:

Proposition 4.8.79. The functors $CN(X)\otimes_{\mathcal{P}} A_-$ and $B(\Sigma X)$ are isomorphic. In particular, the magnitude homology of X (Definition A) is isomorphic to the homology of $CN(X)\otimes_{\mathcal{P}} A_-$.

Proof. By Lemma 4.8.78 and (4.3) the chain complexes $B(\Sigma X)_l$ and $CN(X)\otimes_{\mathcal{P}} A_l$ are canonically isomorphic. Next, for $l'\geq l$, consider the following diagram:

$$\begin{array}{ccc} B(\Sigma X)_l & \longrightarrow & B(\Sigma X)_{l'} \\ \downarrow & & \downarrow \\ CN(X)\otimes_{\mathcal{P}} A_l & \longrightarrow & CN(X)\otimes_{\mathcal{P}} A_{l'} \end{array}$$

where the vertical arrows are the canonical isomorphisms, while the horizontal arrows are zero maps. Thus every such square commutes, so the canonical isomorphisms assemble into a natural isomorphism between $CN(X) \otimes_{\mathcal{P}} A_-$ and $B(\Sigma X)$. □

For ease of reference, we state here the equivalent definition of magnitude homology, as given by Proposition 4.8.79:

Magnitude homology (Definition B'). The **magnitude homology of X** is the homology of the $[0,\infty)$-graded chain complex $CN(X) \otimes_{\mathcal{P}} A_-$.

4.8.3 Magnitude homology categorifies magnitude

The Euler characteristic can be defined very generally for any dualisable object in a closed symmetric monoidal category [150]. Leinster and Shulman consider the closed symmetric monoidal category $\mathrm{ch}_{\mathcal{A}}$ of chain complexes of $[0,\infty)$-graded abelian groups, together with the tensor product $\boxtimes$ induced by the componentwise tensor product of $[0,\infty)$-graded abelian groups, and for a dualisable object $C_\bullet$ of $\mathrm{ch}_{\mathcal{A}}$ they denote by $\chi^{\boxtimes}(C_\bullet)$ its Euler characteristic, according to the Definition in [150]. The Euler characteristic $\chi^{\boxtimes}(C_\bullet)$ is an element of $\mathbb{Z}^{[0,\infty)}$, the endomorphism ring of the unit object for $\boxtimes$, and one has that $\chi^{\boxtimes}(C_\bullet)_l = \chi(C_{\bullet,l})$, where χ is the usual Euler characteristic of a chain complex of abelian groups. The Euler characteristic of a chain complex that is finitely generated and projective can be computed as the alternating sum of the rank of its homology groups, and Leinster and Shulman prove that this is indeed the case for the chain complexes $B(\Sigma X)_l$, whenever X is a finite quasi-pseudo-metric space [139, Lemma 5.8]. Thus, one has

$$\chi^{\boxtimes}(B(\Sigma X)) = \sum_{l\in[0,\infty)} \sum_{n=0}^{\infty} (-1)^n \mathrm{rank} H_n(B(\Sigma X)_l) q^l \,. \tag{4.6}$$

Leinster and Shulman's goal is to prove that $\chi^{\boxtimes}(B(\Sigma X)) = |X|$, where $|X|$ denotes the magnitude of X with respect to the canonical size, which is an element of $\mathbb{Q}(q^{[0,\infty)})$, and thus one would like to restrict attention to $[0,\infty)$-graded chain complexes such that their Euler characteristic $\chi^{\boxtimes}$ lies in $\mathbb{Q}(q^{[0,\infty)})$. They thus first embed $\mathbb{Z}^{[0,\infty)}$ into $\mathbb{Q}^{[0,\infty)}$. However, in general, even for finite metric spaces X, the sum in Eq. (4.6) is not finite (as for finite quasi-pseudo-metric spaces X one has that $B_\bullet(\Sigma X)$ is not n-skeletal for any n, see [139, p. 19]). An appropriate finiteness condition for series in $\mathbb{Q}^{[0,\infty)}$ is given by having well-ordered support: this guarantees that the product defined by "convolution" is well-defined. Thus, they consider the subset of $\mathbb{Q}^{[0,\infty)}$ consisting of functions $[0,\infty) \to \mathbb{Q}$ having well-ordered support. Such functions form a field with componentwise addition and product given by convolution, and it is called the field of **Hahn series**. They thus restrict attention to chain complexes whose Euler characteristic is a Hahn series.

The field of Hahn series is a valuation field, and as such it has a topology induced by the valuation v. Consider a sequence of Hahn series $\{f_n\}_{n\in\mathbb{N}}$, where for each n we have $f_n := \sum_l c_{n,l}q^l$. Such a sequence converges to $\sum_l \sum_{n=1}^{\infty} c_{n,l}q^l$ in the topology given by the valuation if and only if $\lim_{n\to\infty} v(f_n) = +\infty$. (See [27, VI.5] for details.) One then has:

Theorem 4.8.80. [139, Theorem 5.11, Corollary 6.28] Let X be a finite quasi-pseudo-metric space. Then the sequence $\{H_n(B(\Sigma X))\}_{n\in\mathbb{N}}$ converges in the topology of the field of Hahn series, and one has:

$$\chi^{\boxtimes}(B(\Sigma X)) = \sum_n (-1)^n \mathrm{rank}(H_n(B(\Sigma X)))\,.$$

Furthermore

$$|X| = \sum_n (-1)^n \mathrm{rank} H_n(B(\Sigma X))\,,$$

where $|X|$ is the magnitude of X with respect to the canonical size function.

4.9 Persistent homology

Persistent homology is, in an appropriate sense, the generalisation of simplicial homology of a simplicial set to persistent simplicial sets. Given a metric space (X, d), one associates to it a $\mathcal{P}^{\mathrm{op}}$-persistent simplicial set $S(X)$ such that $S(X)(\epsilon)$ captures in a certain sense the topology of X "below distance ϵ". One then considers the functor

$$CS(X) = \left(\mathcal{P}^{\mathrm{op}} \xrightarrow{S(X)} \mathrm{sSet} \xrightarrow{\mathbb{Z}[\cdot]} \mathrm{sAb} \xrightarrow{U} \mathrm{ch}_{\mathrm{Ab}}\right)\,,$$

where $\mathbb{Z}[\cdot]$ and U are defined as in (4.5). Furthermore, one considers the functor of coefficients

$$\begin{aligned} A\colon \mathcal{P}^{\mathrm{op}} &\to \mathrm{ch}_{\mathrm{Ab}} \\ l &\mapsto \mathbb{F}_l \end{aligned}$$

where $\mathbb{F}$ is a (finite) field, and $\mathbb{F}_l$ is the chain complex with a copy of $\mathbb{F}$ concentrated in degree zero; further, for $l' \geq$ the chain map $\mathbb{F}_l \to \mathbb{F}_{l'}$ is the zero map. The composite $H_\star(CS(X) \boxtimes A)$, where $H_\star\colon \mathrm{ch}_{\mathrm{Ab}} \to \mathrm{Ab}$ is the usual homology functor, is then called the "persistent homology of $S(X)$ with coefficients in $\mathbb{F}$." Using this coefficient functor has the advantage that isomorphism classes of such functors that satisfy certain finiteness conditions can be completely characterised by a collection of intervals, called the **barcode**, see e.g., [166, Theorem 1.9]. More generally, we give the following definition:

Definition 4.9.81. Let S be a $\mathcal{P}^{\mathrm{op}}$-persistent simplicial set, and $A\colon \mathcal{P}^{\mathrm{op}} \to \mathrm{ch}_{\mathrm{Ab}}$ a functor. The **persistent homology of S with coefficients in A** is the composition $H_\star(CS \boxtimes A)$. When A is the unit for $\boxtimes$ we call the homology of $CS \boxtimes A$ the **persistent homology of S**.

For arbitrary coefficient functors one in general no longer has a barcode. However, motivated by the Euler characteristic for magnitude homology, we conjecture the existence of a cardinality-like invariant for such persistence modules in Section 4.12. Such general functors of coefficients might be interesting for applications, as they might allow to capture more refined information, for instance different torsion/orientability phenomena over different filtration scales, which might be detected by taking e.g. coefficients over $\mathbb{F}_2$ (the field with two elements) over a certain interval $I \subset [0, \infty)$, and coefficients over $\mathbb{F}_3$ over a different disjoint interval $J \subset [0, \infty)$. More complicated coefficient functors might allow for an even more refined analysis.

4.10 Magnitude meets persistence

In the final section of [139] Leinster and Shulman list a series of open problems, the last two of which are as follows (from [139, Section 8]):

> 9. Magnitude homology only "notices" whether the triangle inequality is a strict equality or not. Is there a "blurred" version that notices "approximate equalities"?
>
> 10. Almost everyone who encounters both magnitude homology and persistent homology feels that there should be some relationship between them. What is it?

In this section we attempt a first answer to these problems, which we believe are intertwined: it is the blurred version of magnitude homology that is related to persistent homology. Indeed, as is apparent from Proposition 4.8.79, the magnitude homology of a metric space X is a homology theory that in a certain sense "forgets" the maps induced on the homology groups by the inclusions of simplicial sets $N(X)(\epsilon) \to NX(\epsilon')$, whenever $\epsilon' \geq \epsilon$, whereas the "persistence" in persistent homology is exactly the information given by such maps. Thus, morally, these are very different homology theories.

Our starting point is Definition B' of magnitude, which we adapt to coefficient functors not supported at points, but on intervals.

Definition 4.10.82. For any $\epsilon \in [0, \infty)$ define the following functor of coefficients:

$$A_{[0,\epsilon]} \colon \mathcal{P}^{\mathrm{op}} \to \mathrm{Ab}$$
$$l \mapsto \begin{cases} \mathbb{Z}, \text{if } l \in [0, \epsilon] \\ 0, \text{otherwise,} \end{cases}$$

which we consider as taking values in $\mathrm{ch}_{\mathrm{Ab}}$ through the canonical inclusion functor $\mathrm{Ab} \hookrightarrow \mathrm{ch}_{\mathrm{Ab}}$.

Now, for any $\epsilon' \geq \epsilon$ we consider the natural transformation $\iota \colon A_{[0,\epsilon]} \Rightarrow A_{[0,\epsilon']}$ where ι_l is the identity if $l \in [0, \epsilon]$, and the zero map otherwise. This natural transformation induces a chain map $CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]} \to CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon']}$, and we thus have a functor $CN(X) \otimes_{\mathcal{P}} A_{[0,-]} \colon \mathcal{P}^{\mathrm{op}} \to ch_{\mathrm{Ab}}$ that assigns to a number ϵ the chain complex $CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]}$. Explicitly, we can describe the chain complexes $CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]}$ as follows:

Lemma 4.10.83. For any $\epsilon \in [0, \infty)$ we have

$$CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]_n} = \mathbb{Z}\left[\left\{(x_0, \ldots, x_n) \mid \sum_{i=0}^{n} d(x_i, x_{i+1}) \leq \epsilon\right\}\right]$$

with boundary maps given by alternating sums of maps induced by the face maps.

Proof. The chain complex $CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]}$ is the coequaliser of the following diagram:

$$\bigoplus_{l' \geq l} CN(X)(l) \boxtimes A_{[0,\epsilon]}(l') \underset{\bigoplus_{l' \geq l} 1 \boxtimes A_{[0,\epsilon]}(l' \geq l)}{\overset{\bigoplus_{l' \geq l} CN(X)(l' \geq l) \boxtimes 1}{\rightrightarrows}} \bigoplus_{l \in \mathcal{P}} CN(X)(l) \boxtimes A_{[0,\epsilon]}(l)\,.$$

First, note that

$$\bigoplus_{l \in \mathcal{P}} CN(X)(l) \boxtimes A_{[0,\epsilon]}(l) \cong \bigoplus_{l \in [0,\epsilon]} CN(X)(l)\,,$$

as the summands vanish if $l > \epsilon$, by definition of $A_{[0,\epsilon]}$. We next see what relations are given by the two parallel morphisms in the diagram. For $l' > \epsilon$ we have that the morphisms are both zero, so we assume that $l' \in [0, \epsilon]$. Furthermore, the morphisms are identical if $l = l'$, so we assume that $l \neq l'$, and thus have $0 \leq l < l' \leq \epsilon$. Thus, the top horizontal morphism is $CN(X)(l) \to CN(X)(l')$, while the bottom morphism is $CN(X)(l) \to CN(X)(l)$. By equating these morphisms in $\oplus_{l \in [0,\epsilon]} CN(X)(l)$ we are

thus identifying the summand $CN(X)(l)$ with the image of the inclusion of $CN(X)(l)$ in $CN(X)(l')$. We thus obtain

$$CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]} \cong CN(X)(\epsilon)\,.$$

Similarly, the relations given by the pair of parallel morphisms tell us that the boundary maps on the quotient chain complex are the boundary maps of the chain complex $CN(X)(\epsilon)$, thus alternating sums of maps induced by face maps. □

Definition 4.10.84. The **blurred magnitude homology** of X is the composition of functors $H_\star(CN(X) \otimes_{\mathcal{P}} A_{[0,-]})$.

We have:

Theorem 4.10.85. The functors $CN(X) \otimes_{\mathcal{P}} A_{[0,-]}$ and $CN(X)$ are isomorphic. In particular, the blurred magnitude homology of X is isomorphic to the persistent homology of the enriched nerve $N(X)$.

Proof. By Lemma 4.10.83 we know that there is an isomorphism between $CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]}$ and $CN(X)(\epsilon)$ for any $\epsilon \in [0,\infty)$. Next, for any $\epsilon' \geq \epsilon$, the square

$$\begin{array}{ccc} CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon]} & \longrightarrow & CN(X) \otimes_{\mathcal{P}} A_{[0,\epsilon']} \\ \downarrow & & \downarrow \\ CN(X)(\epsilon) & \longrightarrow & CN(X)(\epsilon') \end{array}$$

where the vertical maps are the isomorphisms, commutes, as the horizontal morphisms are inclusions.

□

4.11 Limit homology

In [201] Vietoris introduced what is now called the Vietoris–Rips complex, as a way to define a homology theory for compact metric spaces[2]. One starts by considering the simplicial homology $H_\star(CS)$ of a $\mathcal{P}^{\mathrm{op}}$-persistent chain complex $CS = (\mathcal{P}^{\mathrm{op}} \to \mathrm{sSet} \to \mathrm{sAb} \to \mathrm{ch}_{\mathrm{Ab}})$ as what is usually called an "inverse system" in category theory: there are linear maps $\phi_{\epsilon,\epsilon'} \colon H_\star(CS(\epsilon)) \to H_\star(CS(\epsilon'))$ for all $\epsilon' \geq \epsilon$, which are induced by the simplicial maps

[2]We note that while Vietoris used what is called the "Vietoris–Rips simplicial complex" (at level ϵ) $V(X)(\epsilon)$, here we discuss this homology theory using the simplicial set $V^{\mathrm{sim}}(X)(\epsilon)$ associated to it, see Section 4.4 and Lemma 4.2.62.

$CS(\epsilon) \to CS(\epsilon')$. Furthermore these maps satisfy the property that $\phi_{\epsilon,\epsilon} = id$ and $\phi_{\epsilon,\epsilon''} = \phi_{\epsilon',\epsilon''} \circ \phi_{\epsilon,\epsilon'}$ whenever $\epsilon'' \geq \epsilon' \geq \epsilon$.

Vietoris defined the homology of X (for a compact metric space X) to be the inverse limit of the inverse system in the case $S = V^{\mathrm{sim}}(X)$:

$$\mathcal{H}_\star(X) := \varprojlim_\epsilon H_\star(CV^{\mathrm{sim}}(X)(\epsilon))\,. \tag{4.7}$$

In later work, Hausmann [104] proposed a cohomological counterpart of the homology theory introduced by Vietoris, by considering the direct limit of the direct system that one obtains by taking simplicial cohomology of the Vietoris–Rips simplicial sets:

$$\mathcal{H}^\star(X) := \varinjlim_\epsilon H^\star(CV^{\mathrm{sim}}(X)(\epsilon))\,.$$

He called this cohomology theory "metric cohomology", and not Vietoris cohomology, because the adjective Vietoris had already been used to designate a cohomology theory which is in general not isomorphic to the cohomological counterpart of the homology theory introduced by Vietoris [104]. The denomination "Vietoris–Rips" for the complex introduced by Vietoris is also due to Hausmann, as the complex introduced by Vietoris was in the meantime known as Rips complex [177].

Similarly, we can consider the inverse system that one obtains by considering the simplicial sets $\{N(X)(\epsilon)\}_{\epsilon\in[0,\infty]}$:

$$\varprojlim_\epsilon H_\star(CN(X)(\epsilon))\,. \tag{4.8}$$

In the following we relate the inverse limits (4.7) and (4.8). Let C be any category, and P a poset. The category with objects given by P-persistent objects of C, and morphisms given by natural transformations between them, can be endowed with an extended pseudo-distance, called **interleaving distance** [29]. The interleaving distance was first introduced in [42] for the special case that $P = (\mathbb{R}, \leq)$. The central notion is that of interleaving: for $\epsilon \geq 0$ two functors $M, N\colon (\mathbb{R}, \leq) \to C$ are **ϵ-interleaved** if there are collections of morphisms $\{\phi_\epsilon\colon M(a) \to N(a+\epsilon) \mid a \in \mathbb{R}\}$ and $\{\psi_\epsilon\colon N(a) \to M(a+\epsilon) \mid a \in \mathbb{R}\}$ such that all diagrams of the following form commute:

$$\begin{array}{ccc} M(a-\epsilon) & \longrightarrow & M(a+\epsilon) \\ & \searrow \quad \nearrow & \\ & N(a) & \end{array} \qquad \begin{array}{ccc} M(a+\epsilon) & \longrightarrow & M(b+\epsilon) \\ \uparrow & & \uparrow \\ N(a) & \longrightarrow & N(b) \end{array}$$

$$\begin{array}{ccc} & M(a) & \\ \nearrow & & \nwarrow \\ N(a-\epsilon) & \longrightarrow & N(a+\epsilon) \end{array} \qquad \begin{array}{ccc} M(a) & \longrightarrow & M(b) \\ \searrow & & \searrow \\ N(a+\epsilon) & \longrightarrow & N(b+\epsilon)\,. \end{array}$$

Two functors that are ϵ-interleaved have bounded interleaving distance [42, Theorem 4.4].

On the other hand, in many examples of filtered spaces that one considers in topological data analysis, what one obtains is not an interleaving of the corresponding homologies, but rather an "approximation": two functors $M, N\colon (\mathbb{R}, \leq) \to C$ are ***c*-approximations** of each other if there are collections of morphisms $\{\phi_c\colon M(a) \to N(ca) \mid a \in \mathbb{R}\}$ and $\{\psi_c\colon N(a) \to M(ca) \mid a \in \mathbb{R}\}$ such that all diagrams of the following form commute:

$$\begin{array}{ccc} M(a) & \longrightarrow & M(c^2a) \\ & \searrow \quad \nearrow & \\ & N(ca) & \end{array} \qquad \begin{array}{ccc} M(ca) & \longrightarrow & M(cb) \\ \uparrow & & \uparrow \\ N(a) & \longrightarrow & N(b) \end{array}$$

$$\begin{array}{ccc} & M(ca) & \\ \nearrow & & \searrow \\ N(a) & \longrightarrow & N(c^2a) \end{array} \qquad \begin{array}{ccc} M(a) & \longrightarrow & M(b) \\ \downarrow & & \downarrow \\ N(ca) & \longrightarrow & N(cb) \end{array}.$$

It shouldn't then be too surprising that functors that are c-approximations of each other have bounded interleaving distance in the log scale [123].

Our setting differs slightly from the one considered above, because for $a, b \in [0, \infty)$ we have a morphism $a \to b$ whenever $a \geq b$, and furthermore our functors are contravariant. We therefore state the definition of c-approximation for $\mathcal{P}^{\mathrm{op}}$-persistent objects $\mathcal{P}^{\mathrm{op}} \to C$:

Definition 4.11.86. Let C be a category, and let $M, N\colon \mathcal{P}^{\mathrm{op}} \longrightarrow C$ be two functors. For any $c \geq 1$ denote by $D_c\colon \mathcal{P} \to \mathcal{P}$ the functor that sends a to ca. Furthermore, denote by $\eta_c\colon D_c \Rightarrow id_{\mathcal{P}}$ the natural transformation given by $\eta_c(a)\colon ca \to a$. A ***c*-approximation** of M and N is a pair of natural transformations

$$\phi\colon M \Rightarrow ND_c$$

and

$$\psi\colon N \Rightarrow MD_c$$

such that $(\psi D_c)\phi = M\eta_{c^2}$ and $(\phi D_c)\psi = N\eta_{c^2}$.

Lemma 4.11.87. Let $M, N\colon \mathcal{P}^{\mathrm{op}} \longrightarrow \mathrm{Ab}$ be two $\mathcal{P}^{\mathrm{op}}$-persistent abelian groups. If there exists a c-approximation between M and N, then

$$\varprojlim_{\epsilon} M(\epsilon) \cong \varprojlim_{\epsilon} N(\epsilon)\,.$$

Proof. Since the functors are contravariant we have an inverse system $(\{M(\epsilon)\}_\epsilon, \{M(\epsilon \to \epsilon')\}_{\epsilon' \geq \epsilon})$, and similarly for N. Let $\phi\colon M \Rightarrow ND_c$ and $\psi\colon N \Rightarrow MD_c$ be the natural transformations which are part of the data of the c-approximation. These induce homomorphisms

$$\bar{\phi}\colon \varprojlim_\epsilon N(\epsilon) \to \varprojlim_\epsilon M(\epsilon)$$

and

$$\bar{\psi}\colon \varprojlim_\epsilon M(\epsilon) \to \varprojlim_\epsilon N(\epsilon)$$

which are inverse to each other. □

The following is joint with John Baez:

Theorem 4.11.88. For all $k = 0, 1, 2, \ldots$ there is an isomorphism

$$\varprojlim_\epsilon H_k(CN(X)(\epsilon)) \cong \mathcal{H}_k(X)\,.$$

Proof. While there is an inclusion map $N(X)(\epsilon) \to V^{\mathrm{sim}}(X)(\epsilon)$ for all $\epsilon \in [0, \infty]$, in general there is an inclusion $V^{\mathrm{sim}}(X)(\epsilon) \to N(X)(c\epsilon)$ only for $c = \infty$, since a p-simplex in $V^{\mathrm{sim}}(X)(\epsilon)$ may have length equal to $p\epsilon$. On the other hand, when computing simplicial homology in dimension k, we only need to consider simplices up to dimension $k+1$, and we will therefore consider the truncations of the simplicial sets $N(X)(\epsilon)_{\leq k}$, given by precomposing $N(X)(\epsilon)\colon \Delta^{\mathrm{op}} \to \mathrm{Set}$ with the inclusion $\Delta^{\mathrm{op}}_{\leq k} \to \Delta^{\mathrm{op}}$, and similarly for $V^{\mathrm{sim}}(X)(\epsilon)$.

The inclusion $N(X)(\epsilon) \to V^{\mathrm{sim}}(X)(\epsilon)$ induces an inclusion

$$\phi_c(\epsilon)\colon N(X)_{\leq k+1}(\epsilon) \to V^{\mathrm{sim}}(X)_{\leq k+1}(c\epsilon)$$

for any $c \geq 1$. If σ is a p-simplex in $V^{\mathrm{sim}}(X)(\epsilon)$, then its length is bounded by $p\epsilon$, and thus there is an inclusion map

$$\psi_{k+1}(\epsilon)\colon V^{\mathrm{sim}}(X)_{\leq k+1}(\epsilon) \to N(X)_{\leq k+1}((k+1)\epsilon)\,.$$

The collection of maps

$$\left\{\phi_{k+1}(\epsilon)\colon N(X)_{\leq k+1}(\epsilon) \to V^{\mathrm{sim}}(X)_{\leq k+1}((k+1)\epsilon) \mid \epsilon \in [0,\infty)\right\}$$

and

$$\left\{\psi_{k+1}(\epsilon)\colon V^{\mathrm{sim}}(X)_{\leq k+1}(\epsilon) \to N(X)_{\leq k+1}((k+1)\epsilon) \mid \epsilon \in [0,\infty)\right\}$$

are easily seen to satisfy the properties of a $k+1$-approximation, as all maps involved are inclusions. Applying homology we obtain a $k+1$-approximation between the functors $H_k(CV^{\mathrm{sim}}(X))$ and $H_k(CN(X))$. We can now use Lemma 4.11.87 and obtain an isomorphism

$$\varprojlim_{\epsilon} H_k(CN(X)_{\leq k+1}(\epsilon)) \cong \varprojlim_{\epsilon} H_k(CV^{\mathrm{sim}}(X)_{\leq k+1}(\epsilon))\,.$$

Since $H_k(CN(X)_{\leq k+1}(\epsilon))$ is equal to $H_k(CN(X)(\epsilon))$ for all k, and similarly for the truncation of the Vietoris–Rips simplicial set, we obtain the claim.

□

Corollary 4.11.89. Let k be a non-negative integer, and let X be a metric space with $\mathcal{H}_k(X) \ncong 0$. Then

$$\varprojlim_{l} H_k(CN(X) \otimes_{\mathcal{P}} A_l) \ncong \varprojlim_{l} H_k(CN(X) \otimes_{\mathcal{P}} A_{[0,l]})\,.$$

That is, under the limit, the kth ordinary and blurred magnitude homology of X are not isomorphic. In particular, for any finite metric space the limits differ for $k = 0$.

Proof. First, note that

$$\varprojlim_{l} H_k(CN(X) \otimes_{\mathcal{P}} A_l) \cong 0$$

since for any $\epsilon' \geq \epsilon$ we have that $CN(X) \otimes_{\mathcal{P}} A_\epsilon \to CN(X) \otimes_{\mathcal{P}} A_{\epsilon'}$ is the zero chain map. By Lemma 4.10.83 we know that $CN(X) \otimes_{\mathcal{P}} A_{[0,l]} \cong CN(X)(l)$, and further by Theorem 4.11.88 we have that $\varprojlim_l H_k(CN(X)(l))$ is the Vietoris homology of X. □

4.12 An Euler characteristic for persistent homology and the persistence function

Analogously as for magnitude homology, we would like to have a statement of the following form:

Conjecture 4.12.90. For any finite quasi-pseudo-metric space X the Euler characteristic of $CN(X)$ exists, and

$$\chi(CN(X)) = \sum_{n=0}^{\infty} (-1)^n \mathrm{rank} H_n(CN(X))\,.$$

This would then allow to define a **persistence function**, by evaluating the series $\chi(CN(X)) \in \mathbb{Z}^{[0,\infty)}$ at $q = e^{-t}$, for $t \in [0,\infty)$. One could then investigate what information about the topology or geometry of the metric space such a function captures, and

how this information changes if instead of the enriched nerve we consider other persistent simplicial sets associated to the metric space. Furthermore, by considering generalised Lawvere metric spaces (by which we mean categories enriched over $[0,\infty)^r$), one would also obtain a new invariant to study multiparameter persistent homology.

It seems reasonable to assume that $CN(X)$ is dualisable, at least for finite ordinary Lawvere metric spaces; the situation might be more delicate for generalised Lawvere metric spaces, especially in the presence of multifiltrations that are not one-critical. However, it is not clear whether we can expect the series

$$\sum_{l\in[0,\infty)}\sum_{n}(-1)^n \mathrm{rank} H_n(NC(X)_l)q^l$$

to have well-ordered support, for comparing the chain complex for the blurred version of magnitude homology with that for the ordinary magnitude homology we would expect to have many more non-zero terms in the sum.

On the other hand, given a simplicial complex $K(X)$ associated to a finite metric space X, the ith persistence module $H_i(K(X)))$ is finitely presented, and such modules can be approximated by modules graded by the poset $(\mathbb{Z},\leq)$ [140, Proposition 2.9]. Thus, by considering simplicial complexes instead of simplicial sets, we might be able to define an Euler characteristic for the chain complex associated to a simplicial complex filtered by the poset $(\mathbb{Z},\leq)$, which would then be a Laurent series, and then define a persistence function as an interpolation of the evaluation of the corresponding series.

4.13 Additional acknowledgments

I started thinking about magnitude homology in August 2017, when John Baez, after hearing Tom Leinster's talk on magnitude homology at the Applied Algebraic Topology conference in Sapporo, had the idea to investigate the relationship between magnitude homology and the persistent homology of a filtered Vietoris–Rips complex associated to a metric space. While we had originally planned to write this manuscript together, John eventually declined to be a coauthor. I started writing a first draft of this manuscript while visiting John at the Centre for Quantum Computations (CQT) at NUS in Singapore. During my time in Singapore we had daily discussions on magnitude homology and how to best relate this to the persistent homology of a filtered Vietoris–Rips complex, and I owe to John much more than giving me the initial motivation for this work; in particular, the name "persistent object" is due to him, and the main idea for the proof of Theorem 4.11.88 is his. I would like to thank the CQT for the generous hospitality, and the organisers of the Applied Algebraic Topology conference for organising such a stimulating and diverse

conference, and for the generous support during my stay in Sapporo. I am grateful to the Emirates Group for sustaining me with an Emirates Award, and to the Alan Turing Institute for awarding me an enrichment scholarship.

Appendix A

Tutorial

In this tutorial, we give detailed guidelines for the computation of persistent homology and for several of the functionalities that are implemented by the libraries in Table 2.2. We first give some advice on how to install the various libraries, and we then give guidelines for how to compute PH for every step of the pipeline in Fig. 2.6. We explain how to compute PH for networks with the weight rank clique filtration (WRCF); for point clouds with the VR, alpha, Čech, and witness complexes; and for image data sets with cubical complexes. We then give guidelines for visualizing the outputs of the computations and for computing the bottleneck and Wasserstein distances with DIONYSUS and HERA. In addition to the bottleneck and Wasserstein distances, there are also other tools (such as persistence landscapes and confidence sets) that are useful for statistical assessment of barcodes, but we do not discuss them here, as there are already comprehensive tutorials [30, 86] for the packages that implement these methods. All MATLAB scripts written for this tutorial are available at `https://github.com/n-otter/PH-roadmap/tree/master/matlab`. In Fig. A.1, we give instructions for how to navigate this tutorial.

Many tears and much sweat and blood were spent learning about the different libraries, writing this tutorial, and the scripts. If you find this tutorial helpful, please acknowledge it.

A.1 Installation

In this section, we give guidelines on how to get and/or install the software packages.

A.1.1 Dionysus

At `http://www.mrzv.org/software/dionysus/get-build-install.html` one can find the code for DIONYSUS, as well as information on dependencies and how to build the library. The library is written in C++, but it also supports python bindings (i.e., there

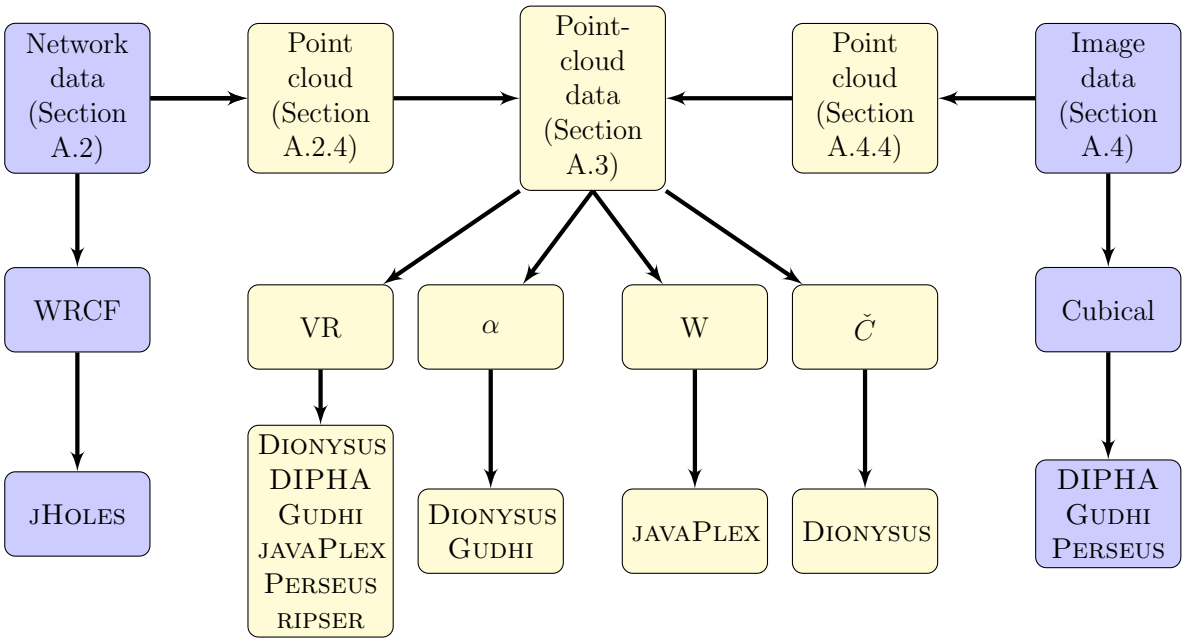


Figure A.1: Schematic for how to navigate the tutorial.

is a python interface to some of the functionalities that are implemented in the library). Depending on the machine on which one is building Dionysus, the python bindings can create some issues. Additionally, from the perspective of performance, it is better to directly use the C++ code. If one wishes to build the library without the python bindings, one needs to delete the `bindings` directory and also to delete the (last) line `add_subdirectory (bindings)` in the file `CMakeLists.txt`.

If one seeks to compute the bottleneck or Wasserstein distances with the library (see Section A.6), then before building the library, one needs to amend a mistake in the bottleneck-distance.cpp script as follows: in the subdirectory `examples`, one finds the file `bottleneck-distance.cpp`. One needs to uncomment the following line (which occurs towards the end of the file):

```
std::cout << "Distance: " << bottleneck_distance(dgm1, dgm2) << std::endl;
```

Now one can build the library as follows (from the directory in which the `CMakeLists.txt` file is):

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

A.1.2 DIPHA

The DIPHA library is available at `https://github.com/DIPHA/dipha`. One can build the library as follows (from the directory in which the `CMakeLists.txt` file is):

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

A.1.3 Gudhi

The GUDHI library is available at `https://gforge.inria.fr/frs/?group_id=3865`. Information about dependencies and how to build the library is available at `http://gudhi.gforge.inria.fr/doc/latest/installation.html`. One can build the library in a similar way as explained for DIPHA.

We note that a python interface was released with the most recent version (at the time of this writing) of the library GUDHI; in this tutorial, we give instructions on how to use the C++ implementation, and we point readers who are familiar with python to the documentation available at `http://gudhi.gforge.inria.fr/python/latest/`.

A.1.4 javaPlex

The JAVAPLEX library does not require installation, and all implementations that we listed in Table 2.2 can be found in the directory `matlab-examples_x.y.z` (where `x.y.z` stands for the version number). This directory, and the accompanying tutorial, can be downloaded at `https://github.com/appliedtopology/javaplex/releases/`. All of the scripts in `matlab-examples_x.y.z` are written in MATLAB.

A.1.5 hera

The HERA library is available at `https://bitbucket.org/grey_narn/hera`. The root folder includes two subfolders: `geom_bottleneck` contains the source code for computing bottleneck distance, and `geom_matching` contains the source code for computing Wasserstein distance.

One can build the library by running the following commands in each of the subfolders `geom_bottleneck/` and `geom_matching/wasserstein`:

```
$ mkdir build
$ cd build
```

```
$ cmake ..
$ make
```

A.1.6 jHoles

Ideally, the JHOLES library should be available for download at `http://cuda.unicam.it/jHoles`. However, this website is often down, so the best way to obtain the library is to contact Matteo Rucco, who is the corresponding author of the companion paper [21]. The library does not require installation or to be built.

A.1.7 Perseus

A compiled version of the PERSEUS library is available at `http://people.maths.ox.ac.uk/nanda/perseus/`. Those wishing to build the library from source code can find the source code at the same website.

A.1.8 ripser

The RIPSER library is available at `https://github.com/Ripser/ripser`. One can build the library by running `make` in the folder that includes the `Makefile`. Note that RIPSER supports several options that can be passed to `make`. See `https://github.com/Ripser/ripser` for more information.

A.2 Computation of PH for networks

In this section, we explain how to compute PH for undirected weighted networks. We represent the nodes of a network with N nodes using the natural numbers $1, \ldots, N$.

A.2.1 Sample network data

To create weighted networks, one can use the script `fractal_weighted.m` available at `https://github.com/n-otter/PH-roadmap/tree/master/matlab/synthetic_data_sets_scripts`. We recall that a fractal network is determined by three non-negative integers n, b and k. The script `fractal_weighted.m` takes four parameters as input: (1) a natural number n (the total number of nodes of the graph is 2^n); (2) a natural number b (the number of nodes of the initial network); (3) a natural number k (this is the connection density parameter); and (4) a string which indicates how weights are associated to edges: this is either `'random'`or `'linear'`.

Example: The command

```
>> fractal_weighted(4,2,1,'random')
```

saves the files `fractal_4_2_1_random.txt` and `fractal_4_2_1_random_edge_list.txt`, where the first file is a text file storing the weighted adjacency matrix of a fractal network with 16 nodes and random weight on every edge, while the second file is a text file storing the weighted edge list of the same network.

A.2.2 Adjacency matrix versus edge-list file

We assume that a network is given either as an adjacency matrix in a MAT-file or as a text file with a list of weighted edges. A typical entry on one line of such a file is a triple "i j w_{ij}", where i and j are the nodes incident to the edge and w_{ij} is the weight of the edge. We call such a file an "edge-list file". We provide the script `adj_matrix_to_edge_list.m` to obtain edge-list files from adjacency matrices. (Note that we also provide the script `edgelist_to_point_cloud_dist_mat.m` to obtain distance matrices from edge-list files, where the distances between nodes are computed using shortest paths.)

A.2.3 PH with the WRCF (with jHoles)

Using edge-list files, we compute PH by constructing the weight rank clique filtration (WRCF) with the library jHoles. Here we give instructions on how to compute PH with Version 3 of jHoles. One needs to run the following command in the terminal:

```
$ java -Xmx<value> -jar jHoles.jar input-file output-file1 output-file2
```

where `-Xmx<value>` is optional and can be used to set the maximum heap size of the garbage collector (we recommend doing this for networks with a large number of nodes or high density). For example, `-Xmx4g` sets the maximum heap size to 4 gigabytes. The file `input-file` is the edge-list file of the network, and `output-file1` is a file in which information (e.g., number of edges, average degree, etc.) about the network is saved, and `output-file2` is the file in which the intervals are saved.

Example: With the command

```
$ java -Xmx4g -jar jHoles.jar fractal_4_2_1_random_edge_list.txt \
fractal_info.txt fractal_intervals.txt
```

one computes PH with the WRCF for the fractal network with parameters $(n, b, k) = (4, 2, 1)$ and random weight on every edge. The persistence diagram is saved in the file `fractal_intervals.txt`, where for every interval also representative cycles are given, while in file `fractal_info.txt` information such as average degree, density, or average

cluster is given. Note that the backslash in the above command indicates that the command continues on the next line, and should therefore be omitted if one writes the whole command on the same line.

A.2.4 Networks as point clouds

One can construe a connected weighted network as a finite metric space and then compute PH by using one of the methods from Section A.3. We now explain how to compute a distance matrix from an undirected weighted network using information about shortest paths between nodes. If two nodes i and j are connected by an edge with weight w, we set the distance between i and j to be $1/w$. Otherwise, we define the distance between i and j to be the minimum of the lengths of all paths between them, where we define the length of a path to be the the sum of the inverse of the weights of the edges in this shortest path. One can compute this distance matrix with the script `shortest_paths.m`. As input, it takes an edge-list file. (See Section A.2.2 for how to obtain an edge-list file from a MAT-file that stores an adjacency matrix.) The output of the script is a text file in which each line gives the entries of a row of the distance matrix. (Note that if a network is not connected, then one sets the distance between nodes in two distinct components of the network to be infinite, and one thereby obtains an extended metric space.)

Example:

```
>> shortest_paths('fractal_4_2_1_random_edge_list.txt')
```

saves the text file `'fractal_4_2_1_random_edge_list_SP_distmat.txt'`.

Additionally, using tools like multidimensional scaling, one can convert a distance matrix into a finite set of points in Euclidean space. This is handy if one wants to use a library that does not support distance matrices as an input type. We provide the script `distmat_to_pointcloud.m` to obtain a point cloud from a distance matrix using multidimensional scaling.

Example:

```
>> distmat_to_pointcloud('fractal_4_2_1_random_edge_list_SP_distmat.txt')
```

A.3 Computation of PH for point clouds

In this section, we explain how to compute PH for finite metric spaces.

A.3.1 Sample point-cloud data

We provide scripts to create point-cloud data. These are available at `https://github.com/n-otter/PH-roadmap/tree/master/matlab/synthetic_data_sets_scripts`. To create points clouds in $\mathbb{R}^3$ and $\mathbb{R}^4$, one can use the scripts `klein_bottle_imm.m` and `klein_bottle_emb.m`, respectively.

Example:

```
>> klein_bottle_imm(5)
```

samples 25 points uniformly at random from the image of the immersion of the Klein bottle in $\mathbb{R}^3$ and saves the point cloud in the text file `klein_bottle_pointcloud_25.txt`, with each line storing the coordinates of one point, as well as in the MAT-file called `klein_bottle_25.mat`.

A.3.2 Distance matrices versus points clouds

Given a finite set of points in Euclidean space, one can compute an associated distance matrix. To get a distance matrix from a point cloud, we provide the script `pointcloud_to_distmat.m`.

Example:

```
pointcloud_to_distmat('klein_bottle_pointcloud_25.txt')
```

computes the distance matrix for the 25 points sampled from the Klein bottle and saves it in the text file `'klein_bottle_pointcloud_25_distmat.txt'`.

Conversely, a distance matrix can yield a finite set of points in Euclidean space by using a method such as multidimensional scaling. We implement such a conversion in the script `distmat_to_pointcloud.m`.

Example:

```
>> distmat_to_pointcloud('klein_bottle_pointcloud_25_distmat.txt')
```

A.3.3 VR complex

A.3.3.1 Input data

The standard input for the construction of the VR complex is a distance matrix. The software packages that take a distance matrix as input are Perseus and DIPHA, but

the other packages do not. Instead, they take a set of points in Euclidean space as input. Note that PERSEUS can also take a set of points in Euclidean space as input, but the implementation does not allow one to set a bound on the upper dimension in the computation of the VR complex, so this implementation is impractical to use for most data sets. We next compute the VR complex with each library. With most of the libraries, one has to indicate the maximum filtration value for which one wants to compute the filtered simplicial complex. We set this value to the maximum distance between any two points. Note, however, that a smaller value often suffices. To compute the maximum distance given a text file `input-file` with the coordinates of a point on each line, one can type the following in MATLAB:

```
>> A=load(input-file);
>> D=pdist(A);
>> D=squareform(D);
>> M=max(max(D));
```

Example:

```
>> A = load('klein_bottle_pointcloud_25.txt');
>> D=pdist(A);
>> D=squareform(D);
>> M=max(max(D));
>> M

M =

    7.0513
```

The maximum distance is given by `M`. Similarly, if one is given a text file `input-file` that stores a distance matrix, then one can compute the maximum distance by typing

```
>> D=load(input-file);
>> M=max(max(D));
```

A.3.3.2 Dionysus

The library DIONYSUS takes a point cloud as input. We can use either the standard or dual algorithm. For the former, we use the command

```
$ ./rips-pairwise -s max-dimension -m max-distance \
-d output-file input-file
```

where the file `./rips-pairwise` is in the `dionysus/build/examples/rips` directory, further `max-dimension` is the maximum dimension for which we want to compute the simplicial complex, `max-distance` is the maximum parameter value for which we want to compute the complex, `output-file` is the file to which the intervals are written, and `input-file` is a text file with the coordinates of a point on each line. To compute the intervals with the dual algorithm, we use the command

```
$ ./rips-pairwise-cohomology -s max-dimension -m max-distance \
-p prime -d output-file input-file
```

where the file `./rips-pairwise-cohomology` is in the directory named `dionysus/build/examples/cohomology`, further `prime` is a prime number and indicates the coefficient field for the computation of cohomology. (For the standard algorithm, there is no choice: one must use $p = 2$.)

Example:

```
$  ./rips-pairwise-cohomology -s 3 -m 7.0513 -p 2 \
-d klein_bottle_25_output.txt  klein_bottle_pointcloud_25.txt
```

A.3.3.3 DIPHA

The DIPHA library reads and writes to binary files. One can convert the text file that stores the distance matrix (see Section A.3.2 for how to obtain a distance matrix from a point cloud) into a binary file of the right input type for DIPHA by using the file `save_distance_matrix.m` provided by the developers of DIPHA, which can be found in `dipha-master/matlab`.

Example:

```
>> D=load('klein_bottle_pointcloud_25_distmat.txt');
>> save_distance_matrix(D,'kleinbottle_25.bin');
```

To run DIPHA using a single process, one types the command

```
$ ./dipha [options] --upper_dim d input-file output-file
```

where options include `--benchmark` to display profiling information and `--dual` to run the dual algorithm. (The default is the standard algorithm.) To run DIPHA on more than one process, one uses the command

```
$ mpiexec -n N dipha options --upper-dim d input-file output_file
```

where `N` is the number of process and the above options are again available.

Example:

```
$ ./dipha --benchmark --dual --upper_dim 3 klein_bottle_25.bin \
 klein_bottle_out.bin
```

A.3.3.4 Gudhi

The library GUDHI takes a point cloud as input. We run the command

```
$ ./rips_persistence -r max-distance -d max-dimension \
-p prime -o output-file input-file
```

where `max-distance`, `max-dimension`, and `prime` are as above, `output-file` is a file to which the intervals are written, and `input-file` is a text file with the coordinates of a point on each line.

Example:

```
$ ./rips_persistence -r 7.0513 -d 3 -p 2 -o klein_bottle_25_output.txt \
 klein_bottle_pointcloud_25.txt
```

A.3.3.5 javaPlex

The library JAVAPLEX takes a point cloud as input. Before starting using this library one has to run the script `load_javaplex.m` which is located in the JAVAPLEX directory `matlab_examples`. We wrote the script `vietoris_rips_javaplex.m` to compute PH with the VR complex in JAVAPLEX. This script takes four parameters as input: (1) the name of the text file with the point cloud; (2) the maximum dimension for which we want to compute the simplicial complex; (3) the maximum filtration step for which we want to compute the VR complex; and (4) the number of filtration steps for which we compute the VR complex.
The script saves text files containing the barcode intervals, one file for each homological dimension. These are the files ending with `i_right_format.txt` where i indicates the homological dimension.

Example:

```
>> vietoris_rips_javaplex('klein_bottle_pointcloud_25.txt',7.0513,3,20);
```

A.3.3.6 Perseus

The library PERSEUS takes a distance matrix as input (see Section A.3.2 for how to obtain a distance matrix from a point cloud). One has to prepare the input file by adding two lines at the beginning of the file that stores the distance matrix. We do this as follows:

```
N
first-step step-increment steps max-dimension
d11 d12 ...
d21 d22 ...
...
```

where `N` is the number of points (and hence the number of rows (or columns) in the distance matrix), `first-step` is the value for the first filtration step, `step-increment` is the step size between any two filtration steps, `steps` is the number of total steps, and `max-dimension` is the maximum dimension for which we compute the complex. We now can compute PH with the command

```
$ ./perseus distmat input-file output-file
```

in the terminal, where `input-file` is the name of the input file and `output-file` is the name of the file in which the intervals will be saved. PERSEUS creates a series of files named `output-file_i.txt` for $i \in \{0, 1, \dots\}$, where `output-file_i.txt` contains the intervals for the homological degree i.

Example:

```
$ ./perseus distmat klein_bottle_25.txt klein_bottle_25_output.txt
```

where these are the first two lines of the file `klein_bottle_25.txt`:

```
25
0 0.1 71 3
```

A.3.3.7 ripser

The library RIPSER takes both a point cloud and a distance matrix as input, and it supports four different format types for the distance matrix. (See `https://github.com/Ripser/ripser#description` for more details.) One of the supported input types for the distance matrix is the format accepted by DIPHA (see Section A.3.3.3).

We run the command

```
$ ./ripser --format input-type --dim max-dimension [options] input-file
```

where `input-type` is a string that indicates the type of the input, `max-dimension` is the maximum dimension of persistent homology that is computed (note the difference with respect to the other libraries, for which one indicates the maximum dimension of the complex), and options includes `-- modulus p` (with which one can choose the coefficient field $\mathbb{F}_p$). Note that one has to enable this option at compilation (see Section A.1.8). The output of the computation is written to the standard output.

Example:

```
$ ./ripser --format dipha --dim 2 klein_bottle_25.bin > \
klein_bottle_25_out.log
```

where `klein_bott_25.bin` is the input file from Section A.3.3.3 and the standard output is saved to the file `klein_bottle_25_out.log`.

A.3.4 Alpha

In this section, we explain how to compute PH with the alpha complex with Dionysus and Gudhi.

A.3.4.1 Dionysus

One can compute PH with the alpha complex for finite subsets of points in $\mathbb{R}^2$ or $\mathbb{R}^3$. For points clouds in $\mathbb{R}^2$, one runs the command

```
$ ./alphashapes2d < input-file > output-file
```

where `input-file` a text file with coordinates of a point in $\mathbb{R}^2$ on each line and `output-file` is the file to which the intervals in the persistence diagram are written. For points clouds in $\mathbb{R}^3$, one runs the command

```
$ ./alphashapes3d-cohomology input-file output-file
```

where `input-file` and `output-file` are as above.[1]

Example:

```
$ ./alphashapes3d-cohomology klein_bottle_pointcloud_25.txt \
klein_bottle_25_output.txt
```

[1] There is also a script `|./alphashapes3d` , but this script has a bug and does not compute.

A.3.4.2 Gudhi

The program GUDHI supports both points clouds in $\mathbb{R}^2$ and $\mathbb{R}^3$. To compute PH with the alpha complex one can use the script `./alpha_complex_persistence` which is in the folder `example/Persistent_cohomology`. The script takes as input an OFF file, as decribed here `http://www.geomview.org/docs/html/OFF.html`. Namely, the first lines of the input file are as follows:

```
OFF
embedding-dimension V 0 0
x11 x12 ... x1d
x21 x22 ... x2d
...
```

where `embedding-dimension` is the dimension d of the Euclidean space, `V` is the number of points, and all other lines store coordinates `xi1`,..., `xid` of the points.

One then computes PH by running the following command in the terminal

```
$ ./alpha_complex_persistence -p prime -o output-file input-file
```

where `p` is a prime number and indicates that one does computations over the coefficient field $\mathbb{F}_p$.

Example:

```
$ ./alpha_complex_persistence  -p 2 -o klein_bottle_25_output.txt \
klein_bottle_25_input.txt
```

where the first two lines of the file `klein_bottle_25_input.txt` are as follows:

```
OFF
3 25 0 0
```

A.3.5 Čech

One can compute PH with the Čech complex for a point cloud in Euclidean space using the implementation in DIONYSUS. One runs the command

```
$ ./cech-complex < input-file > output-file
```

where `input-file` is a text file of the following form:

```
embedding-dimension max-dimension
x11 x12 ... x1d
x21 x22 ... x2d
...
```

where `embedding-dimension` is the dimension d of the Euclidean space, `max-dimension` is the dimension up to which we compute the complex, and all other lines store coordinates `xi1`,..., `xid` of the points.

Example:

```
$ ./cech-complex < klein_bottle_25_input.txt > klein_bottle_output.txt
```

where the first line of the file `klein_bottle_25_input.txt` is as follows:

```
3 3
```

A.3.6 Witness

One can compute the witness complex using javaPlex. Recall that before starting using this library one has to run the script `load_javaplex.m` which is located in the javaPlex directory `matlab_examples`. Given a point cloud S, the witness complex is a simplicial complex constructed on a subset $L \subseteq S$ of so-called "landmark" points. As we explained in Section 2.5.2.4, there are several versions of the witness complex. The ones implemented in javaPlex are the weak Delaunay complex, which is also just called the "witness complex", and parametrized witness complexes, which are also known as "lazy witness complexes". Given a point cloud L, one can compute the witness complex or lazy witness complex using the scripts `witness_javaPlex.m` and `lazy_witness_javaPlex.m`.

The script `witness_javaPlex.m` takes five parameters as input: (1) the name of the text file with the point cloud; (2) the maximum dimension for which we want to compute the simplicial complex; (3) the number of landmark points; (4) how the landmark points are selected (this is either `'random'` or `'maxmin'`); and (5) the number of filtration steps for which we compute the complex.

The script `lazy_witness_javaPlex.m` takes six parameters as input: (1) the name of the text file with the point cloud; (2) the maximum dimension for which we want to compute the simplicial complex; (3) the number of landmark points; (4) how the landmark points are selected (this is either `'random'` or `'maxmin'`); (5) the value for the parameter ν; and (6) the number of filtration steps for which we compute the complex.

See the scripts for further details on input parameters, and see Section 2.5.2.4 and the javaPlex tutorial [3] for further detail on witness complexes.

Example:

```
lazy_witness_javaPlex('klein_bottle_pointcloud_25.txt',3,10,'random',2,20)
```

A.4 Computation of PH for image data

In this section, we discuss how to compute PH for image data using cubical complexes. The packages DIPHA, Perseus, and Gudhi support the construction of filtered cubical complexes from grey-scale image data. As an example of grey-scale image data, we use the data set "Nucleon" from the Volvis repository [1]. This is a 3-dimensional grey-scale image data set; one is given a 3-dimensional lattice of resolution $41 \times 41 \times 41$, where each lattice point is labeled by an integer that represents the grey-scale value for the voxel anchored at that lattice point. The .raw data file from [1] is binary, and it stores 8 bits for each voxel. We read the .raw data file in MATLAB as follows:

```
>> fileID=fopen('nucleon.raw','r');
>> A=fread(fileID,41*41*41,'int8');
>> B=reshape(A,[41 41 41]);
```

so `B` is a 3-dimensional array of size $41 \times 41 \times 41$ that stores the grey-scale values.

Note for this example that the cubical complex constructed in DIPHA and Gudhi has dimension 3 and size 531441, while the cubical complex constructed with Perseus has dimension 3 and size 571787. This is because DIPHA and Gudhi implement the optimized way to represent a cubical complex that was introduced in [202]. However, all three libraries implement the same algorithm for the computation of PH from cubical complexes. When interpreting the results of the computations with Gudhi and Perseus, one needs to take into account the rescaling of the grey values (see Section A.4.2).

A.4.1 DIPHA

To save the array in a file that can be given as input to DIPHA, one can use the MATLAB script `save_image_data.m` provided by the developers of DIPHA, which can be found in `dipha-master/matlab`. One gives the array `B` as input and a name for the input file. One then proceeds in a similar way as for the computation of the VR complex (see Section A.3.3).

Example:

```
>> save_image_data(B,'nucleon.bin');
$ ./dipha --benchmark nucleon.bin nucleon_out.bin
```

A.4.2 Perseus

To compute PH with cubical complexes with PERSEUS, one needs to rescale the grey values so that all grey values are positive, because PERSEUS does not allow cells to have negative birth times. We wrote the script `save_image_data_perseus.m` to save the array in a file that can be given as input to PERSEUS. This script takes the array `B` as input and a name for the input file for PERSEUS.

Example:

```
>> save_image_data_perseus(B,'nucleon.txt');
```

To compute PH with PERSEUS one then runs the following command:

```
$ ./perseus cubtop input-file output-file
```

where `input-file` is the text file prepared with the script `save_image_data_perseus.m`, and `output-file` is the name of the text file to which the barcode intervals are written.

Example:

```
$ ./perseus cubtop nucleon.txt nucleon_output.txt
```

A.4.3 Gudhi

To compute PH with GUDHI, one can use the same input file as for PERSEUS. We run the command

```
$ ./Bitmap_cubical_complex input-file
```

and the output is then saved to a file with name `input-file_persistence`.

A.4.4 Images as point clouds

As we discussed in Section 2.5.1.2, one can construe a collection of images as a metric space, and one can then apply the methods for computing the PH for point clouds that we discussed in Section A.3.

A.5 Barcodes and persistence diagrams

Once we have computed the intervals, we plot the barcodes and persistence diagrams. The format of the output files varies widely across the different packages. To address this issue and to interpret the results of the computations, we first need to change the format of the output files to a common format. In the unified format, each homological dimension has an accompanying text file in which we store the intervals. In this file, the entry in line i has the form

$$x_i \; y_i \, ,$$

where x_i is the left endpoint of the interval and y_i is the right endpoint. If the interval is infinite, we set $y_i = -1$.

- DIONYSUS: We provide the script `dionysus_reformat_output.m` to obtain the right format. This script takes two parameters as input, namely the name of the text file to which the ouput of the computations with DIONYSUS were stored, and a string of five letters indicating the type of file: "dcech" for the output of PH computation with the Čech complex; "alpha" for the output of PH computation with the alpha complex; "VR-st" for the output of PH computation with a Vietoris–Rips complex and the standard algorithm; and "VR-co' for the output of PH computation with a Vietoris–Rips complex and the dual algorithm.

- DIPHA: We provide the script `dipha_reformat_output.m` to obtain the right format. The script takes as input the name of the binary file to which the ouput of the computations with DIPHA were stored.

- GUDHI: We provide the script `gudhi_reformat_output.m` to obtain the right format. The script takes as input the name of the text file to which the output of the computations with GUDHI are stored.

- JAVAPLEX: We wrote the three scripts `vietoris_rips_javaplex.m`, `lazy_witness_javaPlex.m`, and `witness_javaPlex.m`, which have been written to give this type of output.

- JHOLES: We provide the script `jholes_reformat_output.m` to obtain the right format. The script takes as input the name of the text file to which the barcode intervals obtained with JHOLES were stored.

- PERSEUS: The output is already in the right format.

- RIPSER: The script `ripser_reformat_output.m` gives the right format.

Example:

```
>> dionysus_reformat_output('klein_bottle_25_output.txt','alpha')
```

creates the three files `klein_bottle_25_output_0.txt`, `klein_bottle_25_output_1.txt`, and further `klein_bottle_25_output_2.txt`, each storing intervals for PH in dimension 0, 1, and 2, respectively.

We can then plot barcodes using the script `plot_barcodes.m` and plot persistence diagrams using the script `plot_pdg.m`. Both scripts take as inputs (1) the name of a text file storing the intervals for PH in a certain dimension and (2) the title for the plot.

Example:

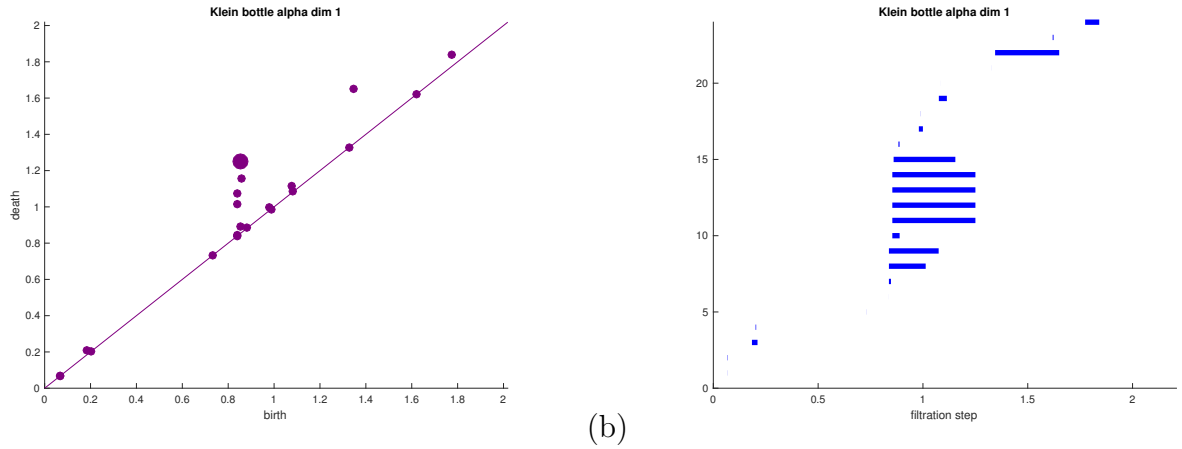
```
>>  plot_pdg('klein_bottle_25_output_1.txt','Klein bottle alpha dim 1')
```

produces the plot in Fig. A.2(a) and saves it as a .pdf file `klein_bottle_25_output_1.pdf`.

Example:

```
>>  plot_barcodes('klein_bottle_25_output_1.txt','Klein bottle alpha dim 1')
```

produces the plot in Fig. A.2(a) and saves it as a .pdf file.



(a) (b)

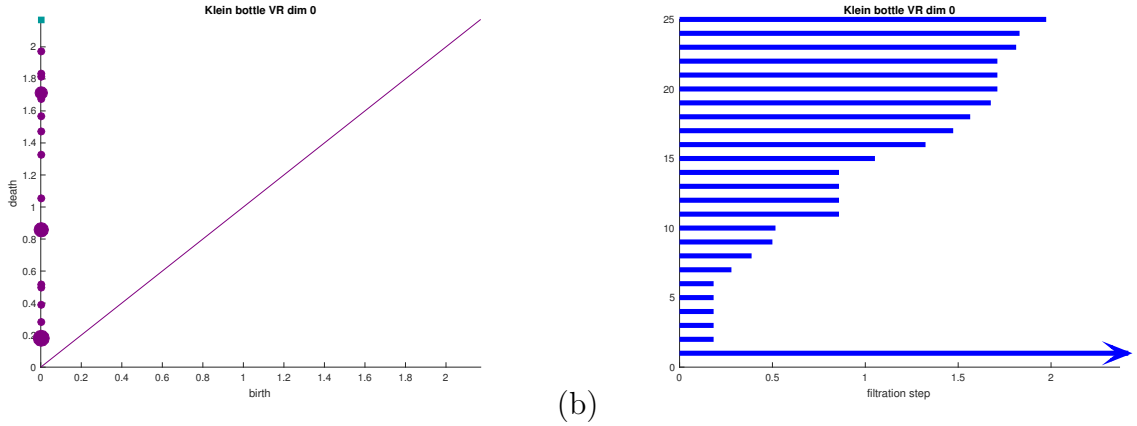
Figure A.2: Visualisation of output for the computation of PH in dimension 1 with the alpha complex (with DIONYSUS) for 25 points sampled uniformly at random form the Klein bottle: (a) persistence diagram and (b) barcode.

In the plots in Fig. A.2, there are no infinite intervals, so we give an additional example to illustrate how infinite intervals are plotted with our scripts.

Example:

```
>> gudhi_reformat_output('klein_bottle_25_output.txt')
>> plot_barcodes('klein_bottle_25_output_0.txt','Klein bottle VR dim 0')
>> plot_pdg('klein_bottle_25_output_0.txt','Klein bottle VR dim 0')
```

produces the two plots in Fig. A.3.



(a) (b)

Figure A.3: Visualisation of output for the computation of PH in dimension 0 with the VR complex (with GUDHI) for 25 points sampled uniformly at random form the Klein bottle: (a) persistence diagram and (b) barcode. Infinite intervals are represented by a square in the persistence diagram, by an arrow in the barcode plot.

A.6 Statistical interpretation of barcodes

Once one has computed barcodes, one can interpret the results using available implementations of tools (such as bottleneck distance, Wasserstein distance, and persistence landscapes) that are useful for statistical assessment of barcodes. In this section, we give instructions for how to compute the bottleneck and Wasserstein distances with DIONYSUS and HERA. See the tutorial for the Persistence landscape toolbox [30] and the TDA package [86] for instructions on how to use these packages. For ease of reference, we recall the definition of Wasserstein distance from Section 2.5.4:

Definition A.6.91. Let $p \in [1, \infty]$. The *pth Wasserstein distance* between X and Y is defined as

$$W_p[\mathrm{d}](X, Y) := \inf_{\phi\colon X \to Y} \left[\sum_{x \in X} \mathrm{d}[x, \phi(x)]^p \right]^{1/p}$$

for $p \in [1, \infty)$ and as

$$W_\infty[\mathrm{d}](X, Y) := \inf_{\phi\colon X \to Y} \sup_{x \in X} \mathrm{d}[x, \phi(x)]$$

for $p = \infty$, where d is a metric on $\mathbb{R}^2$ and ϕ ranges over all bijections from X to Y.

A.6.1 Bottleneck distance

The bottleneck distance is the Wasserstein distance for $p = \infty$ and $\mathrm{d} = L_\infty$ (see Definition A.6.91).

A.6.1.1 Dionysus

To compute the bottleneck distance between two barcodes, one can use the script named `bottleneck-distance.cpp` (appropriately modified as explained in Section A.1.1) in the DIONYSUS subdirectory `examples`. This script requires right endpoints of infinite intervals to be denoted by `inf` ; additionally, if there are intervals of length 0, the script will compute the wrong distance. To make sure that no intervals of length 0 are in the input files and that the intervals of infinite length are in the right format, one can use the script `bottleneck_dionysus.m`. This script takes as input two text files corresponding to two persistence diagrams in the unified format (see Section A.5), with one interval per line, as follows:

```
>> bottleneck_dionysus('pdg1','pdg2')
```

and saves the persistence diagrams to two files called `diagram1.txt` and `diagram2.txt` to the current directory. Now one can compute the bottleneck distance as follows:

```
$ ./bottleneck-distance diagram1.txt diagram2.txt
```

A.6.1.2 hera

To compute the bottleneck distance with HERA, one can use the script `bottleneck_dist` in the subdirectory `geom_bottleneck/build/example`. This script requires right endpoints of infinite intervals to be denoted by `-1` ; this corresponds to the convention in the unified format (see Section A.5). One can compute the bottleneck distance as follows:

```
$ ./bottleneck_dist diagram1.txt diagram2.txt error
```

where `diagram1.txt` and `diagram2.txt` are two text files corresponding to two persistence diagrams in the unified format, and `error` is a nonnegative real number that is an optional input argument. If `error` is nonzero, instead of the exact distance, an approximation to the bottleneck distance with relative error `error` will be computed. (See the explanation in [122].) This option can be useful when dealing with persistence diagrams that include many off-diagonal points, as it can speed up computations.

A.6.2 Wasserstein distance

With DIONYSUS, one can compute the Wasserstein distance for $\mathrm{d} = L_\infty$ and $p = 2$, and one can compute this distance for other values of p with a straightforward modification of the source code. With HERA, one can compute the Wasserstein distance for any choice of metric $\mathrm{d} = L_q$, with $q \in [1, \ldots, \infty]$, for any $p \in [1, \infty)$.

A.6.2.1 Dionysus

The script `bottleneck-distance.cpp` computes both the bottleneck distance and the Wasserstein distance for $\mathrm{d} = L_\infty$ and $p = 2$, so one can follow the instructions in A.6.1.1 to compute the Wasserstein distance for these choices. If one wishes to compute the Wasserstein distance for other values of p, one has to modify the script named `bottleneck-distance.cpp` as follows. Towards the end of the file, in the line

```
std::cout << "L2-Distance: " << wasserstein_distance(dgm1, dgm2, 2)
<< std::endl;
```

one can substitute the third input of the script `wasserstein_distance` with any number $p \in [1, \infty)$.

A.6.2.2 hera

With HERA, one can compute the approximate Wasserstein distance discussed in [122]. One can use the script `wasserstein_dist` in the subdirectory called `geom_matching/wasserstein/build`. This script requires right endpoints of infinite intervals to be denoted by `-1` ; this corresponds to the convention in the unified format (see Section A.5). One can compute the approximate Wasserstein distance as follows:

```
$ ./wasserstein_dist power error distance diagram1.txt diagram2.txt
```

where `power` is the value for p, `error` is the relative error, `distance` is the value for q (where $\mathrm{d} = L_q$ is the employed distance), and `diagram1.txt` and `diagram2.txt` are two text files corresponding to two persistence diagrams in the unified format.

Appendix B

Additional computations

We report additional computations that we perform with the libraries that we study.

B.1 Vietoris–Rips complex

In this section, we report additional results for the computation of PH with the VR complex, as explained in Section 2.7. In Tables B.1–B.4, we give results for the computations on a cluster; in Table B.5, we give results for the computations on a shared-memory system. For the data set **Vicsek**, we computed PH for six different point clouds, as we also explain in the main paper. We report results only for one of these points clouds, because the results of the computations — in terms of wall-time seconds and memory used — were all similar. The point cloud for which we report the results has the following parameters (see the main manuscript for a discussion of these parameters): $N = 300$, $l = 5$, $\theta_0 = 1$, $v_0 = 0.03$, $\eta = 0.1$, and $T = 600$. Of the 600 different points clouds (one for each time step t), we choose the one that corresponds to time step $t = 300$.

Table B.1: Computations of PH with VR complex for the data set **drag 1**. The size of the complex is 1.7×10^8, and the dimension is 2. We run DIPHA on one node and 16 cores.

	Wall-time seconds	CPU seconds	Memory in GB
javaPlex (st)	-	-	> 64
Dionysus (st)	-	-	-
DIPHA (st)	7356	117417	2.4
Perseus	-	-	-
Dionysus (d)	4360	4362	16.8
DIPHA (d)	75	1176	1.8
Gudhi	1524	1517	7.9
ripser	2	1	0.04

Table B.2: Computations of PH with VR complex for the data set **house**. The size of the complex is 1.6×10^9, and the dimension is 3. We run DIPHA (d) on three nodes of 16 cores, and DIPHA (st) on four nodes of 16 cores.

	Wall-time seconds	CPU seconds	Memory in GB
javaPlex (st)	-	-	> 64
DIPHA (st)	53686	243587	5.3
Perseus	-	-	-
Dionysus (d)	-	-	-
DIPHA (d)	1450	22981	7.1
Gudhi	46377	25925	64.6
ripser	11	11	0.8

Table B.3: Computations of PH with VR complex for the data set **senate**. The size of the complex is 4.6×10^6, and the dimension is 3. We run DIPHA on two nodes and 16 cores.

	Wall-time seconds	CPU seconds	Memory in GB
javaPlex (st)	222	571	< 5
DIPHA (st)	5	58	0.1
Perseus	457	457	4.9
Dionysus (d)	183	183	0.5
DIPHA (d)	5	34	0.1
Gudhi	51	51	0.2
ripser	1	1	0.01

Table B.4: Computations of PH with VR complex for the data set **netw-sc**. The size of the complex is 8.5×10^8, and the dimension is 3. We run DIPHA on two nodes of 16 cores.

	Wall-time seconds	CPU seconds	Memory in GB
javaPlex (st)	-	-	> 64
DIPHA (st)	16004	202755	5.6
Perseus	-	-	-
Dionysus (d)	-	-	-
DIPHA (d)	618	9842	5.8
Gudhi	13483	13465	40.6
ripser	9	9	0.5

B.2 Image data set

In this section, we report timings and memory usage for the computation of PH with cubical complexes with the libraries DIPHA, Perseus, and Gudhi for the data set

Table B.5: Computations on the shared-memory system.

(a) Computations on the shared-memory system: wall-time seconds

Data set	**H3N2**	**Vicsek**	**drag 1**	**fract l**
Size of complex	3.4×10^9	3.3×10^8	1.7×10^8	2.8×10^9
Max. dim.	2	3	2	3
javaPlex (st)	-	26257	19054	-
Perseus	-	-	-	-
Dionysus (d)	-	4637	2889	593528
DIPHA (d)	14362	369	1775	3920
Gudhi	3377	433	144	4154
ripser	24	3	2	14

(b) Computations on the shared-memory system: memory in GB

Data set	**H3N2**	**Vicsek**	**drag 1**	**fract l**
Size of complex	3.4×10^9	3.3×10^8	1.7×10^8	2.8×10^9
Max. dim.	2	3	2	3
javaPlex (st)	> 700	< 600	< 600	> 700
Perseus	-	-	-	-
Dionysus (d)	-	33.5	16.8	269
DIPHA (d)	276.1	32.5	13.8	276.1
Gudhi	158.2	15.9	7.9	134.5
ripser	0.2	0.3	0.04	1.2

vertebra. For this data set, the standard implementation in DIPHA performs better in terms of wall time, CPU seconds, and memory usage than the dual implementation. This does not necessarily contradict the fact that the dual implementation gives a heuristic speed-up with respect to the standard implementation, because it is well-known that the dual implementation gives a speed-up when one computes PH with the VR complex but not in general.

Table B.6: Computations of PH with cubical complexes for the data set **vertebra** on the shared-memory system. The complex, as constructed by DIPHA and Gudhi, has 1.1×10^9 cells, and its dimension is 3. Note that Perseus implements a less efficient way for building filtered cubical complexes from grey-scale images, and the resulting complex has more cells than the one constructed by DIPHA and Gudhi.

	DIPHA (d)	DIPHA (st)	Perseus	Gudhi (d)
Wall-time seconds	2115	1793	19604	3668
Memory in GB	82.1	80.9	628.8	59.8

B.3 Additional functionalities

In this section, we report timings and memory usage for the computation of PH with the alpha (α), Čech (Č), weak witness complex (W), and parametrised witness complex for $\nu = 2$ (W_2) (see the main text for a description of these complexes). For the witness complexes we choose the landmark points uniformly at random. We use the two data sets consisting of point clouds in $\mathbb{R}^3$ (i.e., the data sets **Klein** and **Vicsek**). Additionally, note that the newest version of Gudhi (Version 1.3.1) implements the computation of witness complexes, but it does not implement the computation of filtered witness complexes.

Table B.7: Computations of alpha and Čech complexes on the shared-memory system.

(a) Time and memory usage for the data set **Klein**.

	Dionysus α	Dionysus Č	Gudhi α
Size of complex	9.2×10^3	-	9.2×10^3
Wall-time seconds	1	-	2
Memory in GB	0.008	-	0.006

(b) Time and memory usage for the data set **Vicsek**. For the witness complexes we choose 100 landmark points uniformly at random.

	Dionysus α	Dionysus Č	Gudhi α
Size of complex	7.7×10^3	-	7.7×10^3
Wall-time seconds	1	-	2
Memory in GB	0.007	-	0.006

Table B.8: Computations of PH with the witness complex on the shared-memory system.

(a) Time and memory usage for the data set **Klein**. We choose 100 landmark points uniformly at random.

	javaPlex W	javaPlex W_2
Size of complex	6.8×10^5	2.8×10^6
Wall-time seconds	32	328
Memory in GB	< 2	< 5

(b) Time and memory usage for the data set **Vicsek**. For the witness complexes, we choose 100 landmark points uniformly at random.

	javaPlex W	javaPlex W_2
Size of complex	1×10^6	4.1×10^6
Wall-time seconds	50	84
Memory in GB	< 3	< 5

Appendix C

Code

We have written code which takes as input a simplicial complex and one-critical bifiltration, and builds the associated chain complex in the `Macaulay2` package of Grayson and Stillman, available at: `http://www.math.uiuc.edu/Macaulay2/`. The scripts are available at: `http://www.math.iastate.edu/hschenck/MPH.m2`. Note that this code can easily be adapted to the case $r > 2$ (so not necessarily a bifiltration). One important note: in `Macaulay2`, the degrees for a multigraded ring start, by default, with index zero, so output for multigraded Hilbert series always appears with variables T_0, T_1. In particular, when reading Hilbert series computations below and comparing to the body of the paper, $T_i \mapsto t_{i+1}$.

C.1 Computation of homology of chain complex

Here we provide a code that takes as input a simplicial complex together with a one-critical bifiltration and builds the associated chain complex.

```
loadPackage "SimplicialComplexes"

--Turn a row vector into a list.
rvectTolist = (M)->(apply(rank source M, i->M_(0,i)))

--Tor every face, we need to assign it the multidegree
--in which it appears. For now, just work with a Z^2 grading.

AllFaces=(SC)->(apply((dim SC)+2, i->(rvectTolist faces(i,SC))))

WeightedFaces = (SC, W)->(FF=flatten AllFaces(SC);
                          FW=flatten W;
                          apply(#FF, i->({FF#i, FW#i})))

--Build the ith differential. Use the differential from the simplicial complex,
```

```
--and the hash table with the weightings.

WD = (SC,W,i)->(Fi = faces(i,SC);
                Fim = faces(i-1,SC);
                f=rank source Fi;
                g=rank source Fim;
                apply(g,m->(apply(f,n->(if ((CCSC.dd_i)_(m,n) !=0) then
                (y_1^(HT#(Fi_(0,n))#0-HT#(Fim_(0,m))#0)*
                y_2^(HT#(Fi_(0,n))#1-HT#(Fim_(0,m))#1))*
                ((CCSC.dd_i)_(m,n)) else 0_R1)))))


 ---Build the chain complex for a bifiltration.

      BuildCC2 =(SC,W)->(R1=R[y_1,y_2];
                  CCSC = chainComplex SC;
                  d = dim(SC)+1;
                  HT=hashTable WeightedFaces(SC,W);
                  listdiffs=apply(d, i->(WD(SC,W,i+1)));
                  S=ZZ/31991[x_1,x_2, Degrees=>{{1,0},{0,1}}];
                  tidyup=map(S,R1, matrix{{x_1,x_2}});
                  listdiffsS=apply(listdiffs, i->apply(i, j->
                  apply(j, k->tidyup k)));
                  diffs1 = apply(listdiffsS, i->(matrix i)**S);
                  --Top differential has target ZZ^n instead of R1^n, just tensor
                  --everything together
                  Clist = apply(W, j->directSum(apply(j, i->S^{-i})));
                  diffs2 = apply((#Clist-1), i->(map(Clist_i, Clist_(i+1),
                  diffs1#i)));
                  chainComplex(diffs2))
```

We illustrate how to build a chain complex, and then how to compute a minimal presentation of the first homology for the bifiltration in Example 3.3.22.

```
i1 : R=ZZ/31991[a..f];

-- I is the monomial ideal of the non-edges
i2 : I = monomialIdeal (a*c,a*d,a*e,b*d,b*e,c*f,d*f);

-- W is a list of lists: the first list  stores the entry degrees of the vertices,
-- the second list the entry degrees of the edges
-- and the third list  the entry degrees of the faces.
-- Vertices, edges and faces are ordered by lexicographical order.
i3 : W={{{1,1},{0,0},{0,0},{1,0},{0,0},{0,0}},
   {{1,2},{1,2},{2,2},{0,2},{1,0},{0,1},{1,0},{2,0}},
   {{2,2},{3,1}}};

--Build the simplicial complex
```

```
i4 : K=simplicialComplex I;
i5 : AllFaces K;

-- Build the chain complex
i6 : Kcomplex=BuildCC2(K,W);

--Compute a minimal presentation of the first homology of the chain complex

i7 : prune(HH_1(Kcomplex))

o7 = cokernel {1, 1} | x_1^2 0   |
              {1, 2} | 0     x_1 |
              {2, 2} | 0     0   |

                            3
o7 : S-module, quotient of S
```

For space reasons we only report the ouput of the computation of the minimal presentation of the module; note that the command `Kcomplex=BuildCC2(K,W);` gives the same chain complex as in Example 3.3.22.

C.2 Hilbert function and series, local cohomology

Here we show how one can compute the Hilbert function and series, and display the Hilbert function as a table.

```
--Input M a bigraded module. Output is a 6x6 grid displaying
--(low degree) values of the bigraded Hilbert function, and the
--bigraded Hilbert series. Specific to the 2D case. Also for
--the 3 valuse I in {{x_1},{x_2},{x_1,x_2}} compute  H^0_I(M).

Dis = (M)->(print matrix table(6,6,
                  T=(i,j)->hilbertFunction({5-i,j},M));
             print hilbertSeries M;
             print "H^0_I(M) for all associated primes";
             R=ring(M);
             Vlist=rvectTolist vars(R);
             AllPrimes = drop((subsets Vlist),1);
             --get all subsets of vars, drop the empty one
             H0I=apply(AllPrimes, i-> saturate(0_M, ideal(i)));
             --compute H^0_I(M) for all associated primes
             scan(#AllPrimes, i->(print AllPrimes_i;
                          PH0I = prune H0I_i;
                          print PH0I;
                          if PH0I =!=0_M then print hilbertSeries H0I_i)))
```

For instance, for the bifiltration in Example 3.3.22 we let M be its homology in degree 1 and obtain:

```
i22 : Dis M
| 0 0 1 1 1 1 |
| 0 0 1 1 1 1 |
| 0 0 1 1 1 1 |
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 0 0 0 0 0 |
          2    3
T T  + T T  - T T
 0 1    0 1    0 1
------------------
 (1 - T )(1 - T )
       1       0
H^0_I(M) for all associated primes
{x }
  1
cokernel {1, 1} | x_1^2 0   |
         {1, 2} | 0     x_1 |

        2      3      2 2
T T  + T T  - T T  - T T
 0 1    0 1    0 1    0 1
-------------------------
     (1 - T )(1 - T )
           1       0
{x }
  2
0

{x , x }
  1   2
0
```

Similarly, we denote by M the homology in degree 1 of the bifiltration from Example 3.3.23, and compute:

```
i23 : Dis M
| 0 0 1 1 1 1 |
| 0 0 1 1 1 1 |
| 0 0 1 1 1 1 |
| 0 1 2 2 2 2 |
| 0 1 2 1 1 1 |
| 0 0 0 0 0 0 |

    2     3           3      2 3
```

```
T T  + T T  - T T  - T T  + T T
 0 1    0 1    0 1    0 1    0 1
--------------------------------
        (1 - T )(1 - T )
              1       0
H^0_I(M) for all associated primes
{x }
  1
cokernel {1, 1} | x_1^2 0   0   |
         {1, 2} | 0     x_2 x_1 |

   2    3      2 2      3    2 3
T T  + T T  - T T  - T T  - T T  + T T
 0 1    0 1    0 1    0 1    0 1    0 1
---------------------------------------
            (1 - T )(1 - T )
                  1       0
{x }
  2
cokernel {1, 2} | x_2 x_1 |

   2    2 2      3    2 3
T T  - T T  - T T  + T T
 0 1    0 1    0 1    0 1
------------------------
     (1 - T )(1 - T )
           1       0

{x , x }
  1   2

cokernel {1, 2} | x_2 x_1 |

   2    2 2      3    2 3
T T  - T T  - T T  + T T
 0 1    0 1    0 1    0 1
------------------------
     (1 - T )(1 - T )
           1       0
```

We next consider the module M from Example 3.3.27:

```
o16 : M=S^{{-1,-1}}++S^{{-2,-2}}
o16 : S-module, free, degrees {{1, 1}, {2, 2}}

i17 : M=S^{{-1,-1}}++S^{{-2,-2}};

i17 : Dis(M)
| 0 1 2 2 2 2 |
```

```
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 1 1 1 1 1 |
| 0 0 0 0 0 0 |
            2 2
    T T  + T T
     0 1    0 1
----------------
(1 - T )(1 - T )
       1       0

H^0_I(M) for all associated primes
{x }
  1
0

{x }
  2
0

{x , x }
  1   2
0
```

Since M is a free module, the only submodule supported on the (possible) associated prime is the trivial submodule. Next, we consider module N from Example 3.3.27:

```
i19 : N=S^{{-2,-2}}++S^{{-2,-2}}++((coker matrix{{x_2}})**S^{{-1,-1}})
                                 ++((coker gens ideal(x_1))**S^{{-1,-2}})

o19 = cokernel {2, 2} | 0   0   |
               {2, 2} | 0   0   |
               {1, 1} | x_2 0   |
               {1, 2} | 0   x_1 |

i20 : Dis(N)
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 1 2 2 2 2 |
| 0 1 1 1 1 1 |
| 0 0 0 0 0 0 |
          2 2
   T T  + T T
    0 1    0 1
----------------
(1 - T )(1 - T )
      1       0
```

```
H^0_I(M) for all associated primes
{x }
  1
cokernel {1, 2} | x_1 |

      2    2 2
   T T  - T T
    0 1    0 1
----------------
(1 - T )(1 - T )
      1       0
{x }
  2

cokernel {1, 1} | x_2 |
             2
   T T  - T T
    0 1    0 1
----------------
(1 - T )(1 - T )
      1       0

{x , x }
  1   2
0
```

The modules M and N have the same Hilbert function and multigraded Hilbert series, but the local cohomology distinguishes them.

C.3 Rank

Here we provide a code to compute the rank of the module, using the reduction to a line of Section 3.5.

```
 -- Script to change from a Z^2 graded ring to a singly graded ring.

      RestrictLine2 = (U)->(V = vars ring U;
                            n=rank source V;
                            A = ZZ/31991[a_1..a_n];
                            W = vars A;
                            L = apply(n, i->(V_(0,i)=>W_(0,i)));
                            Uzgraded = A**(matrix MatToList sub(U, L));
                            firstVar = matrix{apply(n, i->a_1)};
                            diagonalIdeal = ideal(firstVar-W);
                            diagonalVar = coker mingens diagonalIdeal;
                            Udiag = coker(Uzgraded)**diagonalVar;
```

```
                        hilbertPolynomial Udiag)


 -- Script to take a multigraded module U, and return the
      --same module over a Z graded ring A with the same number
      --of variables, then restrict to the diagonal and return
      --hilbertPolynomial which is of the form rP_0, r = rk(M).
      --NEED TO HAVE INPUT U A MATRIX (use presentation U if U
      --is a module to get the presentation matrix). Calls the
      --script MatToList to fix grading issues.


      MatToList = (AMAT)->(s = rank source AMAT;
                           t = rank target AMAT;
                           apply(t, i->(apply(s, j->(AMAT_(i,j))))))
```

For instance, for the modules in Example 3.3.27 one computes:

```
i1 : RestrictLine2 presentation N

o1 = 2*P
       0

o1 : ProjectiveHilbertPolynomial

i2 :  RestrictLine2 presentation M

o2 = 2*P
       0
o2 : ProjectiveHilbertPolynomial
```

Thus the modules have both rank 2, as was expected.

Bibliography

[1] Volvis repository. http://volvis.org.

[2] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(1):218–252, 2017.

[3] H. Adams and A. Tausz. JavaPlex tutorial. Available at `https://github.com/appliedtopology/javaplex`.

[4] A. Adcock, E. Carlsson, and G. Carlsson. The ring of algebraic functions on persistence bar codes. *Homology, Homotopy and Applications*, 18(1):381–402, 2016.

[5] R. J. Adler. TOPOS, and why you should care about it. *IMS Bulletin*, 43:4–5, 2014.

[6] R. J. Adler, O. Bobrowski, M. S. Borman, E. Subag, and S. Weinberger. *Persistent homology for random fields and complexes*, volume 6 of *Collections*, pages 124–143. Institute of Mathematical Statistics, Beachwood, Ohio, USA, 2010.

[7] R. J. Adler, O. Bobrowski, and S. Weinberger. Crackle: The homology of noise. *Discrete & Computational Geometry*, 52(4):680–704, 2014.

[8] D. Attali, H. Edelsbrunner, and Y. Mileyko. Weak witnesses for Delaunay triangulations of submanifolds. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, pages 143–150, New York, NY, USA, 2007. ACM.

[9] P. Bajardi, M. Delfino, A. Panisson, G. Petri, and M. Tizzoni. Unveiling patterns of international communities in a global city using mobile phone data. *EPJ Data Science*, 4(1):3, 2015.

[10] J. A. Barcelo and A. Carbery. On the magnitudes of compact sets in Euclidean spaces. *American Journal of Mathematics*, 140:449–494, 2018. doi:10.1353/ajm.2018.0011.

[11] J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete & Computational Geometry*, 47(2):301–328, 2012.

[12] U. Bauer. Ripser, 2016. Available at `https://github.com/Ripser/ripser`.

[13] U. Bauer, M. Kerber, and J. Reininghaus. DIPHA (A distributed persistent homology algorithm). Software available at `https://code.google.com/p/dipha/`.

[14] U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In P.-T. Bremer, I. Hotz, V. Pascucci, and R. Peikert, editors, *Topological Methods in Data Analysis and Visualization III*, Mathematics and Visualization, pages 103–117. Springer International Publishing, 2014.

[15] U. Bauer, M. Kerber, and J. Reininghaus. Distributed computation of persistent homology. In *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, chapter 3, pages 31–38. Society for industrial and applied mathematics, 2014.

[16] U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. PHAT: Persistent homology algorithms toolbox. In Hoon Hong and Chee Yap, editors, *Mathematical Software, ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 137–143. Springer Berlin Heidelberg, 2014. Software available at `https://code.google.com/p/phat/`.

[17] P. Bendich, H. Edelsbrunner, and M. Kerber. Computing robustness and persistence for images. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1251–1260, 2010.

[18] P. Bendich, J. S. Marron, E. Miller, A. Pieloch, and S. Skwerer. Persistent homology analysis of brain artery trees. *The Annals of Applied Statistics*, 10(1):198–218, 2016.

[19] S. Bhattacharya, R. Ghrist, and V. Kumar. Persistent homology for path planning in uncertain environments. *IEEE Transactions on Robotics*, 31(3):578–590, June 2015.

[20] S. Biasotti, A. Cerri, P. Frosini, D. Giorgi, and C. Landi. Multidimensional size functions for shape comparison. *Journal of Mathematical Imaging and Vision*, 32, 2008.

[21] J. Binchi, E. Merelli, M. Rucco, G. Petri, and F. Vaccarino. jHoles: A tool for understanding biological complex networks via clique weight rank persistent homology. *Electronic Notes in Theoretical Computer Science*, 306(0):5–18, 2014. Proceedings of the 5th International Workshop on Interactions between Computer Science and Biology (CS2Bio14).

[22] A. Björner. Topological methods. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of combinatorics*, chapter 34, pages 1819–1872. Elsevier Science B.V., 1995.

[23] O. Bobrowski, S. Mukherjee, and J.E. Taylor. Topological consistency via kernel estimation. *Bernoulli*, 23(1):288–328, 02 2017.

[24] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 208–216, New York, NY, USA, 2009. ACM.

[25] J.-D. Boissonnat, L. J. Guibas, and S. Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete & Computational Geometry*, 42(1):37–70, 2009.

[26] J.-D. Boissonnat and C. Maria. Computing persistent homology with various coefficient fields in a single pass. In A. S. Schulz and D. Wagner, editors, *Algorithms - ESA 2014*, volume 8737 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin Heidelberg, 2014.

[27] N. Bourbaki. *Element of Mathematics. Commutative Algebra, Chapters 1–7*. Springer, 1989.

[28] P. Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(1):77–102, January 2015.

[29] P. Bubenik, V. de Silva, and J. Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15:1501–1531, 2015.

[30] P. Bubenik and P. Dłotko. A persistence landscapes toolbox for topological statistics. *J. Symb. Comput.*, 78(C):91–114, January 2017.

[31] P. Bubenik and P. T. Kim. A statistical approach to persistent homology. *Homology, Homotopy and Applications*, 9(2):337–362, 2007.

[32] P. Bubenik and J. A. Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014.

[33] P. G. Cámara, A. J. Levine, and R. Rabadán. Inference of ancestral recombination graphs through topological data analysis. *PLOS Computational Biology*, 12(8):1–25, 08 2016.

[34] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[35] G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. In *Proceedings of the Twenty-fifth Annual Symposium on Computational Geometry*, SCG '09, pages 247–256, New York, NY, USA, 2009. ACM.

[36] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008.

[37] G. Carlsson, G. Singh, and A. Zomorodian. Computing multidimensional persistence. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors, *Algorithms and Computation: 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, pages 730–739. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[38] G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.

[39] C. J. Carstens and K. J. Horadam. Persistent homology of collaboration networks. *Mathematical Problems in Engineering*, 2013.

[40] W. Chacholski, M. Scolamiero, and F. Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221:1055–1075, 2017.

[41] J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 110(46):18566–18571, 2013.

[42] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Y. Oudot. Proximity of persistence modules and their diagrams. pages 237–246. ACM, New York, 2009.

[43] F. Chazal, B. T. Fasy, F. Lecci, B. Michel, A. Rinaldo, and L. Wasserman. Robust Topological Inference: Distance To a Measure and Kernel Distance. *Journal of Machine Learning Research*, 18:1–40, 2017.

[44] C. Chen and M. Kerber. Persistent homology computation with a twist. In *Proceedings of the 27th European Workshop on Computational Geometry*, pages 197–200, 2011.

[45] M. K. Chung, P. Bubenik, and P. T. Kim. Persistence diagrams of cortical surface data. In J. L. Prince, D. L. Pham, and K. J. Myers, editors, *Information Processing in Medical Imaging*, volume 5636 of *Lecture Notes in Computer Science*, pages 386–397. Springer Berlin Heidelberg, 2009.

[46] J. Cochoy and S. Oudot. Decomposition of exact pfd persistence bimodules. *ArXiv e-prints*, May 2016. Preprint available at `https://arxiv.org/abs/1605.09726`.

[47] M. M. Cohen. *A Course in Simple Homotopy Theory*. Graduate Texts in Mathematics. Springer-Verlag New York Inc., 1970.

[48] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[49] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[50] R. Corbet and M. Kerber. The Representation Theorem of Persistent Homology Revisited and Generalized. *ArXiv e-prints*, July 2017. Preprint available at `https://arxiv.org/abs/1707.08864`.

[51] O. T. Courtney and G. Bianconi. Generalized network structures: The configuration model and the canonical ensemble of simplicial complexes. *Phys. Rev. E*, 93:062311, Jun 2016.

[52] L. Crawford, A. Monod, A. X. Chen, S. Mukherjee, and R. Rabadán. Topological Summaries of Tumor Images Improve Prediction of Disease Free Survival in Glioblastoma Multiforme. *ArXiv e-prints*, November 2016. 1611.06818.

[53] J. Curry. Sheaves, cosheaves and applications. *ArXiv e-prints*, March 2013. 1303.3255.

[54] E. B. Curtis. Simplicial homotopy theory. *Advances in Mathematics*, 6:107–209.

[55] C. Curto. What can topology tell us about the neural code? *Bulletin (New Series) of the American Mathematical Society*, 54(1):63–78, 2017.

[56] Y. Dabaghian, V. L Brandt, and L. M. Frank. Reconceiving the hippocampal map as a topological template. *eLife*, 3:e03476, 2014.

[57] T. A. Davis and Y. Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software*, 38:1–25, 2011. Available at `http://www.cise.ufl.edu/research/sparse/matrices`.

[58] V. de Silva. A weak characterisation of the Delaunay triangulation. *Geometriae Dedicata*, 135(1):39–64, 2008.

[59] V. de Silva and G. Carlsson. Topological estimation using witness complexes. In *Proceedings of the First Eurographics conference on Point-Based Graphics*, pages 157–166. Eurographics Association, 2004.

[60] V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7:339–358, 2007.

[61] V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Dualities in persistent (co)homology. *Inverse Problems*, 27(12):124003, 2011.

[62] V. de Silva, D. Morozov, and M. Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, 45(4):737–759, 2011.

[63] A. M. del Campo Sanchéz and N. Otter. Efficient computation of multiparameter persistent homology. 2018. In preparation.

[64] The Sage Developers. *Sage Mathematics Software*. Available at `http://www.sagemath.org`.

[65] D. DeWoskin, J. Climent, I. Cruz-White, M. Vazquez, C. Park, and J. Arsuaga. Applications of computational homology to the analysis of treatment response in breast cancer patients. *Topology and its Applications*, 157(1):157–164, 2010. Proceedings of the International Conference on Topology and its Applications 2007 at Kyoto; Jointly with 4th Japan Mexico Topology Conference.

[66] T. K. Dey, F. Fan, and Y. Wang. Graph induced complex on point data. In *Proceedings of the Twenty-ninth Annual Symposium on Computational Geometry*, SoCG '13, pages 107–116, New York, NY, USA, 2013. ACM.

[67] T. K. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 345–354, New York, NY, USA, 2014. ACM.

[68] T. K. Dey, D. Shi, and Y. Wang. SimBa: An efficient tool for approximating Rips-filtration persistence via simplicial batch-collapse. In *LIPIcs-Leibniz International Proceedings in Informatics. Vol. 57*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[69] P. Dłotko. Persistence landscape toolbox. Available at `https://www.math.upenn.edu/~dlotko/persistenceLandscape.html`.

[70] P. Dłotko, K. Hess, R. Levi, M. Nolte, M. Reimann, M. Scolamiero, K. Turner, E. Muller, and H. Markram. Topological analysis of the connectome of digital reconstructions of neural microcircuits. *ArXiv e-prints*, January 2016. Preprint available at `https://arxiv.org/abs/1601.01580`.

[71] H. Edelsbrunner. The union of balls and its dual shape. *Discrete & Computational Geometry*, 13(3-4):415–440, 1995.

[72] H. Edelsbrunner and J. Harer. Persistent homology — A survey. In J. E. Goodman, J. Pach, and R. Pollack, editors, *Surveys on Discrete and Computational Geometry. Twenty Years Later*, volume 453 of *Contemporary Mathematics*, pages 257–282. 2008.

[73] H. Edelsbrunner and J. Harer. Persistent homology — a survey. *Contemporary Mathematics*, 453:257–282, 2008.

[74] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Applied mathematics. American Mathematical Society, 2010.

[75] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, Jul 1983.

[76] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.

[77] H. Edelsbrunner and D. Morozov. Persistent homology: Theory and practice. In *Proceedings of the European Congress of Mathematics*, pages 31–50, 2012.

[78] H. Edelsbrunner, D. Morozov, and V. Pascucci. Persistence-sensitive simplification functions on 2-manifolds. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 127–134, New York, NY, USA, 2006. ACM.

[79] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994.

[80] S. Eilenberg and N. E. Steenrod. *Foundations of Algebraic Topology*. Princeton Mathematical Series. Princeton University Press, 1952.

[81] D. Eisenbud. *Commutative algebra with a view toward algebraic geometry*. Graduate texts in mathematics. Springer-Verlag, 1995.

[82] D. Eisenbud. *The Geometry of Syzygies: A Second Course in Algebraic Geometry and Commutative Algebra*. Graduate Texts in Mathematics. Springer-Verlag Berlin Heidelberg New York, 2005.

[83] K. Emmett, D. Rosenbloom, P. Camara, and R. Rabadan. Parametric inference using persistence diagrams: A case study in population genetics. *ArXiv e-prints*, June 2014. 1406.4582.

[84] K. Emmett, B. Schweinhart, and R. Rabadan. Multiscale topology of chromatin folding. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS)*, BICT'15, pages 177–180, ICST, Brussels, Belgium, Belgium, 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[85] L. Euler. *Opera Omnia IV.A-4. The Correspondence Euler – Goldbach.* 2011.

[86] B. T. Fasy, J. Kim, F. Lecci, and C. Maria. Introduction to the R package TDA. *ArXiv e-prints*, November 2014. 1411.1830.

[87] B. T. Fasy, J. Kim, F. Lecci, C. Maria, and V. Rouvreau. TDA: Statistical tools for topological data analysis. Available at `https://cran.r-project.org/web/packages/TDA/index.html`.

[88] B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, and A. Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301–2339, 12 2014.

[89] M. Gameiro, Y. Hiraoka, S. Izumi, M. Kramár, K. Mischaikow, and V. Nanda. A topological measurement of protein compressibility. *Japan Journal of Industrial and Applied Mathematics*, 32(1):1–17, 2015.

[90] G. Gan, C. Ma, and J. Wu. *Data Clustering: Theory, Algorithms, and Applications*. SIAM, Philadelphia, PA, 2007.

[91] I. M. Gelfand. Mathematics as an adequate language. volume 244 of *Progress in Mathematics*. Birchhäuser, Boston, Basel, Berlin, 2005.

[92] R. Ghrist. Barcodes: The persistent topology of data. *Bulletin of the American Mathematical Society*, 45:61–75, 2008.

[93] R. Ghrist. *Elementary Applied Topology*. Createspace, 2014. ed. 1.0.

[94] M. Gidea. Topology data analysis of critical transitions in financial networks. *ArXiv e-prints*, 2017. Preprint available at `https://arxiv.org/abs/1701.06081`.

[95] H. Gimperlein and M. Goffeng. On the magnitude function of domains in Euclidean space. *ArXiv e-prints*, June 2017. Preprint available at `https://arxiv.org/abs/1706.06839`.

[96] C. Giusti, R. Ghrist, and D.S. Bassett. Two's company and three (or more) is a simplex. *Journal of Computational Neuroscience*, 41:1–14, 2016.

[97] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

[98] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., Boca Raton, FL, USA, 2 edition, 2004.

[99] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at `http://www.math.uiuc.edu/Macaulay2/`.

[100] L. J. Guibas and S. Y. Oudot. Reconstruction using witness complexes. *Discrete & Computational Geometry*, 40(3):325–356, 2008.

[101] M. Guillemard, H. Boche, G. Kutyniok, and F. Philipp. Signal analysis with frame theory and persistent homology. In *10th International Conference on Sampling Theory and Applications*, pages 309–312, 2013.

[102] H. Harrington, N. Otter, H. Schenck, and U. Tillmann. Stratifying multiparameter persistent homology. *ArXiv e-prints*, 8 2017. Preprint available at `https://arxiv.org/abs/1708.07390`.

[103] A. Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, New York, 2002.

[104] J.-C. Hausmann. On the Vietoris–Rips complexes and a cohomology theory for metric spaces. pages 175–188. Princeton U. Press, Princeton, 1995.

[105] R. Hepworth and S. Willerton. Categorifying the magnitude of a graph. *ArXiv e-prints*, May 2015. Preprint available at `https://arxiv.org/abs/1505.04125`.

[106] Y. Hiraoka, T. Nakamura, A. Hirata, E.G. Escolar, K. Matsue, and Y. Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016.

[107] D. Horak, S. Maletić, and M. Rajković. Persistent homology of complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):P03034, 2009.

[108] J. Jonsson. *Simplicial Complexes of Graphs*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 2007.

[109] M. Joswig and M. E. Pfetsch. Computing optimal Morse matchings. *SIAM Journal on Discrete Mathematics*, 20(1):11–25, 2006.

[110] B. Jubin. On the magnitude homology of metric spaces. *ArXiv e-prints*, March 2018. Preprint available at `https://arxiv.org/abs/1803.05062`.

[111] Department of Computer Science Jyamiti research group (Prof. Tamal K. Dey) and Ohio State University Engineering. GIComplex, 2013. Available at `http://web.cse.ohio-state.edu/~tamaldey/GIC/GICsoftware/`.

[112] Department of Computer Science Jyamiti research group (Prof. Tamal K. Dey) and Ohio State University Engineering. SimpPers, 2014. Available at `http://web.cse.ohio-state.edu/~tamaldey/SimpPers/SimpPers-software/`.

[113] T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational Homology*, volume 157 of *Applied Mathematical Sciences*. Springer, 2004.

[114] M. Kahle. Random geometric complexes. *Discrete & Computational Geometry*, 45(3):553–573, 2011.

[115] M. Kahle. Topology of random simplicial complexes: A survey. In *AMS Contemporary Mathematics – Applied algebraic topology: New directions and applications*, volume 620, pages 201–222. 2014.

[116] L. Kanari, P. Dłotko, M. Scolamiero, R. Levi, J. Shillcock, K. Hess, and H. Markram. Quantifying topological invariants of neuronal morphologies. *ArXiv e-prints*, March 2016. 1603.08432.

[117] R. Kaneta and M. Yoshinaga. Magnitude homology of metric spaces and order complexes. *ArXiv e-prints*, February 2018. Preprint available at `https://arxiv.org/abs/1803.04247`.

[118] M. Kashiwara and P. Schapira. Persistent homology and microlocal sheaf theory. *ArXiv e-prints*, May 2017. Preprint available at `https://arxiv.org/abs/1705.00955`.

[119] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, Inc., New York, NY, 1990.

[120] G. M. Kelly. *Basic Concepts of Enriched Category Theory*, volume 64 of *Lecture Notes in Mathematics 64*. Cambridge University Press, 1982.

[121] M. Kerber, D. Morozov, and A. Nigmetov. Hera, 2016. Available at `https://bitbucket.org/grey_narn/hera`.

[122] M. Kerber, D. Morozov, and A. Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics*, 22:1.4:1–1.4:20, September 2017.

[123] M. Kerber and R. Sharathkumar. Approximate Čech complex in low and high dimensions. In L. Cai, S.-W. Cheng, and T.-W. Lam, editors, *24th International Symposium on Algorithms and Computation (ISAAC 2013)*, Lecture Notes in Computer Science 8283, pages 666–676, 2013.

[124] K. P. Knudson. A refinement of multi-dimensional persistence. *Homology, Homotopy and Applications*, 10:259–281, 2008.

[125] V. Kovacev-Nikolic, P. Bubenik, D. Nikolić, and G. Heo. Using persistent homology and dynamical distances to analyze protein binding. *Statistical Applications in Genetics and Molecular Biology*, 15(1):19–38, 2016.

[126] M. Kramár, A. Goullet, L. Kondic, and K. Mischaikow. Persistence of force networks in compressed granular media. *Physical Review E*, 87:042207, Apr 2013.

[127] M. Kramár, A. Goullet, L. Kondic, and K. Mischaikow. Quantifying force networks in particulate systems. *Physica D*, 283:37–55, 2014.

[128] V. Kurlin, 2015. `http://kurlin.org/projects/persistent-skeletons.cpp`.

[129] V. Kurlin. A one-dimensional homologically persistent skeleton of an unstructured point cloud in any metric space. *Computer Graphics Forum*, 34(5):253–262, 2015.

[130] R. Kwitt, S. Huber, M. Niethammer, W. Lin, and U. Bauer. Statistical topological data analysis - a kernel perspective. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3052–3060. Curran Associates, Inc., 2015.

[131] Los Alamos National Laboratory. HIV Database. Available at `http://www.hiv.lanl.gov/content/index`.

[132] Stanford University Computer Graphics Laboratory. The stanford 3D scanning repository. Available at `https://graphics.stanford.edu/data/3Dscanrep`.

[133] F.W. Lawvere. *Rendiconti del Seminario Matematico e Fisico di Milano*, XLIII:135–166, 1973.

[134] Y. Lee, S. D. Barthel, P. Dłotko, S. Mohamad Moosavi, K. Hess, and B. Smit. Quantifying similarity of pore-geometry in nanoporous materials. *Nature Communications*, 8:15396, 2017.

[135] G. Leibon, S. Pauls, D. Rockmore, and R. Savell. Topological structures in the equities market network. *Proceedings of the National Academy of Sciences of the United States of America*, 105(52):20589–20594, 2008.

[136] T. Leinster. The Euler characteristic of a category. *Documenta Mathematica*, 13:21–49, 2008.

[137] T. Leinster. The magnitude of metric spaces. *Documenta Mathematica*, 18:857–905, 2013.

[138] T. Leinster and M. Shulman. Online discussion at the n-Category Café. `https://golem.ph.utexas.edu/category/2016/09/magnitude_homology.html`.

[139] T. Leinster and M. Shulman. Magnitude homology of enriched categories and metric spaces. *ArXiv e-prints*, 11 2017. Preprint available at `https://arxiv.org/abs/1711.00802`.

[140] M. Lesnick and M. Wright. Interactive visualization of 2-D persistence modules. *ArXiv e-prints*, December 2015.

[141] M. Lesnick and M. Wright. RIVET: The rank invariant visualization and exploration tool, 2016. Software available at `http://rivet.online/`.

[142] D. Lo and B. Park. Modeling the spread of the Zika virus using topological data analysis. *PLOS ONE*, 13(2):1–7, 02 2018.

[143] L.-D. Lord, P. Expert, H. M Fernandes, G. Petri, T. J Van Hartevelt, F. Vaccarino, G. Deco, F. Turkheimer, and M.L. Kringelbach. Insights into brain architectures from the homological scaffolds of functional connectivity networks. *Frontiers in Systems Neuroscience*, 10, 2016.

[144] F. Loregian. This is the (co)end, my only (co)friend. *ArXiv e-prints*, January 2015. Preprint available at `https://arxiv.org/abs/1501.02503`.

[145] S. Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1998.

[146] R. MacPherson and B. Schweinhart. Measuring shape with topology. *Journal of Mathematical Physics*, 53(7):073516, 2012.

[147] S. Maletic, Y. Zhao, and M. Rajkovic. Persistent topological features of dynamical systems. *Chaos*, 26(053105), 2016.

[148] C. Maria. Algorithms and data structures in computational topology. 2014. PhD Thesis, Université de Nice-Sophia Antipolis. Available at `http://www-sop.inria.fr/members/Clement.Maria/docs/ClementMaria_PhDdissertation.pdf`.

[149] C. Maria, J.-D. Boissonnat, M. Glisse, and M. Yvinec. The Gudhi library: Simplicial complexes and persistent homology. In H. Hong and C. Yap, editors, *Mathematical Software, ICMS 2014*, volume 8592 of *Lecture Notes in Computer Science*, pages 167–174. Springer Berlin Heidelberg, 2014. Software available at `https://project.inria.fr/gudhi/software/`.

[150] J. P. May. The additivity of traces in triangulated categories. *Advances in Mathematics*, 163(1):34–73, 2001.

[151] M. W. Meckes. Positive definite metric spaces. *Positivity*, 17(3):733–757, Sep 2013.

[152] M. W. Meckes. Magnitude, diversity, capacities, and dimensions of metric spaces. *Potential Analysis*, 42(2):549–572, Feb 2015.

[153] Y. Mileyko, S. Mukherjee, and J. Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, 2011.

[154] E. Miller. Data structures for real multiparameter persistence modules. *ArXiv e-prints*, September 2017. Preprint available at `https://arxiv.org/abs/1709.08155`.

[155] E. Miller and B. Sturmfels. *Combinatorial Commutative Algebra*. Graduate Texts in Mathematics. Springer-Verlag Berlin Heidelberg New York, 2005.

[156] N. Milosavljević, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Proceedings of the Twenty-seventh Annual Symposium on Computational Geometry*, SoCG '11, pages 216–225, New York, NY, USA, 2011. ACM.

[157] K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete & Computational Geometry*, 50(2):330–353, 2013.

[158] D. Morozov. Dionysus. Software available at `http://www.mrzv.org/software/dionysus/`.

[159] D. Morozov. Persistence algorithm takes cubic time in worst case. *BioGeometry News, Dept. Comput. Sci., Duke Univ*, 2005.

[160] E. Munch, K. Turner, P. Bendich, S. Mukherjee, J. Mattingly, and J. Harer. Probabilistic Fréchet means for time varying persistence diagrams. *Electronic Journal of Statistics*, 9(1):1173–1204, 2015.

[161] V. Nanda. Perseus, the persistent homology software. Software available at `http://www.sas.upenn.edu/~vnanda/perseus`.

[162] V. Nanda. Discrete morse theory for filtrations. 2012. PhD Thesis. Rutgers, The State University of New Jersey.

[163] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, Sep 2006.

[164] M. Nicolau, A. J. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences of the United States of America*, 108(17):7265–7270, 2011.

[165] N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6(17), 2017.

[166] S. Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*, volume 209 of *AMS Mathematical Surveys and Monographs*. American Mathematical Society, 2015.

[167] S. Pal, T. J. Moore, R. Ramanathan, and A. Swami. *Comparative Topological Signatures of Growing Collaboration Networks*, pages 201–209. Springer International Publishing, Cham, 2017.

[168] M. Penrose. *Random Geometric Graphs*. Oxford University Press, Oxford, UK, 2003.

[169] J. A. Perea, A. Deckard, S. B. Haase, and J. Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16(1):1–12, 2015.

[170] P. Perry and V. de Silva. Plex, 2000–2006. Available at `http://mii.stanford.edu/research/comptop/programs/`.

[171] G. Petri, M. Scolamiero, I. Donato, and F. Vaccarino. Topological strata of weighted complex networks. *PLoS ONE*, 8(6):1–8, 2013.

[172] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy. Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, 35(1–3):204–223, January 2016.

[173] K. T. Poole. Voteview. `http://voteview.com`, 2016.

[174] R. E. Reed. Foundations of Vietoris homology with applications to non-compact spaces, 1980. PhD thesis, Polska Akademia Nauk, Instytut Matematyczny.

[175] J. Reininghaus, S. Huber, U. Bauer, and R. Kwitt. A stable multi-scale kernel for topological machine learning. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4741–4748, June 2015.

[176] G. A. Reisner. Cohen–macaulay quotients of polynomial rings. *Advances in Mathematics*, 21:30–49, 1975.

[177] H. Reitberger. Leopold vietoris (1891–2002). *Notices of the AMS*, pages 1232–1236, 2002.

[178] A.H. Rizvi, P.G. Camara, E.K. Kandror, T.J. Roberts, I. Schieren, T. Maniatis, and R. Rabadan. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, Advance Online Publication (`doi:10.1038/nbt.3854`), 2017.

[179] F. Robin. Morse theory for cell complexes. *Advances in Mathematics*, 134(1):90–145, 1998.

[180] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1:27–64, 2007.

[181] S. H. Schanuel. Negative sets have Euler characteristic and dimension. Lecture notes in mathematics 1488, pages 379–385. Springer, Berlin, 1990.

[182] H. Schenck. *Computational Algebraic Geometry*. London Mathematical Society Student Texts. Cambridge University Press, 2003.

[183] M. Scolamiero, W. Chachólski, A. Lundman, R. Ramanujam, and S. Öberg. Multidimensional persistence and noise. *Foundations of Computational Mathematics*, 17:1367–1406, 2017.

[184] D. R. Sheehy. Linear-size approximations to the Vietoris-Rips filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013.

[185] M. Shulman. Online comment at the n-Category Café. `https://golem.ph.utexas.edu/category/2016/08/a_survey_of_magnitude.html#c051059`.

[186] M. Shulman. Homotopy limits and colimits and enriched homotopy theory. *ArXiv Mathematics e-prints*, October 2006. Preprint available at `https://arxiv.org/abs/math/0610194`.

[187] G. Singh, F. Mémoli, and G. Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, pages 91–100, 2007.

[188] G. Singh, F. Mémoli, T. Ishkhanov, G. Sapiro, G. Carlsson, and D. Ringach. Topological analysis of population activity in visual cortex. *Journal of Vision*, 8:1–18, 2008.

[189] N. Singh, H. D. Couture, J. S. Marron, C. Perou, and M. Niethammer. *Topological Descriptors of Histology Images*, pages 231–239. Springer International Publishing, Cham, 2014.

[190] A. Sizemore, C. Giusti, and D.S. Bassett. Classification of weighted networks through mesoscale homological features. *Journal of Complex Networks*, 5(2):245, 2017.

[191] O. Sporns. Small-world connectivity, motif composition, and complexity of fractal neuronal connections. *Biosystems*, 85(1):55–64, July 2006.

[192] B. J. Stolz, H. A. Harrington, and M. A. Porter. Persistent homology of time-dependent functional networks constructed from coupled time series. *Chaos*, 27:047410, 2017.

[193] A. Tausz, M. Vejdemo-Johansson, and H. Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at `http://appliedtopology.github.io/javaplex/`.

[194] D. Taylor, F. Klimm, H. A. Harrington, M. Kramár, K. Mischaikow, M. A. Porter, and P. J. Mucha. Topological data analysis of contagion maps for examining spreading processes on networks. *Nature Communications*, 6, 2015. Article number 7723.

[195] The CGAL Project. *CGAL user and reference manual*. CGAL Editorial Board, 4.7 edition, 2015.

[196] C. M. Topaz, L. Ziegelmeier, and T. Halverson. Topological data analysis of biological aggregation models. *PLoS ONE*, 10(5):1–26, 05 2015. Article number e0126383.

[197] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer. Fréchet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52(1):44–70, 2014.

[198] K. Turner, S. Mukherjee, and D. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference*, 3:310–344, 2014.

[199] R. Vasudevan, A. Ames, and R. Bajcsy. Persistent homology for automatic determination of human-data based cost of bipedal walking. *Nonlinear Analysis: Hybrid Systems*, 7(1):101–115, 2013. IFAC World Congress 2011.

[200] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75:1226–1229, Aug 1995.

[201] L. Vietoris. Über den höheren Zusammenhang kompakter Räume und eine Klasse von zusammenhangstreuen Abbildungen. *Mathematische Annalen*, 97:454–472, 1927.

[202] H. Wagner, C. Chen, and E. Vuçini. Efficient computation of persistent homology for cubical data. In Ronald Peikert, Helwig Hauser, Hamish Carr, and Raphael Fuchs, editors, *Topological Methods in Data Analysis and Visualization II*, Mathematics and Visualization, pages 91–106. Springer Berlin Heidelberg, 2012.

[203] B. Wang and G.-W. Wei. Object-oriented persistent homology. *Journal of Computational Physics*, 305:276–299, 2016.

[204] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):440–442, 1998.

[205] A. S Waugh, L. Pei, J. H Fowler, P. J Mucha, and M. A Porter. Party polarization in congress: A network science approach. *arXiv:0907.3509*, 2012. Data available at `http://figshare.com/articles/Roll_Call_Votes_United_States_House_and_Senate/1590036`.

[206] C. Webb. Decomposition of graded modules. *Proceedings of the American Mathematical Society*, 94(4), 1985.

[207] S. Weinberger. What is... Persistent Homology? *Notices of the AMS*, 58:36–39, January 2011.

[208] J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 314(1165):1–340, 1986.

[209] A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11–15, May 2014.

[210] A. C. Wilkerson, H. Chintakunta, H. Krim, T. J. Moore, and A. Swami. A distributed collapse of a network's dimensionality. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 595–598, Dec 2013.

[211] A. C. Wilkerson, T. J. Moore, A. Swami, and H. Krim. Simplifying the homology of networks via strong collapses. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5258–5262, May 2013.

[212] S. Willerton. The magnitude of odd balls via Hankel determinants of reverse Bessel polynomials. *ArXiv e-prints*, August 2017. Preprint available at `https://arxiv.org/abs/1708.03227`.

[213] K. Xia, X. Feng, Y. Tong, and G. W. Wei. Persistent homology for the quantitative prediction of fullerene stability. *Journal of Computational Chemistry*, 36(6):408–422, 2015.

[214] K. Xia, Z. Li, and L. Mu. Multiscale persistent functions for biomolecular structure characterization. *Bulletin of Mathematical Biology*, 80(1):1–31, Jan 2018.

[215] K. Xia and G.-W. Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International Journal for Numerical Methods in Biomedical Engineering*, 30:814–844, 2014.

[216] K. Xia and G.-W. Wei. A review of geometric, topological and graph theory apparatuses for the modeling and analysis of biomolecular data. *ArXiv e-prints*, 2016. Preprint available at `https://arxiv.org/abs/1612.01735`.

[217] J. Yoo, E. J. Kim, Y. M. Ahn, and J. C. Ye. Topological persistence vineyard for dynamic functional brain connectivity during resting and gaming stages. *Journal of Neuroscience Methods*, 267:1–13, 2016.

[218] W. Zhou and H. Yan. Alpha shape and Delaunay triangulation in studies of protein-related interactions. *Briefings in Bioinformatics*, 15(1):54–64, 2014.

[219] X. Zhu, A. Vartanian, M. Bansal, D. Nguyen, and L. Brandl. Stochastic multiresolution persistent homology kernel. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2449–2455. AAAI Press, 2016.

[220] A. Zomorodian. *Topology for Computing*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009.

[221] A. Zomorodian. Technical section: Fast construction of the Vietoris-Rips complex. *Computers and Graphics*, 34(3):263–271, June 2010.

[222] A. Zomorodian. The tidy set: A minimal simplicial set for computing homology of clique complexes. pages 257–266, 2010. SCG10.

[223] A. Zomorodian. Topological data analysis. In Afra Zomorodian, editor, *Advances in Applied and Computational Topology: American Mathematical Society Short Course on Computational Topology, January 4-5, 2011, New Orleans, Louisiana*, volume 70 of *Proceedings of symposia in applied mathematics*, pages 1–39. American Mathematical Society, 2012.

[224] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2):249–274, February 2005.