Computational Topology in Neuroscience

Bernadette Stolz
Lincoln College
University of Oxford

A dissertation submitted for the degree of
Master of Science in Mathematical Modelling & Scientific Computing

September 2014

To my grandfather, Jaromír Hladký, who was my first critical reader.

Acknowledgements

I would like to thank my supervisors Dr. Heather Harrington and Prof. Mason Porter for providing me with such an interesting topic and being both very supportive and helpful throughout my dissertation. I enjoyed working with them very much and am grateful for all their advice and patience, in particular during the weeks where one IT problem seemed to chase the next.

Further thanks go to Dr. Danielle Bassett from the University of Pennsylvania for letting me use her experimental data [5] and MATLAB codes [24, 30], which I used when editing my own codes, and for providing me with additional information on the data set.

Moreover, I would also like to thank Jacopo Binchi and in particular Matteo Rucco from the University of Camerino for all their help and support during my work with the JHOLES algorithm [11].

I would also like to thank Corrine Previte from the Colorado State University, who answered all my questions about D-neighbourhood complexes and her MATLAB implementations of these patiently, and was moreover happy to discuss ideas of adapting these codes to use on our data sets [36].

On a more general note, I would like to thank Dr. Kathryn Gillow, the course director of the M.Sc., for her great organisation and support throughout the course.

Finally, I would also like to thank the Berrow Foundation, in particular the Marquise de Amodio, for funding me throughout my Masters.

Abstract

Computational topology is a set of algorithmic methods developed to understand topological invariants such as loops and holes in high-dimensional data sets. In particular, a method know as *persistent homology* has been used to understand such shapes and their persistence in point clouds and networks. It has only been applied to neuronal networks in recent years. While most tools from network science focus solely on local properties based on pairwise connections, the topological tools reveal more global features. We apply persistent homology to neuronal networks to see which properties these tools can uncover, which might be invisible to existing methods.

We give an introduction to relevant concepts from algebraic topology such as topological spaces, simplicial complexes and filtrations. Filtrations are the main ingredients for methods from persistent homology and can be imagined as an embedded sequence of networks with some form of geometrical object built from the edges and nodes in each sequence step. We use three different filtrations: a filtration by weights, a weight rank clique filtration and a modified version of the Vietoris-Rips complex to analyse networks. Our example networks consist of data from neuroscientific experiments and the output of a non-linear oscillator model, the Kuramoto model.

Our results reveal that all three methods can be used to investigate different aspects of such networks, but that the methods and their interpretation still need to be developed further. In particular, computational scaling needs to be improved on the more sophisticated methods so that they can be used on networks of a reasonably large size and density.

Contents

List of Figures

Chapter 1

Introduction

In the process of understanding the human brain, the study of neuronal networks using mathematical tools has become increasingly important [13, 14, 39]. The human brain consists of approximately 100 billion neurons, whose major task is to receive, conduct, and transmit signals. Every neuron consists of a cell body and one long axon, which is responsible for propagating signals to other cells [2]. This interaction can be modelled using networks, which one can subsequently analyse using graph theory. A relatively new approach to analyse such networks is based on *computational topology*, a set of algorithmic methods used to understand topological invariants such as connectedness, loops, or holes in high-dimensional data structures [18, 19, 20]. The advantage of such analysis is that it goes beyond pairwise connections and enables one to understand global low-dimensional structures in networks, which is difficult for existing methods. In particular, *persistent homology*, a method that consists of a mathematical formalism to explore the persistence of such structures in data sets, has only in recent years been used on neuronal networks and has already lead to promising results (see, for example, [17, 34]).

The aim of this dissertation is to give both an introduction to persistent homology methods by illustrating their use on small examples and to apply them to networks generated from time-series data. There will be two sources for the time-series used: A mathematical model, the Kuramoto model, and functional magnetic resonance imaging (fMRI) data gained from neuroscientific experiments conducted on human subjects. One creates a functional network from the time-series data regarding every entity monitored over time as a separate node connected to every other entity observed with an edge weight based on the pairwise similarity of the two time-series. One can then analyse the functional networks constructed in this way by tools from computational topology. In Figure 1.1 we give a schematic representation of the approach taken in this dissertation.

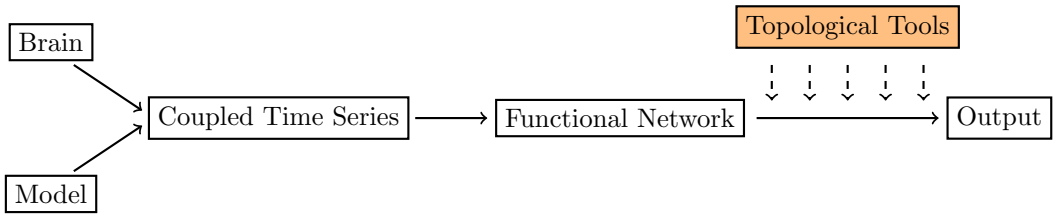

Figure 1.1: Illustration of the approach taken in this dissertation. time-series data is generated from both neuroscientific experiments and a mathematical model. We generate a functional network based on some measure of pairwise similarity between the time-series, which we then analyse using tools from computational topology.

The networks generated from the experimental data [5] evolve in time in the sense that the time-series were monitored on three different days during a learning process. For every subject we will thus be looking at the generated network at three points in time. We study the data from the mathematical model both in the form of one and several networks generated from the time-series based on different time intervals within the series. We use the mathematical model output in addition to the experimental data to form an intuition for the interpretation of the results we obtain by applying the topological methods. Because the Kuramoto model is well-studied (see, for example, [23, 40]), this gives us a way to control some features of the functional networks created and determine what we can learn from employing topological methods. We then use these findings to interpret the functional networks from experimental data. The data set is also well-studied by existing methods from networks science [5, 6], which will also be useful for the interpretation and discussion of our results. In our data analysis we apply several different methods from persistent homology and compare these against each other, which is a novel contribution of this dissertation. Moreover, we use the newly developed algorithm JHOLES [11] in one part of the analysis.

The incorporation of spatial information into the analysis of brain networks is still a major open issue (see [43] for an example of recent research). While the consideration of this is important for neuronal networks, we do not explicitly consider this information in this dissertation. Methods from computational topology could, however, potentially be used when focussing on such questions.

The present dissertation is structured as follows: In Chapter 2, we give an introduction to the topological background needed to understand the methods, that we apply and we also give an introduction to the methods themselves. We illustrate the mathematical concepts on a variety of examples. If not stated otherwise, we construct these examples specifically for this purpose. We give a brief introduction to

the Kuramoto model, neuronal networks in general, and the experimental data set that we use in Chapter 3. In Chapter 4, we give an overview on the employed computational tools and the results that we obtain. We discuss the results in Chapter 5 and summarise the main conclusions in Chapter 6. In Appendix A, we list further definitions from algebra and topology, that can be helpful when following the theoretical introduction in Chapter 2. We list further results in Appendix B.

Chapter 2

Computational Topology

Computational topology consists of algorithmic methods which investigate high-dimensional data in a quantitative manner [18, 19, 20]. One might think of it as a tool for understanding shapes and surfaces in data structures. An important method within computational topology is *persistent homology*, which is a mathematical formalism for analysing topological invariants such as connectedness, loops, or holes in various dimensions. Persistent homology enables one to detect structures persisting in a data set in a certain sense. The applications are particularly promising in the context of network science because persistent homology methods can not only uncover non-local structures in large data sets, which is difficult for existing methods, but also allow the simultaneous analysis of connections of more than just one node, which is also an important advantage over standard methods in network science. We now introduce a set of basic notions from topology before we proceed to explain a few methods from persistent homology, which we use in Chapter 4 to analyse a set of network examples. If not referenced otherwise, the discussion in this Chapter based on [17, 20]. Further algebraic and topological prerequisites for the notions introduced in the main text are listed in Appendix A.

2.1 Topological background

2.1.1 Graphs from a topological perspective

From a topological point of view, a graph is a one-dimensional object that consists of points (the vertices), and curves connecting these points (the edges). The topological definition differs slightly from the commonly used definition in network science, where the vertices describe abstract elements and the edges consist of pairs of these elements

[32]. However, the two definitions are compatible. We now introduce a few topological concepts for graphs.

Definition 2.1.1 (graph)**.** A *graph* $G = (V, E)$ is a pair that consists of a set of vertices V and edges E.

A subgraph $H = (\tilde{V}, \tilde{E})$ of G is defined by the subsets $\tilde{V} \subseteq V$ and $\tilde{E} \subseteq E$. We use the following definition to classify an in practice very commonly used set of graphs with an important property.

Definition 2.1.2 (simple graph)**.** A graph $G = (V, E)$ is called *simple* if the set of edges E is a subset of the set of unordered pairs of vertices.

This implies that there are no multiple edges between two vertices and no node is connected to itself.

Definition 2.1.3 (complete graph)**.** A graph $G = (V, E)$ is called *complete*, if the set E of edges contains exactly one edge for every unordered pair of vertices.

Remark 1.
i. Every simple graph with $n \in \mathbb{N}$ vertices is a subgraph of the complete graph K_n on n vertices.
ii. Complete graphs on $n + 1$ vertices are sometimes also called $n-$*cliques*.

Given two vertices of a simple graph, we might want to know if it is possible to go from one vertex to the other within the graph. This is the case if there is a path between the vertices.

Definition 2.1.4 (path)**.** Let $G = (V, E)$ be a simple graph with $n \in \mathbb{N}$ vertices. Let $u, v \in V$ be a pair of vertices. We say that there is a *path of length* k between vertex u and vertex v if there exists a sequence of vertices $\{u_0, u_1, \ldots, u_k\} \subseteq V$, $k \leq n$, with $u_0 = u$ and $u_k = v$ such that there is an edge $e_{i+1} \in E$ between u_i and u_{i+1} for every $i \in \{0, \ldots, k-1\}$. We call the path *simple*, if the vertices in the path are distinct.

The following few definitions yield one of the most important topological concepts that we will study using persistent homology.

Definition 2.1.5 (connected graph)**.** A simple graph is *connected* if there exists a path between every pair of vertices $v_1, v_2 \in V$.

If a graph is not connected, then we will look at the amount of "non-connectedness" by considering connected subgraphs.

Definition 2.1.6 (connected component)**.** A *connected component* of a simple graph G is a maximal subgraph H that is connected.

In particular, we are interested in combining connected components to obtain the original graph.

Definition 2.1.7 (separation)**.** A *separation* of a simple graph $G = (V, E)$ is a non-trivial disjoint partition of its vertices $V = U \dot\cup W$ with $U, W \neq \emptyset$, such that no edge connects a vertex in U with a vertex in W.

We note that a graph that has no separation is connected. We will see that the concept of connectedness in a general topological space is defined very similarly in the following Section.

2.1.2 Topological spaces

The notion of topological spaces arose from the generalisation of the study of the real line and Euclidian space [31]. We define a few basic concepts needed to understand notions such as simplicial complexes, which will be important in the following Sections. We begin by defining a topology.

Definition 2.1.8 (topology)**.** A *topology* on a set $\mathbb{X}$ is a collection $\mathcal{T}$ of subsets of $\mathbb{X}$ with the following properties:

i. $\emptyset$ and $\mathbb{X}$ are in $\mathcal{T}$.
ii. The union of elements of any subcollection of $\mathcal{T}$ is in $\mathcal{T}$.
iii. The intersection of elements of any finite subcollection of $\mathcal{T}$ is in $\mathcal{T}$.

We call sets that belong to the collection $\mathcal{T}$ *open sets* of $\mathbb{X}$.

The set $\mathbb{X}$ can, for example, be a set of points but it can also be an uncountable set such as the real line. The definition of a topological space follows naturally.

Definition 2.1.9 (topological space)**.** A *topological space* is an ordered pair $(\mathbb{X}, \mathcal{T})$ that consists of a set $\mathbb{X}$ and a topology $\mathcal{T}$ on $\mathbb{X}$.

Remark 2. We often refer to $\mathbb{X}$ as the topological space.

Example 2.1.1. One of the simplest examples of a topological space is the real line with all open intervals.

As with graphs, we are interested in connected components of a topological space. However, in this case, these are not defined based on the existence of paths. [1]

[1] Path connectedness as a concept does exist on topological spaces, but it is a far stronger property than connectedness and we will not be needing it further. See [31] for further details.

Definition 2.1.10 (separation)**.** A *separation* of a topological space $\mathbb{X}$ is a disjoint partition $\mathbb{X} = U \dot{\cup} W$ into two non-empty, open subsets. We say that a topological space is *connected* if there exists no separation of $\mathbb{X}$.

Connectedness is an important topological property that stays invariant under continuous functions. This will be relevant for our methods used in Chapter 4. For further background on these topological concepts also see [25, 31].

2.1.3 Simplicial complexes

One can represent the underlying structures of topological spaces by partitioning the space into smaller and topologically simpler pieces, which when assembled back together carry the same aggregate topological information as the original space. These building blocks can not be chosen arbitrarily, but we will demand that they fulfil a certain set of properties. For example, we will want that the pieces intersect only in smaller pieces with the same properties. There are two different approaches that one can take to decompose a topological space [20]. One can either choose to use a small number of complicated pieces or we can use a large number of simple pieces. From a computational point of view, the latter is preferable [20] so we take this perspective.

A simple example for such a construction is the tetrahedron in Euclidian space, which consists of four triangular faces that are bounded by three edges (which each connect two points). The triangular faces intersect in edges only, and the edges themselves intersect in the four points of the tetrahedron. One can look at the tetrahedron as a sort of simplified version of a 2-sphere, as it carries the same topological properties (e.g. connectedness, enclosing a hole) as the sphere.

We can now easily imagine using triangles as building blocks to build more complex constructs resembling for example a torus or some other manifold.[2] While representing the underlying manifold in a simplified way, these constructs carry the same topological properties as the manifold. Note that one can consider the triangle constructs as a version of the manifold with a network on its surface.

To properly grasp these concepts, we need additional definitions. For the discussion, we consider the space $\mathbb{R}^d$ with dimension $d \in \mathbb{N}$.

Definition 2.1.11 (affine combination and affine hull)**.** Let $\mathcal{U} = \{u_0, u_1, \ldots, u_k\}$ be points in $\mathbb{R}^d$. A point $x \in \mathbb{R}^d$ is an *affine combination* of the points $u_i \in \mathcal{U}$, $0 \leq i \leq k$, if there exist $\lambda_i \in \mathbb{R}$ such that

[2] See Definition A.1.4 in the Appendix.

i. $x = \sum_{i=0}^{k} \lambda_i u_i$,
ii. $\sum_{i=0}^{k} \lambda_i = 1$.

We call the set of all affine combinations of $\mathcal{U}$ the *affine hull* of $\mathcal{U}$.

To ensure uniqueness of the affine combination, we introduce the following definition.

Definition 2.1.12 (affinely independant)**.** Let $\mathcal{U} = \{u_0, u_1, \ldots, u_k\}$ be points in $\mathbb{R}^d$. The $k+1$ points in $\mathcal{U}$ are said to be *affinely independent*, if the vectors $\{u_i - u_0 : 0 \leq i \leq k\}$ are linearly independent.

Remark 3. In $\mathbb{R}^d$ there are at most $d+1$ independent points.

Example 2.1.2. Any two distinct points in $\mathbb{R}^2$ are affinely independent. Similarly, any three points in $\mathbb{R}^2$ are affinely independent, provided they do not lie on the same straight line.

Convex combinations and hulls are a special case of affine combinations:

Definition 2.1.13 (convex combination and convex hull)**.** An affine combination $x = \sum_{i=0}^{k} \lambda_i u_i$ is a *convex combination*, if $\lambda_i \geq 0$ for all $0 \leq i \leq k$. The set of all convex combinations of the points in $\mathcal{U}$ is called the *convex hull* of $\mathcal{U}$.

Example 2.1.3. A triangle spanned by three points $u_0, u_1, u_2 \in \mathbb{R}^2$ is the convex hull of these points.

We can now finally define a k-simplex:

Definition 2.1.14 (k-simplex)**.** A *k-simplex* $\sigma = [u_0, u_1, \ldots, u_k]$ is the convex hull of the $k+1$ affinely independent points $u_0, u_1, \ldots, u_k \in \mathbb{R}^d$. We call k the *dimension* of the simplex.

Example 2.1.4. In Figure 2.1 we examples of simplices for the first few dimensions: a point is a 0-simplex, an edge is a 1-simplex, a triangle is a 2-simplex and the tetrahedron that we have encountered before is a 3-simplex.

We observe that the lower-dimensional simplices from example 2.1.4 appear to be contained in the higher dimensional simplices. This is due to the fact that subsets of affinely independent points are also affinely independent. Moreover, the lower dimensional simplices form so called *faces* of the higher dimensional objects:

Definition 2.1.15 ((proper) faces and cofaces)**.** A *face* τ of a k-simplex σ is the convex hull of a subset $\mathcal{V} \subseteq \mathcal{U}$. We call the face *proper*, if the subset is proper. If τ is a (proper) face, we call σ a *(proper) coface* of τ.

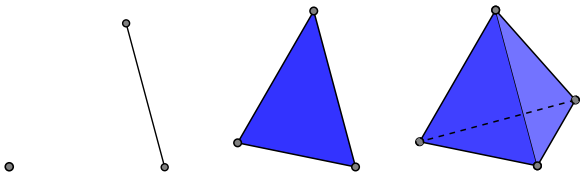

Figure 2.1: Examples of a 0-simplex, a 1-simplex, a 2-simplex and a 3-simplex (from left to right) [20].

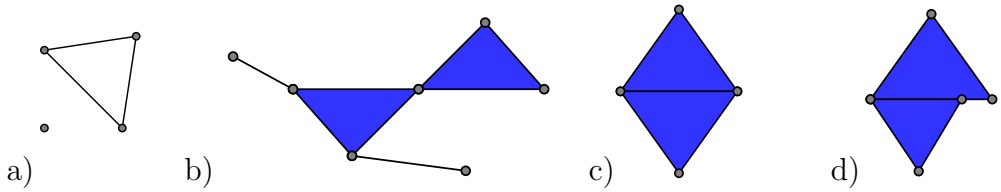

Figure 2.2: a), b) and c) are examples of simplicial complexes. The collection of simplices we show in d) is not a simplicial complex. The colours are used to indicate 2-simplices.

Remark 4. We use the notation $\tau \leq \sigma$ to denote a face of σ and $\tau < \sigma$ to denote a proper face of σ.

Remembering the building blocks we described in the beginning of this Section, we can ask ourselves whether it is only possible to build shapes using 2-simplices (i.e. triangles) or whether we could also combine these with higher- or lower-dimensional simplices. The result such a combination is called a *simplicial complex*:

Definition 2.1.16 (simplicial complex)**.** A *simplicial complex* is a finite collection of simplices Σ such that

i. If $\sigma \in \Sigma$ and $\tau \leq \sigma$, it follows that $\tau \in \Sigma$.

ii. If $\sigma, \tilde{\sigma} \in \Sigma$, it follows that the intersection of both simplices is either the empty set or a face of both.

Examples 2.1.1.

1. The simplest example of a simplicial complex is a simplex.

2. In Figure 2.2 we show examples of simplicial complexes. Example a) illustrates that simplicial complexes are not necessarily equal to simplices. The three edges do not form a 2- simplex, but form a simplicial complex consisting of 1-simplices. In examples b) and c), all 1- and 2-simplices are connected by 0-simplices. Example d) is a collection of simplices that violates the definition of a simplicial complex because the intersection between the two triangles does not consist of a complete edge. Note that any combination of the three simplicial complexes a), b) and c) is again a simplicial complex.

We take the *dimension* of Σ to be the dimension of its highest-dimensional simplex. Simplicial complexes can also be defined in a more abstract way (see [20]). However, this is not relevant for the methods we use. Simplicial complexes can be used to represent topological spaces as described in the following definitions.

Definition 2.1.17 (underlying space of a simplicial complex)**.** The *underlying space* $|\Sigma|$ of a simplicial complex Σ is the union of its simplices together with the topology inherited from the Euclidian space, in which the simplicial complex is defined.

Definition 2.1.18 (triangulation)**.** A *triangulation* of a topological space $\mathbb{X}$ is a simplicial complex Σ together with a homeomorphism[3] $\phi : \mathbb{X} \longrightarrow |\Sigma|$. We call a topological space *triangulable* if a triangulation exists.

We now see that the tetrahedron in the beginning of the chapter, was in fact a triangulation of the 2-sphere. We can imagine the homeomorphism between the two objects to fill the tetrahedron with air until it is completely round. It is important that the map is a homeomorphism, because topological properties such as connectedness are invariant under homeomorphisms and we thereby ensure that the topological space and its triangulation carry the same topological properties. One can also shown that every compact surface[4] is triangulable [31].

2.1.4 Homology and Betti numbers

Homology is a formal way of quantitatively detecting loops and holes in various dimensions to give insight into the way a topological space is connected. It for example makes it possible to distinguish a 2-sphere from a torus by capturing the fact that one can contract any one-dimensional loop on the sphere to a point, whereas there are two distinct loops on the torus surface, that cannot be continuously deformed into each other. Moreover, these loops can not be contracted to a point since they surround different holes.

Even though homology is not the only and most detailed formalism that can be used for this, it so far has the fastest algorithms [20]. *Homology groups*, which are topological invariants of a space, and *Betti numbers* (which are derived from them) play the key role in this endeavour. Homology groups detect holes and loops indirectly by looking at the space surrounding them, whereas Betti numbers give a way to count the number of distinct loops and holes. We start constructing the homology groups, by looking at formal sums of simplices, which themselves form an Abelian group. [5]

[3] See Definition A.1.1 in the Appendix.
[4] See Definition A.1.5 in the Appendix.
[5] See Definition A.1.6 in the Appendix.

Definition 2.1.19 (p-chain)**.** Let Σ be a a simplicial complex, let p be a given dimension and let G be an Abelian group. A *p-chain* c is a formal sum of p-simplices in Σ:

$$c = \sum_{i \in I} a_i \sigma_i, \tag{2.1}$$

where $a_i \in G$ are coefficients, σ_i are p-simplices, and I is an index set.

For a more general version of the definition, see [10].

Remark 5.
i. In a more theoretical context $\mathbb{Z}$ is a common choice for the commutative group G. In computational topology, the commutative group G is usually given by $\mathbb{Z}/2\mathbb{Z}$. Using $\mathbb{Z}/2\mathbb{Z}$ has the advantage, that we can regard p-chains as subsets of the set of all p-simplices in Σ by assigning the coefficient 1 to simplices that form part of the subset and the coefficient 0 to those not in the subset. Moreover, because $\mathbb{Z}/2\mathbb{Z}$ is in fact a field, we can also think of p-chains as elements of a vector space.
ii. We allow all possible integer values for p. Note that for $p = 0$ and $p > \dim \Sigma$ however, we only obtain trivial p-chains.
iii. We use $\mathcal{C}_p = \mathcal{C}_p(\Sigma)$ to denote the set of all p-chains of a simplicial complex Σ.

Example 2.1.5. We consider the simplicial complex in Figure 2.3 and assume that we are working with coefficients in $\mathbb{Z}/2\mathbb{Z}$. The 2-chains of the simplicial complex are $\{0$, the blue 2-simplex pointing upwards, the blue 2-simplex pointing downwards, the union of both 2-simplices$\}$. The 1-chains are given by the set of all subsets of the 11 edges.

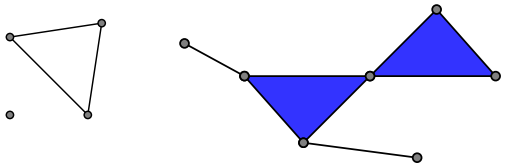

Figure 2.3: Example of a simplicial complex. The blue 2-simplices are examples of 2-chains, the 11 edges are examples of 1-chains of the simplicial complex.

We define the summation of two p-chains, $c = \sum_{i \in I} a_i \sigma_i$ and $c' = \sum_{i \in I} b_i \sigma_i$, on Σ componentwise:

$$c + c' = \sum_{i \in I} (a_i + b_i) \sigma_i. \tag{2.2}$$

This leads us to an important property of p-chains.

Proposition 2.1.1. *The set of p-chains of a simplicial complex Σ together with summation forms an Abelian group $(\mathcal{C}_p, +)$.*

Proof. i) Summation on $\mathcal{C}_p$ is associative. Because we have defined summation to be componentwise, this property is inherited from the fact that the coefficients a_i are elements of a commutative group.

ii) Existence of an element neutral under summation. Because the commutative group of coefficients has a neutral (i.e. identity) element e, we naturally can define the neutral element of $\mathcal{C}_p$ as $\sum_{i\in I} e\sigma_i$.

iii) Existence of inverse elements: The inverse elements are also inherited from the commutative group. We define the inverse of a p-chain $c = \sum_{i\in I} a_i\sigma_i$ to be $-c = \sum_{i\in I}(-a_i)\sigma_i$.

iv) Summation on $\mathcal{C}_p$ is commutative. This is again inherited from the commutativity of the summation of the coefficients.

□

We observe when working with coefficients from $\mathbb{Z}/2\mathbb{Z}$ that the sum of two p-chains results in the sum of all p-simplices in which the two original p-chains differ. The p-simplices which the two p-chains have in common will be present in the sum twice and therefore vanish by the properties of addition on $\mathbb{Z}/2\mathbb{Z}$. From now on, we assume that the coefficients of our p-chains are taken from $\mathbb{Z}/2\mathbb{Z}$, which will simplify some the following definitions and proofs. For their corresponding more general versions, see [25]. The following definition will help to relate the different p-chain groups of a simplicial complex.

Definition 2.1.20 (boundary of a p-simplex)**.** We define the *boundary* $\partial_p\sigma$ of a p-simplex $\sigma = [u_0, u_1, \ldots, u_p]$ as the formal sum of its $(p-1)$-dimensional faces:

$$\partial_p\sigma = \sum_{j=0}^{p} [u_0, \ldots, \hat{u}_j, \ldots, u_p], \tag{2.3}$$

where $\hat{u}_j$ denotes the point not included in spanning the simplex.

We can naturally extend this definition to p-chains by defining the boundary of a p-chain $c = \sum_{i\in I} a_i\sigma_i$ as $\partial c = \sum_{i\in I} a_i\partial\sigma_i$. We can now construct a family of boundary homomorphisms[6] ∂_p between the different groups of p-chains of a simplicial complex

[6]See Definition A.1.9 in the Appendix.

by mapping p-simplices to their boundaries:

$$\ldots \xrightarrow{\partial_{p+2}} \mathcal{C}_{p+1} \xrightarrow{\partial_{p+1}} \mathcal{C}_p \xrightarrow{\partial_p} \mathcal{C}_{p-1} \xrightarrow{\partial_{p-1}} \ldots \xrightarrow{\partial_1} \mathcal{C}_0$$
$$c \longmapsto \partial c.$$

By construction, taking the boundary of a p-chain satisfies the property $\partial_p(c + c') = \partial_p c + \partial_p c'$. Thus ∂_p is indeed a homomorphism. We call such a sequence of chains and homomorphisms a *chain complex*. The following theorem states a crucial property of the boundary homomorphisms in a chain complex.

Theorem 2.1.2. *Let $d \in \mathcal{C}_{p+1}$. It holds, that*

$$\partial_p \partial_{p+1} d = 0. \tag{2.4}$$

Proof. It is sufficient to show that statement (2.4) holds for a $(p+1)$-simplex $\tau = [u_0, \ldots, \hat{u}_j, \ldots, u_{p+1}]$:

$$\partial_p \partial_{p+1} \tau = \partial_p \sum_{j=0}^{p+1} [u_0, \ldots, \hat{u}_j, \ldots, u_{p+1}] \tag{2.5}$$
$$= \sum_{j=0}^{p+1} \partial_p [u_0, \ldots, \hat{u}_j, \ldots, u_{p+1}] \tag{2.6}$$
$$= \sum_{j=0}^{p+1} \sum_{i=0}^{p+1} [u_0, \ldots, \hat{u}_i, \ldots, \hat{u}_j, \ldots, u_{p+1}] \tag{2.7}$$
$$= 0. \tag{2.8}$$

The last step follows from the fact, that we are working on $\mathbb{Z}/2\mathbb{Z}$ and every $(p-1)$-simplex on the right-hand side appears in the sum as the face of two p-simplices. □

For simplicity, we often denote the boundary homomorphism by ∂, i.e. we omit the specification of p. If we go back to the example of the simplicial complex in Figure 2.3, we see that the 1-chains are formed by edges coming from 1-simplices as well as boundaries of 2-simplices. Given the above Theorem 2.1.2 we should now be able to distinguish between these two sets because applying the boundary homomorphism to the 1-chains should map all edges of 2-simplices to 0. We therefore now look at two subgroups of $(\mathcal{C}_p, +)$ that are based exactly on this fact. Together with Theorem 2.1.2 they form the main ingredients in constructing the homology group of a simplicial complex.

Definition 2.1.21 (p-cycle)**.** A *p-cycle* is a p-chain c that satisfies $\partial c = 0$.

We denote the set of p-cycles as $\mathcal{Z}_p$.

Proposition 2.1.3. *$(\mathcal{Z}_p, +)$ is a subgroup[7] of $(\mathcal{C}_p, +)$.*

Proof. This follows immediately from the fact that the boundary operator is a homomorphism. Let $a, b \in \mathcal{Z}_p$. It follows that $a + b$ is also an element of $\mathcal{Z}_p$ because $\partial(a + b) = \partial a + \partial b = 0$. Similarly, for $a \in \mathcal{Z}_p$ it follows that $\partial(-a) = -\partial a = 0$. Hence, $\mathcal{Z}_p$ is a subgroup of $\mathcal{C}_p$. □

Remarks 1.
i. Since $(\mathcal{C}_p, +)$ is an Abelian group, it follows immediately that $(\mathcal{Z}_p, +)$ is also Abelian.
ii. An equivalent definition for the group of p-cycles $\mathcal{Z}_p$ is $\mathcal{Z}_p = \ker \partial_p$, where $\ker \partial_p$ denotes the kernel[8] of ∂_p.

Definition 2.1.22 (p-boundary)**.** A *p-boundary* is a p-chain c for which there exists a $(p + 1)$-chain with $\partial d = c$.

We denote the set of p-boundaries as $\mathcal{B}_p$.

Proposition 2.1.4. *$(\mathcal{B}_p, +)$ is a subgroup of $(\mathcal{C}_p, +)$.*

Proof. Let $a, b \in \mathcal{B}_p$. By definition there exist $\tilde{a}, \tilde{b} \in \mathcal{C}_{p+1}$ such that $\partial\tilde{a} = a$ and $\partial\tilde{b} = b$. It must therefore hold that $a + b = \partial\tilde{a} + \partial\tilde{b}$ and hence that $a + b = \partial(\tilde{a} + \tilde{b})$. Similarly, with $a \in \mathcal{B}_p$, we obtain $-a = -\partial\tilde{a} = \partial(-\tilde{a})$. □

Remarks 2.
i. Because $(\mathcal{C}_p, +)$ is an Abelian group, it follows immediately that $(\mathcal{B}_p, +)$ is also Abelian.
ii. An equivalent definition for the group of p-boundaries $\mathcal{B}_p$ is $\mathcal{B}_p = \text{Im } \partial_{p+1}$, where $\text{Im } \partial_{p+1}$ denotes the image[9] of ∂_{p+1}.

Using Theorem 2.1.2 we can now relate the subgroups to each other: By Theorem 2.1.2, we have $\partial_p(\text{Im } \partial_{p+1}) = 0$, so $\mathcal{B}_p \subseteq \mathcal{Z}_p$. It is easy to show that $\mathcal{B}_p$ is indeed a subgroup of $\mathcal{Z}_p$.

Examples 2.1.2. Consider the 1-chains of the simplicial complex in Figure 2.3. We observe that the boundaries of the blue 2-simplices are elements of the boundary subgroup B_1. These are, however, not the only 1-chains that form part of the kernel of ∂_0 and thus $\mathcal{Z}_0$. If we consider the loop that consists of three edges at the left-hand

[7] See Definition A.1.7 in the Appendix.
[8] See Definition A.1.10 in the Appendix.
[9] See Definition A.1.11 in the Appendix.

side of the simplicial complex, we can see that every vertex in this loop forms part of the boundary of two separate edges. Because we are working with coefficients from $\mathbb{Z}/2\mathbb{Z}$, we see that these vertices are not part of the image of ∂_0. This is not the case for any of the other vertices. These vertices are therefore mapped non-trivially to the set $\mathcal{B}_{-1}$, i.e. the set of vertices that form part of the boundaries of edges.

Example 2.1.2 illustrates how one-dimensional loops[10] behave differently from other edges. Any parts of their boundaries vanish completely when applying ∂_0. This does not happen to any other 1-chain, because even long tails of vertices and edges map to at least one 0-chain if their ends are not joined. We have thus come very close to our goal of being able to count holes and loops. Although we have so far identified the subgroup, to which such loops will belong, this subgroup still also contains the boundaries of higher dimensional chains. To differentiate between the two subgroups, we need to define the p-th homology group of a simplex.

Definition 2.1.23 (p-th homology group)**.** The *p-th homology group* $\mathcal{H}_p$ of a simplicial complex Σ is the quotient group[11] of the group of p-cycles $\mathcal{Z}_p$ modulo the group of boundaries $\mathcal{B}_p$:

$$\mathcal{H}_p = \mathcal{Z}_p/\mathcal{B}_p.$$

Two p-cycles in the p-th homology group are regarded as different if they differ by more than just a boundary. Otherwise, the quotient group treats them as being in the same homology class. Two elements in the same homology class are said to be *homologous*. For example, all p-boundaries are homologous to the $\emptyset$-chain. Every hole of dimension p found in a simplicial complex is surrounded by at least one p-cycle in the homology group. Counting the number of classes in $\mathcal{H}_p$ thus gives an estimate of the number of p-dimensional loops of a simplicial complex. However, cycles that surround the same hole are counted separately. A solution to this problem is to count the minimal number of elements needed to generate the group.[12] This leads to the definition of p-th Betti number.

Definition 2.1.24 (p-th Betti number)**.** The *p-th Betti number* β_p of a simplicial complex is defined by

$$\beta_p = \operatorname{rank} \mathcal{H}_p.$$

[10] We use the term loop to describe connected chains which are not faces of a higher dimensional simplex and can be mapped to an k-dimensional sphere by a homomorphism for some dimension $k \in \mathbb{N}$. Note that this corresponds to what we intuitively would call a loop or hole for dimensions 1 and 2.

[11] See Definition A.1.13 in the Appendix.

[12] See Definition A.1.14 in the Appendix.

Remark 6. Recall that we are working with coefficients from $\mathbb{Z}/2\mathbb{Z}$. This turns the set of p-cycles into a vector space and we can think of the homology group $\mathcal{H}_p$ as a quotient vector space. The p-th Betti number is then given by the dimension of this vector space.

One can interpret the first three Betti numbers, β_0, β_1 and β_2, to represent respectively the number of connected components, the number of 1-dimensional loops and the number of 2-dimensional holes in a simplicial complex.

2.2 Persistent homology

We now explain how to use Betti numbers of a simplicial complex for the analysis of graphs. We base our discussion on [16, 19, 22].

2.2.1 Filtrations

We first define what we mean by a subcomplex of a simplicial complex Σ.

Definition 2.2.1 (subcomplex of a simplicial complex)**.** A *subcomplex* of a simplicial complex is a subset of simplices that satisfy the properties of a simplicial complex.

We can now build sequences of simplicial complexes that form subcomplexes of each other.

Definition 2.2.2 (filtration)**.** A *filtration* of a simplicial complex Σ is a nested sequence of subcomplexes starting with the empty complex $\emptyset$ and ending with the full simplicial complex:

$$\emptyset = \Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \cdots \subseteq \Sigma_k = \Sigma. \tag{2.9}$$

We observe that there are natural inclusion maps $i_j : \Sigma_j \hookrightarrow \Sigma_{j+1}$ that can be defined along the filtration.

Example 2.2.1. In Figure 2.4 we show a simple example of a filtration. For practical reasons, we have not included the $\emptyset$-complex in the filtration. The filtration starts with four 1-simplices that form a square. In the second filtration step, a 0-simplex appears in addition to the square. We see that the first filtration step is embedded in the second. In the third and fourth filtration steps the "framework for the roof" of the house is added to the simplicial complex. In the final step, the collection of 1-simplices that previously formed the "framework of the roof" become faces of a 2-simplex, that forms the full "roof".

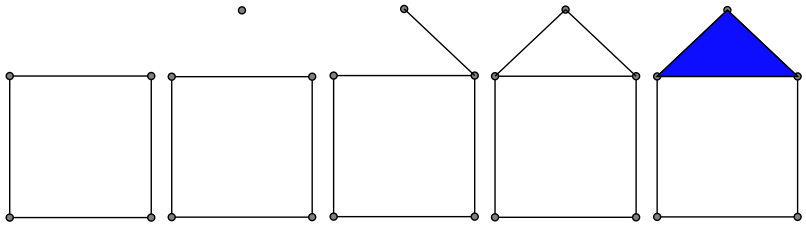

Figure 2.4: Simple filtration example: House (modified from [1]).The filtration steps are shown in numerical order starting with filtration step 1 on the left-hand side.

We now use the same concept on a graph. Similar to before, we define a filtration of graphs to be a nested sequence of subgraphs. In Figure 2.5 we show an example of a graph filtration on a complete graph. For graphs we often think of the 0-th filtration

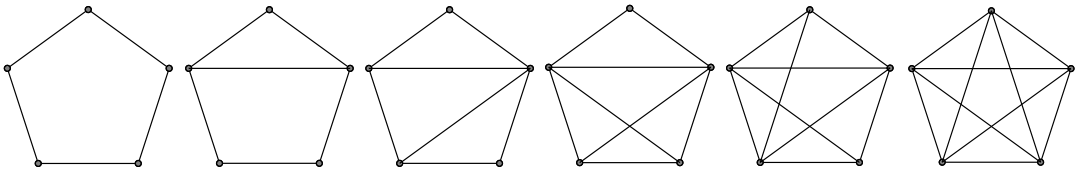

Figure 2.5: Graph filtration example: Pentagon. The filtration steps are shown in numerical order starting with filtration step 1 on the left-hand side.

step as the set of vertices instead of the $\emptyset$-complex. As a consequence, no vertices appear in later filtration steps, even though we have seen in the House filtration example that this not pathological for general filtrations of simplicial complexes.

In order to apply topological tools to a graph filtration, we need to define simplicial complexes on the graph. We assume that our filtrated graph is embedded in a higher dimensional space, such that its edges do not cross. The simplest way to define simplicial complexes on the graph, is to look at the edges and vertices as 1- and 0-simplices forming a simplicial complex. The graph filtration thus already is a filtration of simplicial complexes. We are now interested in when a prominent feature, for example a homology class, of the simplicial complex first appears in the filtration and when it disappears.

Definition 2.2.3 (birth and death of a homology class, persistence)**.** A homology class $h \in \mathcal{H}_p(\Sigma)$ is *born* at Σ_m, if h is an element of $\mathcal{H}_p(\Sigma_m)$ but it is not in the image of the inclusion map $i_{m-1} : \Sigma_{m-1} \hookrightarrow \Sigma_m$.
A homology class $g \in \mathcal{H}_p(\Sigma)$ *dies* entering Σ_n, if g is an element of $\mathcal{H}_p(\Sigma_{n-1})$ but it is not in the image of the inclusion map $i_{n-1} : \Sigma_{n-1} \hookrightarrow \Sigma_n$.
We denote the filtration step in which h is born by m_h and the filtration step in which h dies by n_h. We define the *persistence* of a homology class $h \in \mathcal{H}_p(\Sigma)$ as

$$p_h = n_h - m_h.$$

Persistence was first used as a measure to rank topological features by their life-time within a filtration in [18].

Examples 2.2.1. 1. In the filtration example from Figure 2.4, the 1-loop that forms the framework of the roof before it becomes a 2-simplex has persistence 1: It is born in the fourth filtration step and dies when entering the fifth filtration step. The 1-loop that forms the basis of the house lives until the very end of the filtration. By convention, we say that it has persistence ∞. Similarly, we examine the connected components, the "0-loops", of the filtration. The vertex that appears in the second step of the filtration only forms a separate connected component during that filtration step and it dies when entering the next filtration step. Its persistence is therefore 1. The connected component that exists in the filtration from the first step onwards is the base of the house. It has persistence ∞.

2. In the filtration example from Figure 2.5, the loop that forms in the beginning of the filtration has persistence ∞. In every following step, an additional 1-dimensional loop with persistence ∞ is added to the homology group.

2.2.2 Barcodes

We now introduce a visualisation of persistence of homology classes of a filtration of simplicial complexes, so called *barcodes* [22].

Definition 2.2.4 (persistence interval)**.** The *persistence interval* for a homology class $h \in \mathcal{H}_p(\Sigma)$ is given by $[m_h, n_h]$.

One can think of a barcode for a filtration of a simplicial complex as a parametrised version of Betti numbers depicting persistence intervals of homology classes.

Definition 2.2.5 (barcode)**.** A p-dimensional *barcode* for a filtration of a simplicial complex is a collection of horizontal line segments on a plane representing the persistence intervals of homology generators of the p-th homology group arbitrarily ordered along the vertical axis.

We read a p-dimensional barcode as follows: Every line segment represents a p-loop, the length of the segment indicates its persistence in the filtration. We can trace the filtration step in which the loop is born and in which it dies on the x-axis.

Examples 2.2.2. 1. We show the barcode for the house filtration from Figure 2.4 in the left-hand side of Figure 2.6. The 0-barcode consists of two line segments that represent a connected component of persistence 1 that is born in filtration step 1 and dies in filtration step 2 and a connected component of persistence ∞ that is born in the beginning of the filtration. These correspond to the vertex added in filtration step 2 and the base of the house respectively. In the 1-barcode the "framework for the roof" of the house is represented by a short lived 1-loop of persistence 1 before it gets covered by the blue 2-simplex. The base of the house corresponds to the line segment of persistence ∞.

2. We show the barcode representation for the pentagon filtration from Figure 2.5 on the right-hand side of Figure 2.6. Since the 0-th filtration step consists of the 5 vertices of the graph, we can see 5 connected components in the 0-barcode that merge into one connected component in the first step of the filtration. The 1-dimensional barcode shows the 1-loops that are born in every step of the filtration and persist until the end.

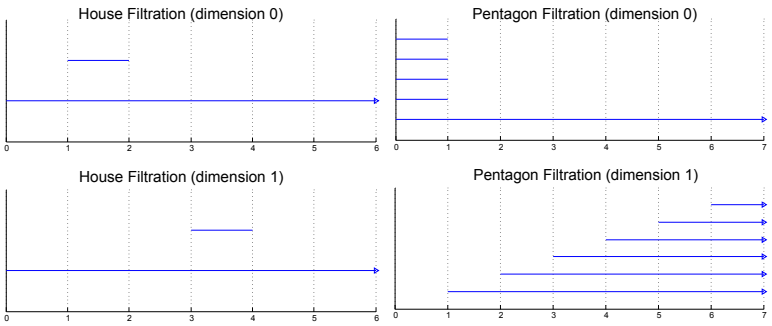

Figure 2.6: Betti barcodes for the house filtration and the pentagon filtration.

For a graph the amount of connected components in the 0-dimensional barcodes will always consist of the number of nodes in the graph. The death of a connected component in the barcode can signify one of two possible situations: it either represents a node's first connection with some part of the graph or it represents the connection of two subgraphs. With a barcode alone it is impossible to distinguish between these two cases. However, the death of a connected component also indicates more than just that two components have merged. If the difference between the persistence of two succeeding components is large, we can deduce that the connections made by the graph in the meantime are all made within existing subgraphs.

Persistent homology is not necessarily restricted to the application on data in the form of graphs but has in fact more often been used for point cloud data [16, 22].

2.2.3 Simplicial complexes for point cloud data

Point cloud data consists of an unordered sequence of points $S = \{x_1, \ldots, x_k\}$ embedded in an n-dimensional Euclidian space $\mathbb{E}^n$. One defines simplicial complexes on point cloud data by considering each point in the metric space as a vertex of a graph. Two vertices are connected by an edge based on their proximity. Higher-dimensional simplices can then be defined on the graph in different ways. One of the most commonly used complexes is the *Vietoris-Rips* complex [22].

Definition 2.2.6 (Vietoris-Rips complex)**.** Let $S = \{x_1, \ldots, x_m\}$ be a collection of points in the Euclidian space $\mathbb{E}^n$ and ϵ a given distance. The *Vietoris-Rips complex* $\mathcal{R}_\epsilon$ is the simplicial complex, whose k-simplices are defined by unordered $(k+1)$-tuples of points $\{x_i\}_{i=0}^k$, whose pairwise distance is at most ϵ.

The choice of the distance ϵ determines important properties of the resulting simplicial complex: If we choose ϵ to be very small we obtain a discrete set while for a very large epsilon all points are joined in a single high-dimensional simplex. A filtration of Vietoris-Rips complexes is therefore constructed by varying ϵ [16, 22]. We show an example of Vietoris-Rips complex filtration with a corresponding barcode in Figure 2.7. As ϵ increases, more vertices are connected by edges. The first two connected

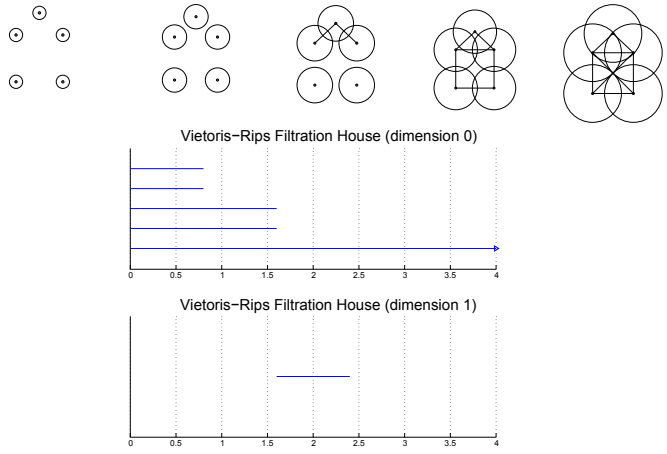

Figure 2.7: Vietoris-Rips complex and barcodes for the house filtration [1]. The value of ϵ increases over the filtration steps and is shown on the x-axis of the barcode.

components die in filtration step 3, when the vertices forming the "roof" of the house are joined. In filtration step 4, all vertices are part of one connected component. At the same time, a 1-loop is born in the square forming the "base" of the house. Is dies in the subsequent filtration step when all nodes in the base are connected to each other.

Chapter 3

Model Networks and Data

The main goal of the present dissertation is to analyse time-dependant network data generated from neuroscientific experiments using methods from computational topology. Since experimental data has a tendency to be influenced by a variety of mathematically unpredictable factors, we first use the methods on the output of a well-studied dynamical system, the Kuramoto model, which we analyse on a network. This will help to shape our intuition how to interpret the results of applying these methods and provide insight into what kinds of properties one can detect.

3.1 The Kuramoto model

The Kuramoto model is a well-studied non linear model for a large set of coupled limit-cycle oscillators with distinct natural frequencies traditionally drawn from some prescribed distribution [4, 23, 26, 40]. While the Kuramoto model can be regarded as a toy model, the properties of the model are well-known and hence a good choice for this study.

3.1.1 The basic model

The Kuramoto model was developed in the 1970s to understand the phenomenon of collective synchronisation of a large system of oscillators and it has been used as a toy model by neuroscientists because a few underlying characteristics of the synchronisation patterns bear resemblance to those found in neuronal communities [12]. It also has a variety of further applications in chemical or biological contexts [4, 23, 40, 41]. If not declared otherwise, the following discussion is based on [40].

The governing equations of the model are most commonly written as

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i), \quad \text{for } i = 1, \ldots, N, \tag{3.1}$$

where θ_i denotes the phase of oscillator i, the parameter ω_i is its natural frequency, $K \geq 0$ parametrises the coupling strength between different oscillators, and N is the number of oscillators in the model. The normalisation factor $\frac{1}{N}$ ensures that the equations are bounded for $N \to \infty$. The distribution from which the frequencies ω_i are drawn is usually assumed to be unimodal and symmetric about its mean frequency, which can be set to 0 due to the rotational symmetry of the model (Equation (3.1) stays invariant under a translation of θ_i). The parameter ω_i then denotes the deviation from the mean frequency.

One can summarise the dynamics of the model by a parameter r that gives the collective rhythm and ψ that represents the mean phase. Both parameters are time-dependent and satisfy the relation

$$re^{\mathrm{i}\psi} = \frac{1}{N}\sum_{j=1}^{N} e^{\mathrm{i}\theta_j}. \tag{3.2}$$

If the phases θ_i are evenly distributed around the circle, their centroid lies in the middle of the circle, so $r \approx 0$. If however, the phases are close together such that the centroid lies almost on the boundary of the unit circle, then $r \approx 1$ and the oscillators resemble a single coherent oscillator that moves around the unit circle. One can write Equation (3.1) as

$$\frac{d\theta_i}{dt} = \omega_i + Kr\sin(\psi - \theta_i), \quad \text{for } i = 1, \ldots, N, \tag{3.3}$$

using the parameters in Equation (3.2). Although every oscillator is an own entity, it is coupled to all other oscillators via the parameters r and ψ. Equation (3.3) illustrates that an individual oscillator is influenced primarily by the mean phase and the centroid of the phases rather than by any individual oscillator. The interaction between one oscillator and the other oscillators is stronger if r is large. This can lead to a positive feedback loop since for a growing phase coherence the coupling increases, which in turn leads to an even stronger aggregate coherence. The process can, however, also be self-limiting, because oscillators can synchronise with the rest of the population without leading to an increased coherence in the population as a whole.

It has been shown that there is a critical coupling strength thereshold K_c determining whether or not the oscillators tend to synchronise [26, 27]. For values $K \leq K_c$ we find that the coherence r decays quickly over time to a very small value so that the oscillators almost decouple completely. For $K \geq K_c$ however, the phase coherence r reaches a level close to 1. We use this property to test codes that were written to simulate the Kuramoto model.

3.1.2 The Kuramoto model in a network setting

We can adapt Equation (3.1) to create a network of N oscillators by introducing binary coupling between the oscillators, which is an approach that was used in [3, 4, 6]. We consider the following generalised version of Equation (3.1):

$$\frac{d\theta_i}{dt} = \omega_i + \sum_{j=1}^{N} \kappa A_{ij} \sin(\theta_j - \theta_i), \quad i = 1, \ldots, N, \tag{3.4}$$

where $\kappa \geq 0$ denotes the normalised coupling strength and the entries of the coupling matrix $A = (A_{ij})_{i,j=1}^{N}$ indicate whether oscillators i and j are coupled (i.e. $A_{ij} = 1$), or not (i.e. $A_{ij} = 0$). Note that Equation (3.4) can be even more general by using different coupling strengths $\kappa_{i,j}$ for every different pair of oscillators or by considering a functions other than sine on the right-hand side.

We use the same constants and underlying set up as in [6]. We choose the coupling strength to be $\kappa = 0.2$, the number of oscillators to be $N = 128$, and we draw the natural frequencies ω_i from a Gaussian distribution with mean 0 and standard deviation 1. We construct an underlying network by dividing the oscillators into 8 separate communities [1] of 16 distinct oscillators each and let every oscillator have exactly 14 connections, 13 of which are with oscillators in the same community and 1 outside of the community.[2] We call this network a *structural network* in analogy to neuronal data, as we will see later. In figure 3.1 we illustrate the planted community structure.

We observe the system for $M + 1$ time steps until time $T = 10$ and create time vectors $\tau_i = (\theta_i(t_0), \ldots, \theta_i(t_M))$ for every oscillator θ_i. These give a *time-series* for each oscillator. To quantify the pairwise synchrony of two oscillators i and j throughout a set number of $k \leq M$ time steps, we partition the time-series into vectors $\tau_i^{\hat{k}}$,

[1] In this context, we use the term *community* to denote a set of densely connected nodes with little connections to other nodes outside of this set. In network science however, the term can also be used differently. For a detailed discussion see for example [35].

[2] This network set up differs slightly from [6], where every oscillator had at least 13 connections inside its community and at least 1 connection outside of the community.

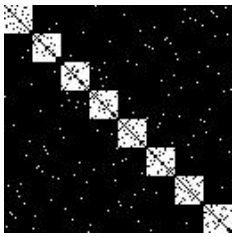

Figure 3.1: Example of a coupling matrix A we use for the interaction between oscillators in the Kuramoto model. Each oscillator has exactly 13 connections to oscillators within its community and exactly 1 connection to an oscillator outside of it.

that consist of k consecutive elements each, and we define the following local measure, which was introduced by [3] and adapted by [6]:

$$\phi_{ij}^{\hat{k}} = \langle |\cos[\tau_i^{\hat{k}} - \tau_j^{\hat{k}}]| \rangle, \tag{3.5}$$

where the angular brackets indicate an average over several simulations, in our case 20. We then use the values ϕ_{ij} to define the edge weights in the fully connected weighted temporal network of Kuramoto oscillators. In analogy to neuronal networks, we call this network a *functional network* for the Kuramoto model. We vary the number $k \leq M$ to create separate time layers of the functional network from the partitioned time-series in our analysis.

We show an illustration of the steps we take to create a functional network from the Kuramoto model in Figure 3.2.

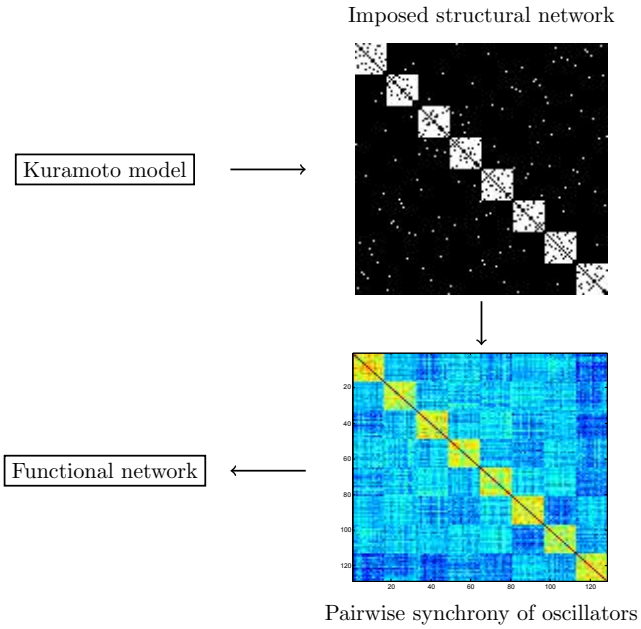

Figure 3.2: Steps for creating a functional network from the Kuramoto model.

3.1.3 Null models for the Kuramoto data

To assess whether structures revealed by the methods that we use are influenced significantly by the actual behaviour of the Kuramoto model and to what extent these can be explained as by random processes, we also analyse two different null models based on the time-series output. The first null model consists of randomly permuting the time-series for every oscillator before computing the similarity measure with Equation (3.5). We refer to this null model as the *simple null model*. The second null model is based on creating surrogate data using a discrete Fourier transformation. This approach, which was introduced for time-series in [37], has the advantage of preserving not only the mean and the variance of the original time-series but also its autocorrelation function. In a first step, we take the discrete Fourier transform of the time-series data τ of length M:

$$\hat{\tau}_n = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} \tau_m e^{\frac{2\pi i n m}{M}}. \tag{3.6}$$

We then construct surrogate data by multiplying the Fourier transform by random phases a_n chosen uniformly from the interval $[0, 2\pi)$ and fulfilling a symmetry property: For every $n \leq M$, there exists $\tilde{n}$ such that $a_n = -a_{\tilde{n}}$. This symmetry ensures that the inverse Fourier transform yields real values. The surrogate data $\sigma = (\sigma_1, \ldots, \sigma_M)$ is thus given by

$$\sigma_m = \frac{1}{\sqrt{M}} \sum_{n=0}^{M-1} e^{\mathrm{i}a_n} \hat{\tau}_n e^{-\frac{2\pi i n m}{M}}. \tag{3.7}$$

We call this null model *Fourier null model*. Both null models were used in [6] and performed differently under various community-detection techniques.

3.2 Neuronal network data

The human brain is a source for real-life networks. Neurons, which are specialised brain cells, receive, process, and transmit signals to each other via networks. Even though this has been known for a long time, graph theory has only been applied to neuronal networks in the past 15 years [13] (for a more recent review, see also [39]). In the following Section we give a brief overview on networks based on data from neuroscientific experiments. We also introduce a particular data set [5], which we will analyse using tools from computational topology in Chapter 4.

3.2.1 The use of graph theory in neuroscience

Treating the brain as a complex system offers the possibility to use mathematical tools that can help identifying key regions in the brain that are involved in various physiological and pathological processes, as well as giving more general information about the structure of such networks. There is for example strong evidence that the brain has an underlying modular structure, i.e. it is organised in small subunits, which can carry out specific functions without influencing the network as a whole [13, 14, 33].

There are two different types of brain networks: *structural* and *functional* networks. In a structural network, the nodes consist of neurons, and the physical connections between them form the edges. Obtaining such information experimentally is laborious and usually requires microscopical techniques, which can only be carried out on a dead subject [32]. Moreover, although structural networks provide information on the underlying architecture of brain connections, they do not provide direct information about the neurophysiological dynamics. Therefore, it is also important to study functional networks. Functional networks consist of multiple spatially distinct brain regions, e. g. defined by a fixed anatomical atlas that form the nodes, and edges that are based on behavioural similarity between these regions. We recall that in the previous chapter we imposed a network on the Kuramoto model, which corresponds to what we now know as a structural network for the model. We then created the functional network from the time-series output of the model. The data for neuronal functional networks are usually obtained by imaging methods such as functional MRI (fMRI) or electrode-based methods monitoring N predetermined regions and producing time-series for these regions. One then defines an association matrix $\tilde{A} = (\tilde{a}_{ij})_{i,j=1,\dots,N}$ for the functional network using a measure for the statistical association between the time-series of node i and j (see Figure 3.3 for schematic overview

of the process). The measure for the association matrix is chosen depending on the underlying neuroscientific question. Even though structural and functional networks differ greatly in their definition, as in the Kuramoto case the structural network of a brain underlies the functional network and there is strong evidence that network topological parameters are conserved, many of which can undergo changes for example in neurological diseases [13, 32]. In most studies, the graphs analysed after processing

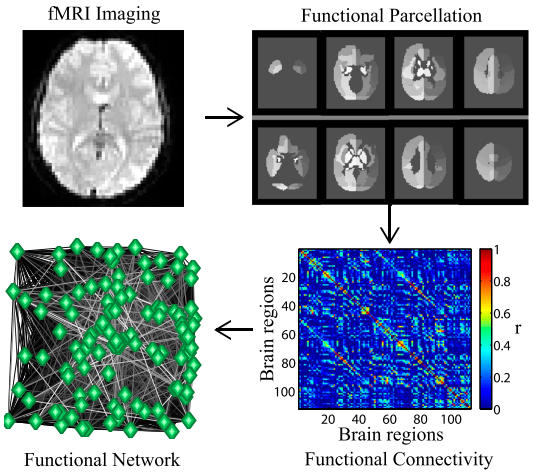

Figure 3.3: Creating a functional network from fMRI data (with permission from [5]).

experimental data are binary. Frequently one applies a global threshold $\tau \in \mathbb{R}^+$ to the association matrix yielding a binary[3] adjacency matrix $A = (a_{ij})_{i,j=1,\dots,N}$ that represents the graph to be studied as follows:

$$a_{ij} = \begin{cases} 1, & \text{if } \tilde{a}_{ij} \geq \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

The choice of threshold has a major influence on the resulting graph and inevitably leads to a loss of information. Moreover, it has been suggested the chosen threshold determines the properties that can be observed [14]. The two approaches normally taken to address this problem are either to try and find one optimal threshold or to threshold the association matrix at different values and examine the resulting network properties as a function of the threshold [14, 33].

Using graph theory on neuronal networks has led to many interesting medical insights. For example, a series of neurological and psychiatric disorders appear to be accompanied by changes in network connectivity, and some network properties can even be used as diagnostic markers for conditions such as Alzheimers, strokes, autism,

[3] Alternatively, one can study a thresholded version of the weighted graph.

and schizophrenia. However, such results need to be treated with caution, because other factors can also influence network structure [13, 44].
The full potential of using network theory in neuroscience is yet to be tapped. Thus far, it has mainly been used for descriptive purposes of static networks, seldom taking into account temporal evolution or interactions with outside networks. These form promising areas of recent research and offer opportunities for future work [33].

3.2.2 Data set: Human brain networks during learning

The data set that we study in the present dissertation is a set of time-dependent functional brain networks that has been studied by several existing methods. The data was analysed in [5] to investigate the temporal correlation of neuronal activity during the acquisition of motor learning skills. Twenty subjects were monitored while performing a simple motor learning task similar to a musical sequence executed by four fingers using their non-dominant hand. In addition, the data from the daily sessions were cut into 25 non-overlapping time windows, each of the length of 80 data points. The data set therefore consists of 60 networks with 25 time layers each. The edge weights of the functional networks were calculated from the temporal correlation between the activity of every pair of nodes, which in this case are fixed in time and represent 112 different brain structures based on a neuroanatomical atlas. The association matrices for the the functional networks were corrected for a false discovery rate. i.e. matrix elements under a certain threshold which represent a correlation that is to be expected at random, were set to zero while the other matrix elements were retained.

The results indicated that there is a significant segregation of the nodes in the functional networks into a small number of different communities with dense high edge weight connections inside the communities and sparse high edge weight connections to other nodes. Within these communities certain nodes appeared to stay within the same community over the time of the experiment while others seemed to constantly switch between different communities. In [8] it was shown that two very important groups of nodes forming parts of such a community during the acquisition of the motor-learning task described above over a longer amount of time are the primary and secondary sensimotor regions and the primary visual cortex. With task practice, the strength of their interaction has been observed to decrease as they presumably become more autonomous. Moreover, [7] showed that these modules form part of a densely connected stiff temporal core surrounded by a periphery of nodes frequently

changing connections among each other. Thereby, a stronger segregation of the core and periphery was observed among subjects who showed better learning.

The data set was further used in [6] to illustrate the use of null models in robust community detection.

Chapter 4

Topological network analysis

As mentioned in Subsection 3.2.1, graphs are often studied after applying a threshold to the data. The choice of such a threshold is difficult since there is no guarantee that a subinterval of thresholds shows conserved network properties. In particular, many parameters such as graph size need to be taken into account when interpreting results on thresholded networks [44]. One of the major advantages of using persistent homology is that we can examine a graph filtration generated by all possible thresholds and analyse how persistent certain topological features are across the filtration. We now give a detailed explanation of the methods that we use for topological network analysis, and we then present results on the various data sets studied.

4.1 Methods and algorithms

As we saw in Section 2.2, the main ingredients to using persistent homology on network data are simplicial complexes and filtrations. There are many ways to define simplicial complexes and filtrations on graphs, and the choice is either motivated by the type of questions to be answered or by consideration of computational scaling. We use three different filtrations for our analysis: a filtration by weights, a weight rank clique filtration and a modified version of the Vietoris-Rips complex for weighted networks. All three filtrations have previously been applied to weighted neuronal networks [29, 34, 28], although we use a slightly different approach when transforming edge weights into distances in a metric space for the Vietoris-Rips complex since the idea to use this approach was independent of [28].

4.1.1 Filtration by weights

The simplest graph filtration is a filtration by weights [29], which we obtain via the following steps:

1. Define filtration step 0 as the set of all nodes.
2. Rank all edge weights $\{\omega_1, \ldots, \omega_\tau\}$, where $\omega_1 = \omega_{max}$ and $\omega_\tau = \omega_{min}$, where τ is the amount of distinct weights in the graph.
3. In filtration step t, threshold the graph at weight ω_t, for $1 \leq t \leq \tau$ and create a binary graph.

We then define the simplicial complexes on the graph: Every node defines a 0-simplex and every edge in the graph forms a 1-simplex. We include unconnected nodes as 0-simplices in the simplicial complex. We call this simplicial complex an *edge-node complex*. As mentioned in Subsection 3.2.1, we imagine the graph to be embedded in a higher-dimensional space to avoid edges crossing. Note that the simplicial complex on a graph is maximally of dimension 1.

4.1.2 Weight rank clique filtration

The weight rank clique filtration, first introduced by [34], is also based on a graph filtered by weights. However, it differs in the subsequent definition of the simplicial complex on the graph. Instead of just building 0- and 1-simplices, we build p-simplices from p-cliques. This is a valid simplicial complex because every $(p+1)$-clique in the graph guarantees the existence of a p-face on that clique, due to the fact that cliques are closed under intersection or taking subsets. Hence, they satisfy the requirements for a simplicial complex. This type of simplicial complex on a graph is called a *clique complex*.

4.1.3 Comparison of the two filtrations based on graph filtrations by weight

We now present an example to illustrate the advantages and disadvantages of using a filtration by weights and the weight rank clique filtration. We consider the graph filtration in Figure 4.1 and build an edge-node complex as well as a clique complex on it. Figure 4.2 shows the resulting barcodes. The 0-dimensional barcodes in both cases show us what we expect: Prior to the first filtration step, all points are separate connected components. In the first step, the graph is separated into two distinct

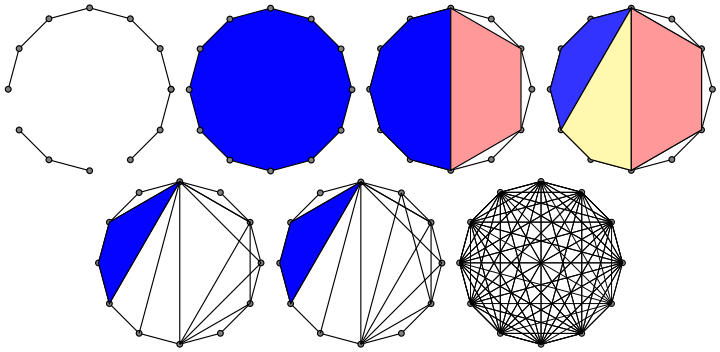

Figure 4.1: Dodecagon filtration: We show the three holes recognised by the weight rank clique filtration in colour.

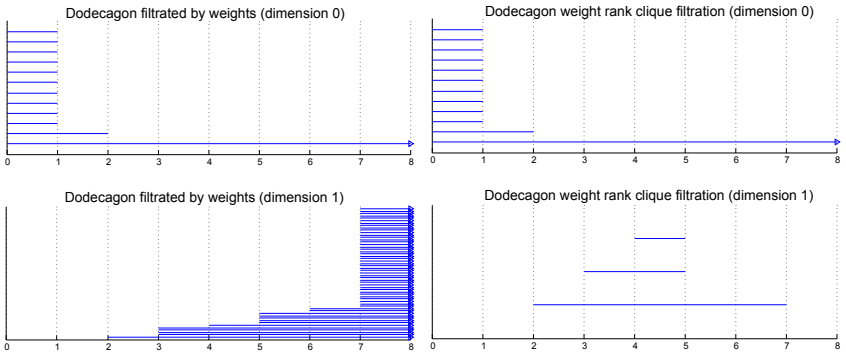

Figure 4.2: Barcodes for the dodecagon filtration using two different methods for building the simplicial complex on the graph filtration.

connected components before it becomes fully connected in the second filtration step. Although the barcodes for dimension 0 do not exhibit any difference between the two filtrations, the filtrations provide very different information for the 1-dimensional case. The filtration by weights counts every new connection closing a loop as a separate 1-loop in the network. These 1-loops do not die throughout the filtration because no higher-dimensional simplices are built on top of them. The 1-dimensional barcodes thus primarily point us towards filtration steps, where a large number of edges closing a new loop or crossing an existing loop is added. This can be observed in filtration step 5 and 7 of our example. The 1-dimensional barcode does however not reveal any information on how large these loops are or whether they are crossed by a new edge at a later filtration step. In the weight rank clique filtration a 1-loop needs to consist of at least 4 edges to appear in the 1-dimensional barcode and it subsequently dies as soon as it is completely filled with triangles of connected vertices forming 2-simplices. In our example, the blue loop created in filtration step 2 is crossed by weaker edges in filtration steps 3 and 4 giving rise to the pink and yellow loops respectively and it only disappears in the last filtration step when it is fully filled with 2-simplices. Persistent

1-loops in the weight rank clique filtration indicate that the loop in question consists of strong edges and the edge completing the last 2-simplex in the loop is much weaker in comparison. 1-loops that are born and die very early in this filtration can indicate the existence of very strongly connected node communities of the network, such as the 4 nodes surrounding the yellow and pink hole in Figure 4.1. Note, that since both of these filtrations are constructed on a graph filtered by weights, the persistence of loops is always relative to the edge weights, i.e. we obtain the exact same barcodes as above for a graph, which could for example have a very different spectrum of edge weights, if their ranking and distribution on the graph edges remains the same. Both filtrations reveal different aspects of the studied graph, which, when taken together, give an impression of the types of questions that can be answered by using these methods.

4.1.4 Modified Vietoris-Rips complex

The Vietoris-Rips complex, as described in Subsection 2.2.3, provides a good method for analysing point cloud data in a metric space. In order to use the complex on weighted network data, we define a map ϕ that assigns elements of the association matrix $\tilde{A} = (a_{ij})_{i,j=1}^{N}$ to elements in a distance matrix $\tilde{D} = (\tilde{d_{ij}})_{i,j=1}^{N}$, such that the entry $\tilde{d}_{ij} = \phi(\tilde{a}_{ij})$ represents the distance between nodes i and j embedded in a metric space. To detect communities with high edge weights between nodes within the community and much lower edge weights to nodes outside the communities, we define ϕ as follows:

$$\phi(a_{ij}) = \frac{1}{a_{ij}}, \tag{4.1}$$

for $i, j = 1, \ldots, N$. We then build the Vietoris-Rips complex on the resulting point cloud, varying the radius ϵ of the balls around every point in the metric space in 1000 steps from zero to a set maximum. In comparison to the other two filtrations the barcodes resulting from the Vietoris-Rips complex are therefore not relative to the edge weight spectrum.

4.1.5 Computational tools and issues

We analysed networks with MATLAB code constructed using javaPlex [42], a software package for persistent homology. For a given filtration of a simplicial complex, javaPlex outputs Betti intervals that include representative cycles and barcodes.

For the filtration by weights and the modified Vietoris-Rips complex these tools posed no problems. The weight rank clique filtration, however, exhibited several

major computational difficulties. We first attempted to write a MATLAB code using a maximal clique finding algorithm from the Mathworks library [45] based on the Bron-Kerbosch algorithm, which is to date the most efficient algorithm known for this problem. A maximal clique finding algorithm with a good running time with respect to the network size n has been shown to be impossible and as a worst case such an algorithm has a runtime of order $O(3^{\frac{n}{3}})$ [9]. For the weight rank clique filtration all cliques that are found in one filtration step need to be saved and compared with all later cliques because every reported clique can be added as a simplex only the first time it appears. For the given data, the code appeared to be very slow. One first solution was to restrict the dimension of the simplicial complex to a maximum of 4 and only consider barcodes to a maximal dimension of 2 — the estimated runtime for one network sample from the analysed networks is in the range of several weeks to months. We made another attempt by combining k-clique finding algorithms for $k \leq 4$. This, however, did not speed up the process presumably because more cliques had to be stored and compared in every filtration step.

Therefore, we further attempted to use the newly developed algorithm JHOLES in addition to the MATLAB codes. JHOLES was presented in [11] as an efficient java implementation of the weight rank clique filtration built on the JAVAPLEX library. For a given set of nodes and edges, JHOLES performs the weight rank clique filtration and outputs Betti intervals and, in the newest version released in August 2014, even barcodes. There are two disadvantages of JHOLES that we observed on computation. Although JHOLES is very fast and uses little RAM for small networks with a small number of cliques, it appears that there are major computational issues when it is used with almost complete networks such as our functional networks. The analysed networks needed more than 180 GB of RAM and ran very slowly (the estimated runtime for one data sample is several weeks). The large number of cliques appears to be causing these issues.

Several strategies were attempted to overcome these problems. The results which we present are based on the MATLAB codes. To reduce computational time, we consider a thresholded version of the weighted graphs, including at least half of the edges. We used JHOLES to check the MATLAB codes for small examples, which were consistent.

4.2 The Kuramoto model

4.2.1 Simulation

We simulated the basic Kuramoto model for all-to-all coupled oscillators using the Runge-Kutta MATLAB solver ODE45. We checked the code using an existing code based on explicit Euler by producing a plot of the mean coherence between the oscillators as a function of the coupling strength. [1] We carried out a total of 10 simulations using the same initial conditions and we averaged the coherence value over these. The model exhibits the sigmoidal dependency that is known to occur. The two codes fully agree for small time steps as shown in figure 4.3.

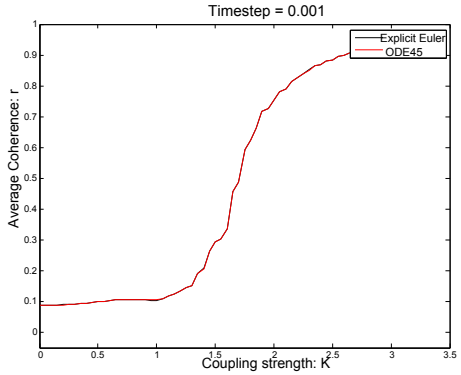

Figure 4.3: Mean oscillator coherence between all-coupled Kuramoto oscillators versus the coupling strength. We tested the code based on ODE45 using an existing explicit Euler code for the same model.

We then constructed the community coupled version of the Kuramoto model as described in section 3.1.2. We set the maximal time to $T = 10$ and we used $\Delta t = 0.02$ for the time step. We carried out a total of 20 simulations, which yielded one functional network each using the same coupling matrix $A = (A_{ij})_{i,j=1}^{N}$. We varied the initial conditions for the oscillators as well as the natural frequencies between simulations. We averaged the resulting functional networks over the 20 realisations and we varied the number of time-series elements that we used for one time layer to gain insights into the synchronisation process at different time scales.

We also simulated a second version of the community-coupled Kuramoto model, where we fixed the natural frequencies and the coupling matrix for all 20 simulations. In both cases, we created the null models used simultaneously to the Kuramoto output (see Section 3.1.3). We set up the Fourier null model in the same way as in [24].

[1] During these investigations we found a minor bug in the reference code. This implementation was based on the code [30], which included the same bug and was published online together with [41].

4.2.2 Filtration by weights

We first discuss the results based on simulations using different natural frequencies for each simulation. In Figure 4.4 we show functional networks based on the coherence measured according to Equation (3.5) over all time steps – this corresponds to one time layer – represented in matrix form. We created the underlying community structure using the coupling matrix shown in Figure 3.1.

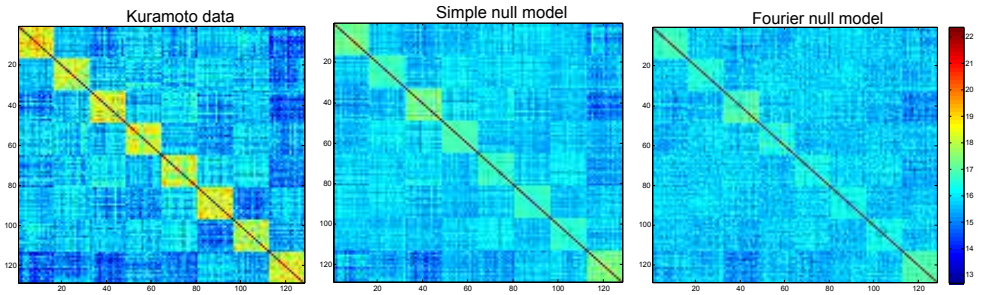

Figure 4.4: Functional networks generated from the Kuramoto model, the simple null model, and the Fourier null model over all time steps.

Although both null models also detect stronger edge weights within the communities than between them, the distinction appears to be weaker in comparison to the Kuramoto model output. Moreover, the maximal edge weights in the null models appear to be smaller than in the Kuramoto data. This reflects in the length of the filtration by weights as well as in the resulting barcodes (see Figure 4.5). Although

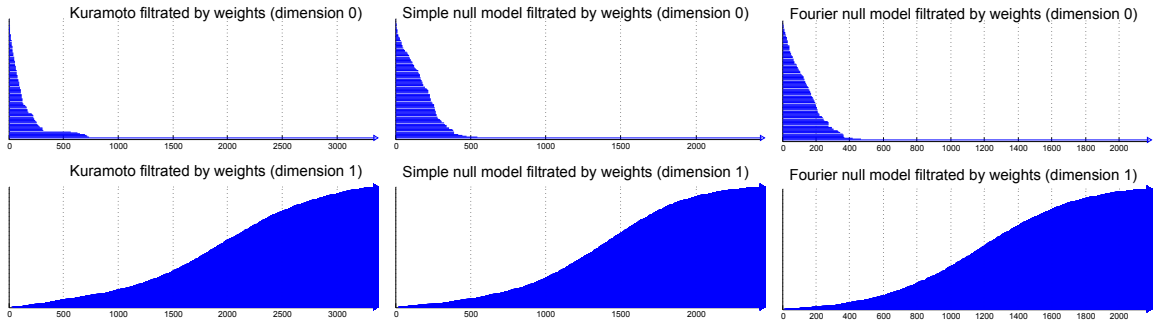

Figure 4.5: Filtration by weights barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. The functional networks were based on all time steps.

the Kuramoto data includes about 3400 filtration steps, both null models have fever than 2500 filtration steps.

Recall for the 0-dimensional barcodes that every node of a network is regarded as an individual connected component prior to the first filtration step. A component's life within the barcode ends when it is joined to another component by an edge. We

cannot distinguish whether a connected component consists of only a node or a whole set of connected nodes by looking at the barcode only. javaPlex outputs the Betti intervals combined with representative cycles, so it is possible to trace the cause of death of an individual component to some extent. In the 0-dimensional case, these cycles consist of nodes potentially representing an entire component of connected nodes and we therefore need to go back to the actual filtration step in the graph filtration to detect whether the death of a component is caused by a node joining another node or a larger component, or whether two larger components are joined.

In the case of the Kuramoto data the very long lived components in the 0-barcode in Figure 4.5 with a persistence of over 500 filtration steps can be traced back to the joining together of the 8 separate communities, which were implanted in the structural network. This indicates that there are about 500 different, very strong edge weights, which connect nodes within the communities before the communities are joint together in one component later in the filtration. In the two null models the long living components often consist of nodes joining the network at a later time, which reflects the less distinct communities.

The 0-barcode based on Kuramoto data has a different shape compared to the two null models. In the first filtration steps connected components disappear very rapidly, forming a very sharp peak in the barcode with about two thirds of the connected components dying within the first 150 filtration steps. The 0-barcodes generated from the two null models appear to have more persistent components from the beginning onwards forming a wider cone in the upper two thirds of the shown components even after taking into account that the filtration lengths differ. Together with the observation that individual nodes tend to join the network later, this could indicate that more connections are made within the communities early. The communities are then joined together earlier than in the Kuramoto case, where most nodes connect to their communities early in the filtration, but the communities are connected later.

After the rapidly dying components, the Kuramoto data exhibits a regime of linearly longer lived components followed by the last eight very long lived components. We observe a similar phenomenon for the null models.

The 1-dimensional barcodes look very similar in all three cases and exhibit a very slow increase of 1-loops in the first third of the filtration steps. This indicates that in this regime nodes are joined sparsely to each other without forming many loops. Note that this corresponds to the filtration steps during which all separate components in the network are connected. We observe that in the filtration step, in which the network finally consists of one single component, a larger proportion

of 1-loops has formed in the Kuramoto model output than in the two null models. This presumably again reflects the fact that in-community edges are stronger for the Kuramoto communities, and thus form more 1-loops early on, than in the null models.

After this first regime the 1-loops increase more rapidly, presumably when more connections within the communities are made. They then almost saturate as more connections between the communities are formed in the end of the filtration.

The form of the 0- and the 1-dimensional barcodes are determined by the underlying community structure as well as the behaviour of the model. Random networks of the same size constructed by drawing edge weights from a uniform distribution are longer in filtration length and also have fewer long-lived components in the 0-barcode. They exhibit a strictly linear increase in 1-loops (see Figure B.1 in Appendix B).

We now examine different time scales in the Kuramoto dynamics by creating two time layers of the functional network from the model output. One is based on the first half of the time steps and the other is based on the second half of the time steps. In Figure 4.6 we show the two matrix representations of the functional networks that we create in this way. During the first 250 time steps, we observe a strong synchro-

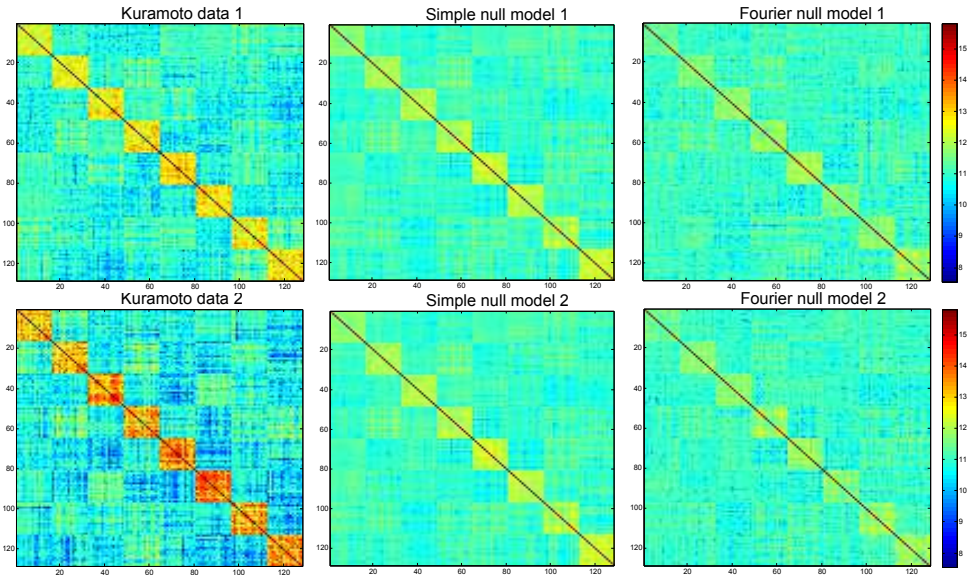

Figure 4.6: Functional networks generated from the Kuramoto model, the simple null model, and the Fourier null model. Top row: Networks based on time steps 1–250. Bottom row: Networks based on time steps 251–500.

nisation within the communities in the Kuramoto data. The synchrony gets even stronger in the second half of the time steps and some communities exhibit stronger inter-community synchronisation than others. Both null models exhibit a weaker

in-community synchrony than the Kuramoto case, but they appear to exhibit less variation in the synchronising patterns among the communities leading to a higher overall synchronisation of the oscillators. The corresponding barcodes are shown in Figure 4.7.

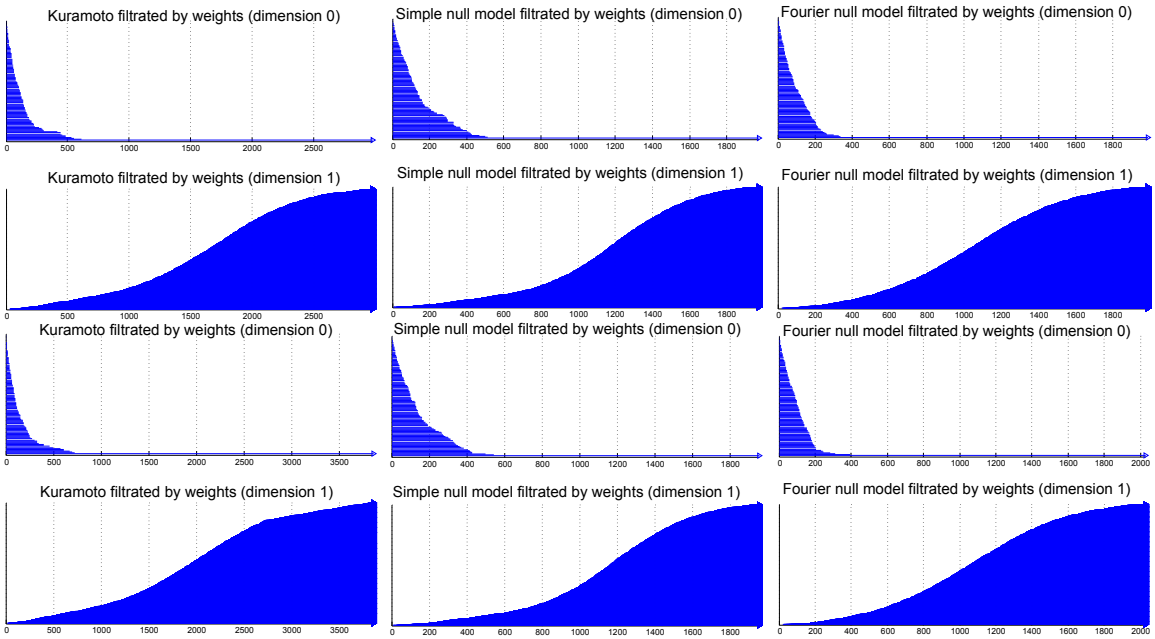

Figure 4.7: Filtration by weights barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. Top row: Barcodes generated from functional matrices based on time steps 1–250. Bottom row: Barcodes generated from functional matrices based on time steps 251–500.

The shape of the 0-barcodes for both time ranges differs slightly in the Kuramoto case with more components dying prior to filtration step 500 in the first time range. These components only consist of merging communities in some cases, mostly they represent nodes joining the network at a later filtration step. For the barcode generated from the second time range the death of the longest living components in about half the cases represent the joining of the eight communities, which can be expected from the functional matrix.

Both 0-barcodes generated for the null models have similar traits in both time ranges, as can be expected. The longest living components in all four barcodes represent nodes connecting to the rest of the network late in the filtration. This can be expected given the similarly strong connections in and outside the communities.

The shape of the 1-barcodes hardly differs from the previously shown barcodes in Figure 4.5. Only the barcode based on the second time range of the Kuramoto time-series seems to differ slightly from the previous shape from filtration step 2700

onwards. This could be caused by the fact that instead of just closing connections within existing components we also have a large proportion of new nodes entering the network subsequently closing 1-loops. Indeed, looking at the functional matrices in Figure 4.6 we observe that there are a few nodes with relatively low edge weights within their communities compared to the first time range.

We consider a final example, which we create from the Kuramoto model using a fixed set of natural frequencies over 20 simulations. In Figure 4.8 we show the matrix representation of a functional network created from 20 community-coupled Kuramoto simulations using the same underlying coupling matrix and natural frequencies in every simulation.

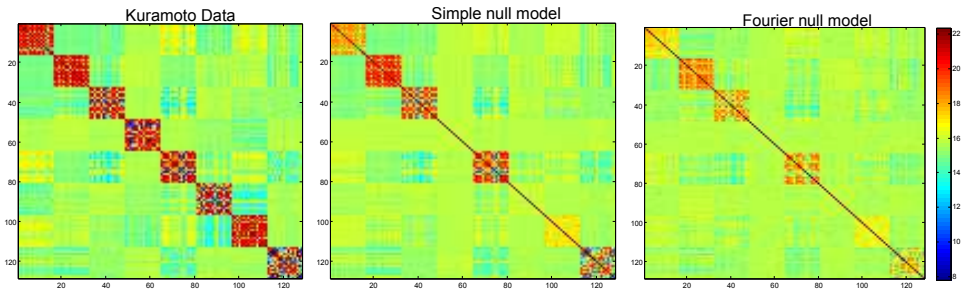

Figure 4.8: Functional networks generated from the Kuramoto model, the simple null model, and the Fourier null model over all time steps using fixed natural frequencies.

Observe that in the Kuramoto case the communities are generally more strongly internally connected than among each other, but some internal connections are also very weak. Both null models appear to have less communities than the Kuramoto case and in some cases also weaker internal connections. This has an influence on the corresponding barcodes, which we show in Figure 4.9.

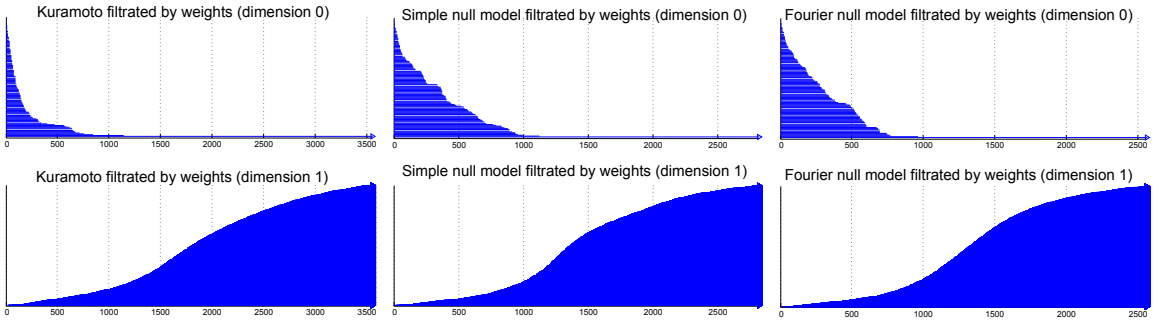

Figure 4.9: Filtration by weights barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. The functional networks are based on all time steps and the simulations used fixed natural frequencies.

The 0-barcodes for the null models now appear to exhibit a much thicker peak caused by more persisting components in the beginning of the filtration. The overall persistence of the components also seems to be longer for the null models than for the Kuramoto case. While all three 0-barcodes exhibit long lived components of persistence 500-1000 filtration steps, there are only about half as many long lived components in the Kuramoto case. Moreover, the Kuramoto oscillators exhibit a phase between filtration step 250 to 600, where edges appear to connect nodes within already existing components rather than joining separate components together. Even though the increase in the number of 1-loops in the corresponding filtration phase of the 1-barcodes are of similar magnitude for all three cases, the connections made in the Kuramoto case during this phase are presumably within communities while most in-community connections in the null models have at this stage presumably already been made due to the smaller amount of strongly connected communities.

Further examples of filtration by weights barcodes for the Kuramoto output can be found in Appendix B.2.

It generally appears difficult to deduce information from the 1-barcodes in this filtration, even though they carry important information when compared to the 0-barcodes of the filtration. In Subsection 4.2.3, we present the results from the weight rank clique filtration, which carries more information in the 1-barcodes.

4.2.3 Weight rank clique filtration

As we discussed in Subsection 4.1.5, the weight rank clique filtration is computationally very demanding. We therefore produced barcodes for a small set of examples and considered only a little more than half of the filtration steps. We run the codes on the Kuramoto data, the simple null model, and the Fourier null model. We used the functional matrices with two time layers introduced in Figure 4.6. In Figure 4.10 we show the corresponding 1- and 2-barcodes. We omitted the 0-barcodes since they are the same as for the filtration by weights (see Figure 4.7).

Both barcodes generated from Kuramoto data appear to have short lived 1-loops in the beginning of the filtration. More persistent 1-loops appear later in the filtration. In the functional network created from the early time steps most of the 1-loops up to filtration step 900 are short lived. In particular the ones appearing before filtration step 300. Recalling the 0-barcodes in Figure 4.7, this is also the point in the filtration when the remaining separate components in the network become more persistent. As before, we can look for representative cycles in the javaPlex output. In this case this information needs to be treated with caution since the output can consist of a

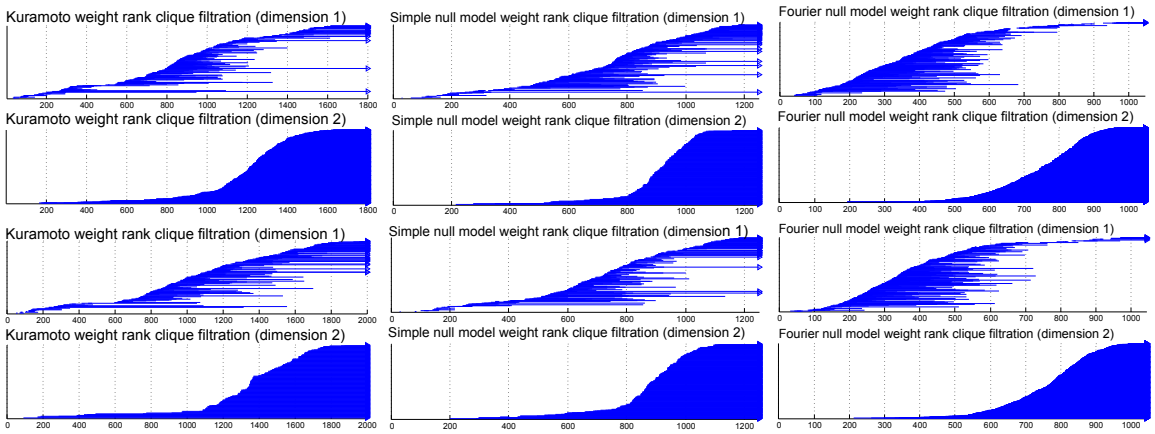

Figure 4.10: Weight rank clique filtration barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. Top row: Barcodes generated from functional matrices based on time steps 1–250. Bottom row: Barcodes generated from functional matrices based on time steps 251–500.

1-loop, which does measure the same hole as the one we are interested in, but that could appear in the network in a much later filtration step than the first appearance of the hole in the network. Hence, this could for example give the impression that the loop is born by an inter community connection occurring later in the filtration when in fact it is created as an in-community loop at first. Around filtration step 900 many of the representative loops observed are in communities, which is what we expect. None of these loops appear to be very persistent, which indicates that they are filled quickly with 2-simplices, i.e. the nodes in the community are densely connected by high edge weights. This is also the point in the filtration at which we observed a rapid increase of 1-loops in the filtration by weight (see Figure 4.7).

The functional matrix for the second time range of the Kuramoto data appears to have even more short-lived 1-loops in the beginning of the filtration than the previous time range, which is presumably caused by the even more synchronised communities in the functional network. The early 2-loops are also represented by in-community connections supporting this claim. Moreover, we recall that the merging connected components at the end of this phase in the filtration consisted of communities being connected. We also observe more late-formed persistent loops than in the previous time range. Because the later parts of the filtration are not visible, we cannot discuss precisely how persistent these loops are. We can only speculate that these more persistent 1-loops could be inter community connections.

Both null models also exhibit quickly dying cycles in the beginning of the filtration, although in the case of the Fourier null model this phase is much shorter than in the two other examples. These are followed by more very persistent 1-loops than we

observed in the Kuramoto case. The 2-loops up to about filtration step 700 are in both null models very often represented by in-community loops, which could indicate that the short-lived 1-loops in the beginning of the filtration are also mostly in-community loops. The barcodes for the later time range for both null models is very similar to the first. We recall that the functional networks for both null models showed less in-community synchronisation compared to the overall synchronisation than the Kuramoto output (see Figure 4.6). This was in particular the case for the Fourier null model. Since all shown barcodes exhibit very short-lived 1-loops in the beginning of the filtration, we will thus treat this as a sign of densely connected communities, in particular if this is supported by the early forming 2-loops. Even though the functional networks for the null models do not seem to change significantly between the time ranges, the barcodes differ in the appearance of long-lived 1-loops and the form of the 2-barcodes, indicating that these properties are subject to variability and should only be used with caution when characterising the data networks we analyse later on. Note that both short-lived 1-loops in the beginning of the filtration followed by persistent 1-loops do not occur in a uniformly at random generated network (see Figure B.1 in the Appendix).

4.2.4 Modified Vietoris-Rips complex

We now consider the same set of functional networks (see Figure 4.6) using the modified Vietoris-Rips complex. We show the barcodes for the Kuramoto data, the simple null model, and the Fourier null model in Figure 4.11.

Recall that nodes connected by strong edge weights in the network are placed in proximity of each other in the metric space while weakly connected nodes are further apart. We thus expect for nodes to connect to their communities much earlier than to other communities. For all three cases the 0-barcodes show that the separate connected components merge within a small interval of ϵ values. Only the Kuramoto data exhibits a group of longer lived components. For both time ranges most of these components die due to communities that are merging: seven respectively six out of nine long lived components die due to inter-community connections. The null models do not share this property. This reflects expect on the basis of the differences between the functional networks (see Figure 4.6), where the communities in the Kuramoto data are stronger connected than in the null models.

In the 1-barcodes, we also see very clear differences between the Kuramoto case and the two null models. In the Kuramoto case, there is a clear gap in the 1-loops which appears as the nodes merge with their distinct communities. 1-loops only start

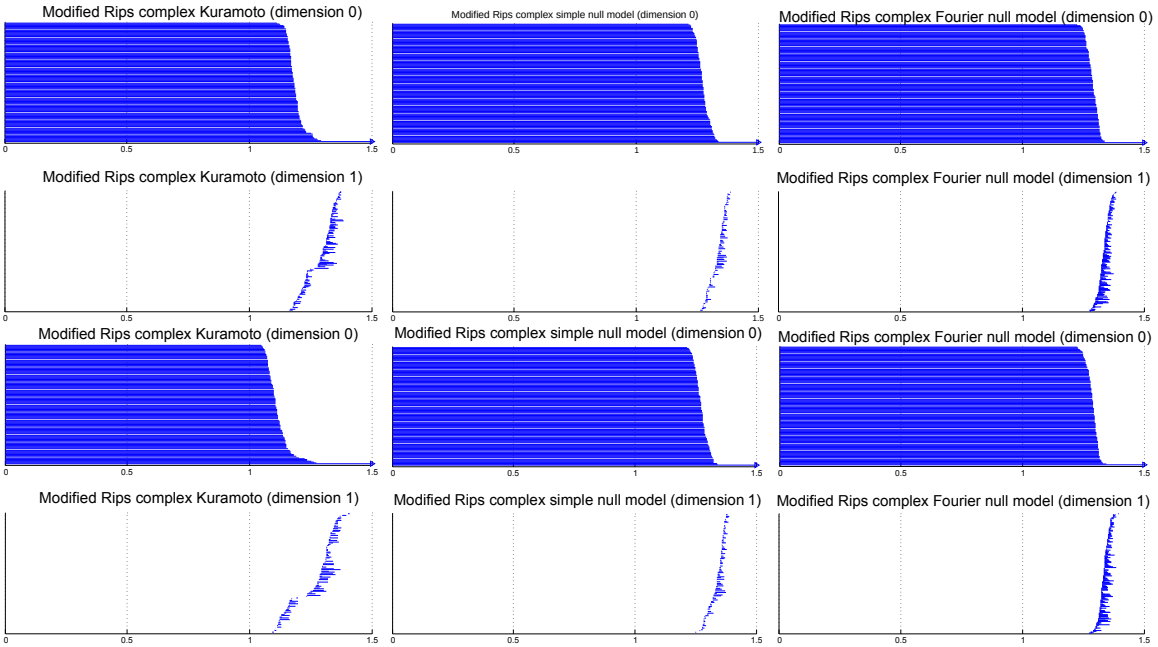

Figure 4.11: Vietoris-Rips barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. Top row: Barcodes generated from functional matrices based on time steps 1–250. Bottom row: Barcodes generated from functional matrices based on time steps 251–500.

forming again, once the communities start to join. The gap is more prominent in the second time layer which reflects the stronger connections within the communities in the second time regime. The barcodes therefore not only point us towards the joining communities, but also indicate how strong the edge weights within the communities are. Moreover, the 1-loops formed between communities appear to be more persistent in the filtration, which is due to the reciprocal relationship we introduced between the distance in the metric space and the strength of the edge weights.

Both null models show a less clear separation in the 1-barcode, again indicating that the modified Vietoris-Rips complex is sensitive to the strength of edge weights in the communities as well as the absolute difference between the edge strengths in and outside of the communities (see Figure B.6 in the Appendix for barcodes of more pronounced communities in the null models as observed in the functional networks in Figure 4.4).

As a final example, we show the modified Vietoris-Rips complex barcodes for the functional network we introduced in Figure 4.8, which for the null models exhibited less communities than for the Kuramoto data, in Figure 4.12.

For the Kuramoto case, we again see a strong separation between the 1-loops formed in communities and those formed between communities. This is not the case for the null models and only the Fourier null model shows a prominent gap in the

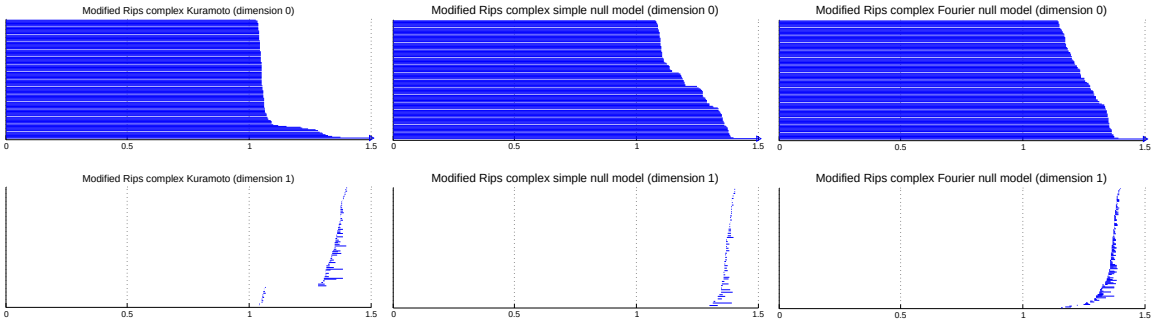

Figure 4.12: Vietoris-Rips barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. The functional networks were based on all time steps.

1-barcode. This is surprising because the communities in the Fourier null model are overall connected by weaker edge weights than it is the case in the simple null model. The 0-barcodes also differ greatly in shape and only the Kuramoto case shows clear signs of late merging communities. This indicates that while the modified Vietoris-Rips complex is a good indicator of very strong in-community connections combined with weaker inter-community connections, it might not point us to a small amount of communities with a smaller difference in edge weights to the connections between communities.

For modified Vietoris-Rips complex barcodes of a network with edge weights drawn uniformly at random, see Figure B.1 in the Appendix.

4.3 Functional imaging data

We recall that the data was collected from 20 subjects during a time period of 3 days. We analyse several versions of the data. The first version that we study consists of one matrix per subject per day. A second version of the data is further divided into 25 time windows for every day, which resultis in 1500 functional matrices in total. In the third version of the data, these 1500 matrices are also corrected for false detection. We present barcodes of version one of the data only and consider the other two versions as references for the filtration by weights only. We observe whether we can detect any of the features we observe in version one are characteristic for all three versions.

We show three functional matrices from the data in Figure 4.13. Generally, high edge weights seem to be underrepresented in comparison to weaker edge weights. Some of the edge weights seem to get perceivably stronger over the course of the three days.

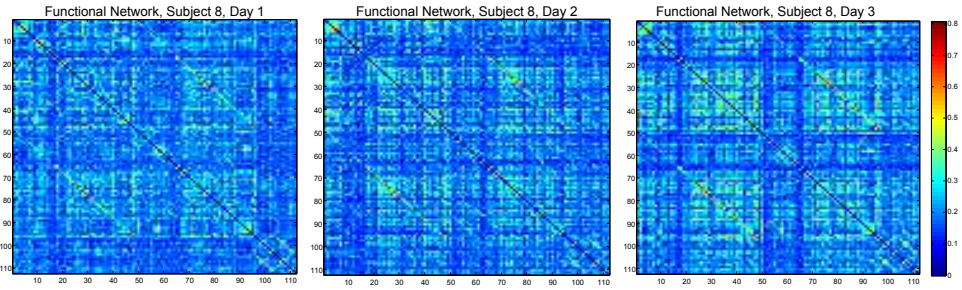

Figure 4.13: Example of functional matrices, days 1–3 for subject 8. The matrices are based on version one of the data.

4.3.1 Filtration by weights

In Figure 4.14 we show example barcodes that we created by using a filtration by weights on the data set from subject 8.

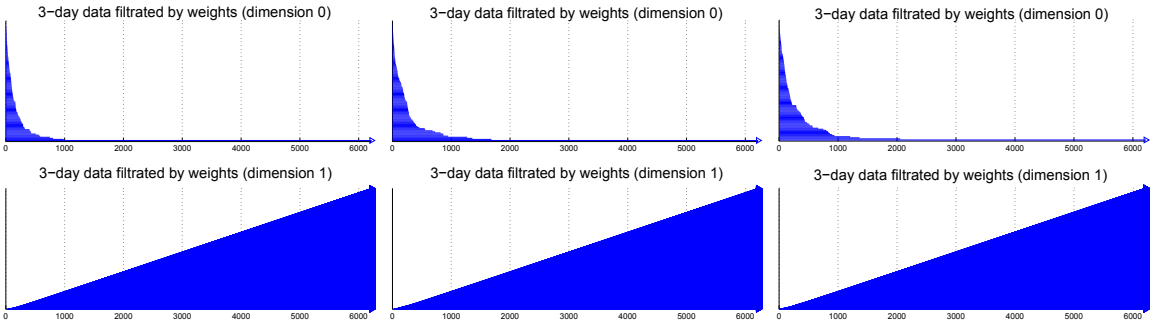

Figure 4.14: Example of filtration by weights barcodes generated from data from subject 8. Every barcode represents one of the three days of the experiment shown in temporal order. Top row: 0-dimensional barcodes. Bottom row: 1-dimensional barcodes.

We consider the 1-dimensional barcodes first. Throughout the filtration there seems to be no particular phase of filtration steps, where a very large amount of 1-loops are born. We can explain this with the fact that in the given network two edges hardly ever carry the same edges weight and thus the filtration in most cases consisted of one filtration step per edge enabling at most two 1-loops to be generated in one step. The 1-barcodes for this filtration in all versions we analysed and across all subjects and days do not exhibit any further noteworthy or changing features and we thus concentrate on the 0-dimensional barcodes. The 0-barcodes in Figure 4.14 look slightly different for each of the three days, but there seems to be an underlying distinctive shape of the barcodes. For all three days components die rapidly in the beginning and persist for under 150 filtration steps. After about two thirds of the nodes are connected, individual components begin to persist for 300 filtration steps or more. For day three there even seems to be a group of longer-lived components

that die within 100 filtration steps of each other. The representative cycles in the code output however do not point us towards any specific long-living communities. In general, the persistence of the connected components appears to increase for the studied subject. This was however not a tendency that we observed across all subjects, and we also did not observe any apparent connection of this phenomenon to the task performance of an individual subject. The last particularly long-lived components on all three days consist of nodes joining the network at a late filtration step.

We observe the shape, which we describe for the 0-barcode of day three in Figure 4.14, as an underlying barcode form across most subjects. In versions two and three of the data this form even appears at least once for every subject. Apart from this underlying form we observe no general tendency for changes from day 1 to day 3 over all the 20 subjects.

We show further examples of filtration by weight barcodes for other subjects in the Appendix B.3.1.

4.3.2 Weight rank clique filtration

Due to the computational issues with the weight rank clique filtration, we only analyse thresholded versions of the functional networks. We mostly used around 2600 filtration steps, i.e. a little more than half of the edge weights. For five subjects we were able to carry out the analysis up the filtration step 4000 for one or two of the days. Even though the 0-barcodes do not differ from the 0-barcodes we created from the filtration by weights, we include the plots when we want to refer to points in the filtration when all or most components are connected. We present weight rank clique filtration barcodes based on data from subject 1 in Figure 4.15. All three 1-

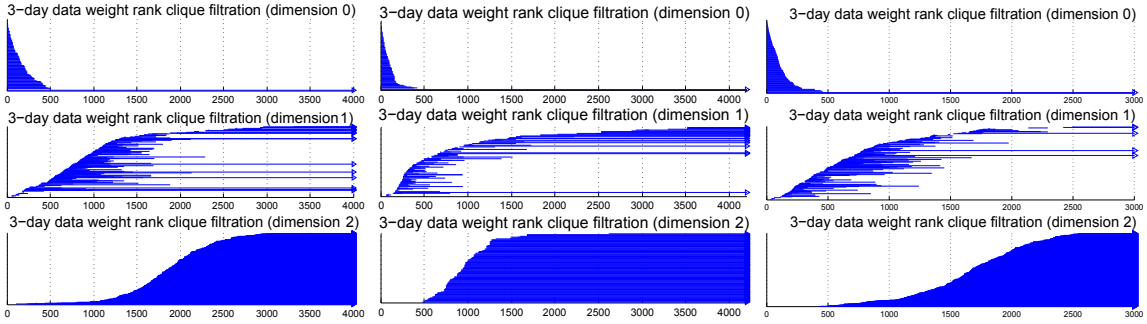

Figure 4.15: Weight rank clique filtration barcodes generated from thresholded data from subject 1. Every barcode represents one of the three days of the experiment shown in temporal order.

barcodes exhibit very short-lived 1-loops in the beginning of the filtration. Recalling the Kuramoto findings, these could potentially be indicators of communities of nodes

connected densely by high edge weights. We further notice that these 1-loops form at a point in the filtration at which only a small part of the connected components are already joint together. The 2-loops generally do not begin to form until almost all components are connected. This was not the case in the Kuramoto barcodes. Thus, even if the short-lived 1-loops are indicators of communities, the nodes within the communities are not quite as densely interconnected by high edge weights as they were in the Kuramoto case. We observe both these features across almost all subjects and days with very few exceptions. However, we do not observe any clear tendency of these short-lived 1-loops to become less or more over the course of the three days and we also do not see any connection to task performance of the subjects.

Although the 1-barcodes for subject 1 could suggest that more persistent 1-loops tend to appear later in the filtration over the course of the three days, this is not a common tendency across the subjects. We, however, observe that there is often a noticeable horizontal gap between early persistent 1-loops and persistent 1-loops that appear later in the filtration. Again, this does not occur for all subjects or for any particular day of the experiment. We show further examples of weight rank clique filtration barcodes for the data in Figure 4.16. We present these barcodes

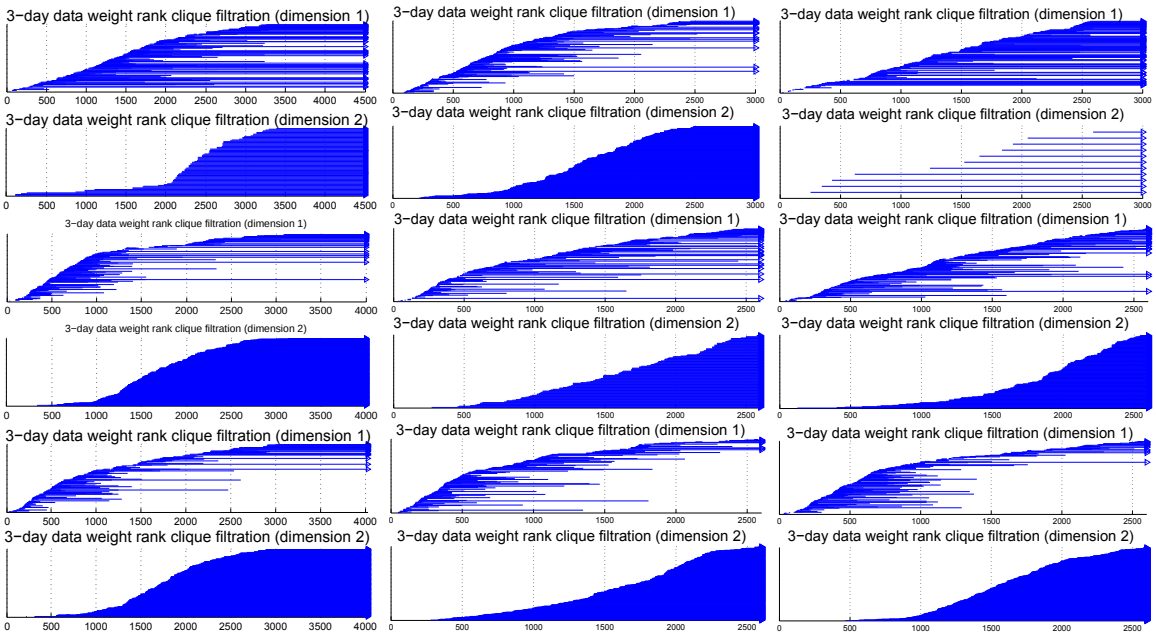

Figure 4.16: Weight rank clique filtration barcodes generated from thresholded data from subjects 5 (row 1), 9 (row 2) and 13 (row 3). Every barcode represents one of the three days of the experiment shown in temporal order.

mainly as an illustration of how different the barcodes are for different subjects. This difference is particularly striking in the 2-barcodes, which do not even seem to show

a similar amount of 2-loops across the subjects or the days. The 1-barcodes show a similar form and, as described above, short lived 1-loops in early filtration steps, but no common features among the more persistent 1-loops. A gap is visible between persistent 1-loops occurring for subject 9 on day 1 and 3 and subject 5 on day 2.

4.3.3 Modified Vietoris-Rips complex

We show two examples of modified Vietoris-Rips complex barcodes in Figure 4.17. Similar to the Kuramoto examples, we observe a small gap forming in the 1-barcodes,

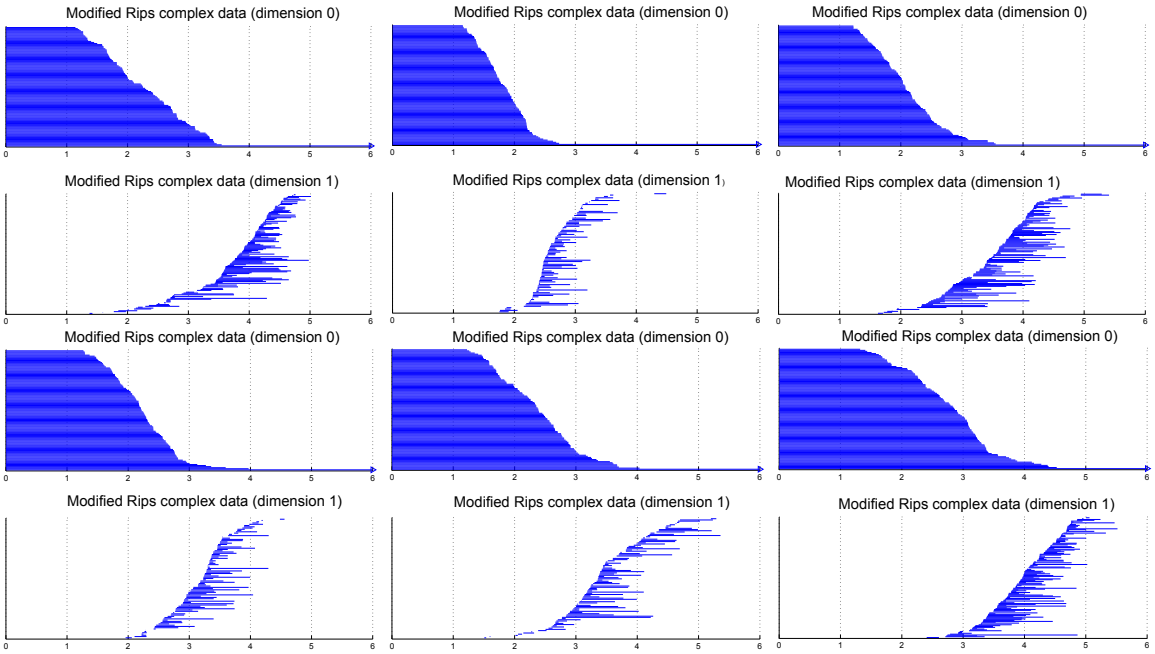

Figure 4.17: Modified Vietoris-Rips complex barcodes generated from data from subjects 1 (row 1),and 3 (row 2). Every barcode represents one of the three days of the experiment shown in temporal order.

although it is not quite as prominent as before. These first 1-loops are formed early on in the filtration when about half of the separate components are yet to be connected. We find them across most subjects on at least one day of the experiment, although the amount of short-lived 1-loops does not seem to be the same, and they are not always similarly prominent across subjects. We assume that these 1-loops indicate either the existence of a small number of small communities, in which nodes are densely connected by high edge weights or a small number of larger communities, in which nodes are more sparsely connected by high edge weights, since we would expect a more clear separation of early short-lived 1-loops from a large number of communities, in which nodes are densely connected by high edge weights.

Chapter 5

Discussion

The analysed Kuramoto examples suggest that all three methods which we employed can detect the imposed communities to some extent. In the 0-barcodes which we obtained from the filtration by weights, we observed that very rapidly merging network components followed by later very long-lived components can be a sign of distinct communities in the network. This barcode shape can, however, also be a sign of a network exhibiting a large amount of high edge weights followed by long-lived individual nodes, which are connected to the network by weaker edges. In the cases where we identified long-lived separate components in the 0-barcode as separate communities being joined late in the filtration, we needed prior knowledge on the nodes belonging to one community. We then used this knowledge to check the representative nodes given by the output of the MATLAB codes and to further trace these nodes in the respective filtration step of the network filtration. We observed a large variation of the shapes of the 0-barcodes for the null models, which usually exhibited less distinct communities than the Kuramoto data. The 1-barcodes which we analysed for the filtration by weights did not appear to give us any important information.

The results from the weight rank clique filtration seemed to detect the communities in the Kuramoto examples. We observed short-lived 1-loops in a phase of the filtration when the communities were not yet interconnected according to what we found in 0-barcodes. Indeed, there seemed to be a larger number of these short-lived 1-loops in the barcodes for cases when the nodes were more synchronised within their community. Again, we needed prior knowledge of the communities to deduce this. In addition, the representative cycles given by the MATLAB outputs did not necessarily give us the information we were interested in, i.e. the 1-cycles that cause specific 1-loops to be born can not necessarily be expected to be listed as representative cycles by the output of the code. Moreover, we were not able to perform the weight rank

clique filtration on the full network due to major computational issues, so we could not observe potentially important features in later filtration steps.

For the Kuramoto examples, the modified Vietoris-Rips complex showed the most promising results. For the Kuramoto data we found a clear divide into a phase of strong connections that are formed between in-community nodes leading to short-lived 1-loops in the barcode, and into a second phase where more persistent 1-loops are formed by weaker inter-community connections. We found the gap between these phases more pronounced for stronger in-community synchrony. We observed a weaker divide into these two phases for the null models. We interpreted this as a sign that the method not only detects communities, but also gives an impression of how strong the in-community edges are.

For the neuronal imaging data it was difficult to find clear observable tendencies across subjects or days, although the shapes of the barcodes did have some underlying similar traits. For the weight rank clique filtration and the modified Vietoris-Rips complex the barcodes often exhibited features, which we had connected to network communities in the Kuramoto examples. These consisted of short-lived 1-loops forming very early in the filtration in a phase during which the networks consist of separate connected components. This was the case in both filtrations. We further observed visible gaps between these short-lived 1-loops and the later 1-loops in the barcodes for the modified Vietoris-Rips complex.

As mentioned in Section 3.2, a similar data set was analysed by [8], focussing on changes in the architecture of functional connectivity patterns during learning. The brain regions studied were the same as in the data set that we analyse here and it was observed that two brain regions were consistently forming a common network community during the motor-learning task: the sensimotor regions and the primary visual cortex. We tried to trace members of both of these components in the representative cycles given by the MATLAB output (see Appendix B.4 for the sets of nodes). For the weight rank clique filtration we found that the early short-lived 1-loops were predominantly represented by cycles of nodes from these two areas. Often, there even seemed to be a clear separation into loops that were formed by edges between nodes from the visual component and loops that were formed within the sensimotor component. However, we need to keep in mind that the representative cycles are only homologous to the cycles which form these loops in the first instance. For the modified Vietoris-Rips complex the short-lived 1-loops also often consisted of connected members of the motor visual components, but there were less tendencies

towards a separation between the two components. We observed no further tendencies across days in the representative cycles of the short-lived 1-loops. None of the other properties observed in the same or a similar data set [5, 6, 7, 8] seemed to feature in the barcodes and we also did not appear to observe any barcode features which we could have associated with task performance of the individual subjects.

Over all, from the above observations the filtration by weights did not appear to be very useful. This was mostly due to the fact that only the 0-barcodes seemed to carry information. In general, we did not observe clear relevant features in the 0-barcodes alone for any of the methods. In particular in cases, where no prior knowledge of the network was given, we were not able to draw any conclusions from these barcodes. Previous studies using the filtration by weights, such as [29], had used the 0-barcodes as one of several approaches to compare two different data sets focussing on the differences in the shapes of the barcodes. They did, however, not attempt to find any underlying network features solely by observing these barcodes, but based their conclusions on a combination of methods they used. Similarly, [28] compared 0-barcodes for different data sets using a version of the modified Vietoris-Rips complex. This supports our observation that these barcodes are not informative on their own. They were, however, very helpful when used in combination with 1-barcodes for the weight rank clique filtration and the modified Vietoris-Rips complex. Moreover, if one is merely interested in observing connected components in a network, methods based on eigenvalue spectrums, such as in [3], might be more advisable in general.

One of the major problems encountered during the topological data analysis was the computational scaling. In particular the weight rank clique filtration posed major problems, even after limiting the analysis to the 0-, 1- and 2-homology groups and storing only up to 4-cliques. The algorithm JHOLES, which we tried to apply, did also not overcome these problems. An approach to overcome the issues for this filtration in future could be to use minimal simplicial sets that capture the topology of a simplicial complex [46]. The advantage of such an approach is that it omits constructing the clique complex, which caused the bad computational scaling. Another line of action could be to analyse separate filtration windows. This could however result in a large loss of important information.

Chapter 6

Conclusions and future work

In the present dissertation, we studied two sets of functional networks using tools from computational topology. The first set consisted of data generated from the Kuramoto model, the second set consisted of neuronal imaging data, which was based on experiments investigating the acquisition of a motor learning task [5]. We used three different methods from persistent homology to analyse these data sets: a filtration by weights [29], a weight rank clique filtration [34] and a version of a modified Vietoris-Rips complex [28]. The data generated from the model served mainly to shape our intuition for the interpretation of the barcodes. For the neuronal imaging data set our aim was to see whether we could detect any previously found properties of these networks [5, 6] or similar data sets [7, 8].

We found that for the weight rank clique filtration and the modified Vietoris-Rips complex the 1-barcodes show very short-lived 1-loops in the beginning of the filtration, which we also observed in Kuramoto examples with a strong underlying segregation of the network into functional communities. Further, these 1-loops were often represented by cycles of nodes from the motor-visual component, which was found to play an important role in the acquisition of motor learning tasks in [8]. Since this analysis is based on representative cycles only, further computational methods would need to be developed to identify the 1-cycles which form these 1-loops when they first become visible in the barcode. We also observed major computational issues for the weight rank clique filtration, even when using the newly developed algorithm JHoles. These computational obstacles still need to be overcome to enable future use of the method on densely connected networks such as the data sets studied.

For the topological methods, a better framework for the analysis of barcodes needs to be developed in order to study networks without prior knowledge of their features. This could be done by further analysing outputs from different set-ups for the Kuramoto model. For example, the model could be modified by using different coupling

strengths between the oscillators, adding internal or external Gaussian noise or an inertial term as it has been done by [23] in the all-to-all coupled oscillator case. This changes the dynamics of the model and could therefore lead to interesting insights. Further, more neuroscientific models such as the Fitzhugh-Nagumo or integrate-and-fire models [21] could also lead to relevant insights.

From the topological methods tested it has become clear that while all filtrations used were very promising at a theoretical level, this is no indication of their practical use. Further definitions of filtrations and simplicial complexes should therefore be considered. An example for a further definition of a simplicial complex is the D-neighbourhood complex, which is studied by [36]. This complex is formed by considering sets of neighbourhoods for every vertex in a graph. These neighbourhoods consist of vertices which are within a set distance D of one vertex and are used to define the simplices. Codes to apply these simplicial complexes have been developed by [36], but can so far only be used on connected graphs. Two approaches could be taken to use for analysis on our data sets: Firstly, the edge weights of the functional networks could be interpreted as distances [36]. Secondly, a filtration by weights could be applied followed by an algorithm that identifies separate connected components in every filtration step. The D-neighbourhood codes could then be used on all of these separately.

In general, to draw a significant conclusion for the data set, it would be important to use statistical methods to detect which significant features can be observed across the subjects. Moreover, these features would have to be detected in similar experiments. From a big-picture perspective, it is also important that when defining a functional network based on similarity between two time-series, strong edge weights do not imply a causal relationship. There are many other measures for similarity besides the one used in the data set (and the Kuramoto model) [38], which can retain different information in a similarity network. Similarly, in the data set, the definition of the nodes and the choice of the number of nodes to be measured also influences the outcome [44]. Combining findings from different node definitions and similarity measure would thus be useful, but would make both the experiments and the interpretation of the data unrealistically time-consuming.

In conclusion, the employed topological methods show a great potential for interesting insights in neuronal networks. However, issues such as computational scaling need to be addressed. Moreover, a good framework for barcode analysis needs to be developed and the full range of practically suitable definitions of filtrations and simplicial complexes is yet to be explored.

Bibliography

[1] Henry Adams and Andrew Tausz, JAVAPLEX *tutorial*, 2014. PDF version available at: `http://javaplex.googlecode.com/svn/trunk/reports/javaplex_tutorial/javaplex_tutorial.pdf`

[2] Bruce Alberts, Dennis Bray, Karen Hopkin, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts and Peter Walter, *Essential Cell Biology*. Garland Science, New York and London, 2004.

[3] Alex Arenas, Albert Díaz-Guilera and Conrad Pérez-Vicente, *Synchronization reveals topological scales in complex networks*. Physical Review Letters **96**: 11, 2006: 114102.

[4] Alex Arenas, Albert Díaz-Guilera, Jurgen Kurths, Yamir Moreno and Changsong Zhou, *Synchronization in complex networks*. Physics Reports **469**: 3, 2008: 93 – 153.

[5] Danielle S. Bassett, Nicholas F. Wymbs, Mason A. Porter, Peter J. Mucha, Jean M. Carlson and Scott T. Grafton, *Dynamic reconfiguration of human brain networks during learning*. Proceedings of the National Academy of Sciences of the United States of America **108**: 18, 2011: 7641 – 7646.

[6] Danielle S. Bassett, Mason A. Porter, Nicholas F. Wymbs, Scott T. Grafton, Jean M. Carlson and Peter J. Mucha, *Robust detection of dynamic community structure in networks*. Chaos **23**, 2013: 013142.

[7] Danielle S. Bassett, Nicholas F. Wymbs, M. Puck Rombach, Mason A. Porter, Peter J. Mucha and Scott T. Grafton, *Task-based core-periphery organisation of human brain dynamics*. PLoS Computational Biology **9**: 9, 2013: e1003171.

[8] Danielle S. Bassett, Muzhi Yang, Nicholas F. Wymbs and Scott T. Grafton, *Learning-Induced Autonomy of Sensimotor Systems*. arXiv: 1403.6034, 2014.

[9] Frederic Cazals and Chinmay Karande, *A note on the problem of reporting maximal cliques*. Theoretical Computer Science **407**: 1-3, 2008: 564 – 568.

[10] Fred H. Croom, *Basic Concepts of Algebraic Topology*. Springer, New York, Heidelberg, Berlin, 1978.

[11] Jacopo Binchi, Emanuela Merelli, Matteo Rucco, Giovanni Petri and Francesco Vaccarino, JHOLES*: A tool for understanding biological complex networks via clique weight rank persistent homology*. Electronic Notes in Theoretical Computer Science **306**,2014: 5 – 18.

[12] Michael Breakspear, Stewart Heitmann and Andreas Daffertshofer, *Generative models of cortical oscillations: neurobiological implications of the Kuramoto model*. Frontiers in Human Neuroscience **4**, 2010: 190.

[13] Edward T. Bullmore and Olaf Sporns, *Complex brain networks: graph theoretical analysis of structural and functional systems*. Nature Reviews Neuroscience **10**, 2009: 186 – 198.

[14] Edward T. Bullmore and Danielle S. Bassett, *Brain Graphs: Graphical Models of the Human Brain Connectome*. Annual Review of Clinical Psychology **7**, 2011: 113 – 140.

[15] Edward T. Bullmore and Olaf Sporns, *The economy of brain network organization*. Nature Reviews Neuroscience **13**, 2012: 336 – 349.

[16] Gunnar Carlsson, *Topology and data*. Bulletin of the American Mathematical Society **46**, 2009: 255 – 308.

[17] Yu Dabaghian, Facundo Mémoli, L. Frank and Gunnar E. Carlsson, *A topological paradigm for hippocampal spatial map formation using persistent homology*. PLoS ONE **8**: 8, 2012: e1002581.

[18] Herbert Edelsbrunner, David Letscher and Afra Zomorodian, *Topological persistence and simplification*. Discrete and Computational Geometry **28**, 2002: 511 – 533.

[19] Herbert Edelsbrunner and John L. Harer, *Persistent homology - a survey*. Contemporary mathematics **453**, 2008: 257 – 282.

[20] Herbert Edelsbrunner and John L. Harer, *Computational Topology*. American Mathematical Society, Providence R. I., 2010.

[21] Jianfeng Feng, *Is the integrate-and-fire model good enough? – A review*. Neural Networks **14**, 2001: 955 – 975.

[22] Robert Ghrist, *Barcodes: The persistent topology of data*. Bulletin of the American Mathematical Society **45**, 2008: 61 – 75.

[23] Shamik Gupta, Alessandro Campa and Stefano Ruffo, *Kuramoto model of synchronization: Equilibrium and nonequilibrium aspects*. arXiv: 1403.2083, 2014.

[24] David Kaplan, *Fftsurrogate: A code creating phase randomized surrogate data from a time series*, 1996. Code available at: `http://www.macalester.edu/~kaplan/Software/Software/fftsurr.m`

[25] Czes Kosniowski, *A First Course in Algebraic Topology*. Cambridge University Press, Cambridge, London, New York, New Rochelle, Melbourne, Sydney, 1980.

[26] Yoshiki Kuramoto, *Chemical Oscillations, Waves, and Turbulence*. Springer, Berlin, New York, 1984.

[27] Yoshiki Kuramoto, *Self-entertainment of a population of coupled non-linear oscillators*, in: Arakai, H. (Ed.): International Symposium on Mathematical Problems in Theoretical Physics, Lecture Notes in Physics **39**. Springer, Berlin, New York, 1975: 420 – 422.

[28] Hyekyoung Lee, Moo K. Chung, Hyejin Kang, Bung-Nyun Kim and Dong Soo Lee, *Discriminative persistent homology of brain networks*. IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2011: 841 – 844.

[29] Hyekyoung Lee, Hyejin Kang, Moo K. Chung, Bung-Nyun Kim and Dong Soo Lee, *Weighted functional brain network modelling via network filtration*. NIPS Workshop on Algebraic Topology and Machine Learning, 2012.

[30] Joakim Munkhammar, *Simulation of Kuramoto's model*, 2010. Code available at: `http://www.collective-behavior.com/Simulations/Ch6BoxA.m`

[31] James R. Munkres, *Topology*. Pearson Prentice Hall, New Jersey, 2000.

[32] Mark E. J. Newman, *Networks: An introduction*. Oxford University Press, Oxford, 2013.

[33] David Papo, Massimiliano Zanin, José A. Pineda-Pardo, Stefano Boccaletti and Javier M. Buldú, *Functional brain networks: great expectations, hard times, and the big leap forward*. arXiv: 1406.4006, 2014.

[34] Giovanni Petri, Martina Scolamiero, Irene Donato and Francesco Vaccarino, *Topological Strata of Weighted Complex Networks*. PLoS ONE **8**: 6, 2013: e66505.

[35] Mason A. Porter, Jukka-Pekka Onnela and Peter J. Mucha, *Communities in networks*. Notices of the American Mathematical Society **56**: 9, 2009: 1082 – 1166.

[36] Corrine Previte, *Personal communication*.

[37] Dean Prichard and James Theiler, *Generating surrogate data for time series with several simultaneously measured variables*. Physical Review Letters **73**: 7, 1994: 951 – 954.

[38] Stephen M. Smith, Karla L. Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F. Beckmann, Thomas E. Nichols, Joseph D. Ramsay, Mark W. Woolrich, *Network modelling methods for fMRI*. NeuroImage (Elsevier) **54**: 2, 2011: 875 – 891.

[39] Olaf Sporns, *Contributions and challenges for network models in cognitive neuroscience*. Nature Reviews Neuroscience **17**: 5, 2014: 653 – 660.

[40] Steven H. Strogatz, *From Kuramoto to Crawford: Exploring the onset of synchronisation in populations of coupled oscillators*. Physica D **143**, 2000: 1 – 20.

[41] David Sumpter, *Collective Animal Behaviour*. Princeton University Press, 2010.

[42] Andrew Tausz, Mikael Vejdemo-Johansson and Henry Adams, JAVAPLEX*: A research software package for persistent (co)homology*, 2011. Software available at: `http://javaplex.github.io/`

[43] Rebecca M. Todd, Margot J. Taylor, Amanda Robertson, Daniel B. Cassel, Sam M. Doesberg, Daniel H. Lee, Pang N. Shek, Elizabeth W. Pang, *Temporal-spatial neural activation patterns linked to perceptual encoding of emotional salience*. PLoS ONE **9**: 8, 2014: e105648.

[44] Fabrizio De Vico Fallani, Jonas Richiardi, Mario Chavez and Sophie Archard, *Graph analysis of functional brain networks: practical issues in translational neuroscience*. arXiv: 1406.7391, 2014.

[45] Jefferey Wildmann, *Bron-Kerbosch maximal clique finding algorithm*, 2011. Code available at: `http://www.mathworks.co.uk/matlabcentral/fileexchange/30413-bron-kerbosch-maximal-clique-finding-algorithm`

[46] Afra Zomorodian, *The tidy set: A minimal simplicial set for computing homology of clique complexes*. Proceedings of the 2010 annual symposium on Computational geometry, 2010: 257 – 266.

Appendix A

Appendix

A.1 Additional definitions from topology and algebra

Most of the following definitions can be found in [10]. For some of the more basic topological definitions we used [31], but note that [10] contains equivalent but more complicated formulations in most cases.

Definition A.1.1 (homeomorphism)**.** Let $\mathbb{X}$ and $\mathbb{Y}$ be topological spaces and $\phi : \mathbb{X} \longrightarrow \mathbb{Y}$ be a bijection. ϕ is called a *homeomorphism* if and only if both ϕ and

$$\phi^{-1} : \mathbb{X} \longrightarrow \mathbb{Y}$$

are continuous.

Definition A.1.2 (Hausdorff space)**.** A topological space $\mathbb{X}$ is called a *Hausdorff space* if for every pair $x_1, x_2 \in \mathbb{X}$ of distinct points there exist neighbourhoods U_1 and U_2 of x_1 and x_2 respectively that are disjoint.

Definition A.1.3 (compact)**.** A topological space $\mathbb{X}$ is said to be *compact* if for every covering of $\mathbb{X}$ by open sets we can find a finite subcollection of open sets that also covers $\mathbb{X}$.

Definition A.1.4 (m-manifold)**.** An *m-manifold* is a compact, connected Hausdorff space $\mathbb{X}$ such that each point $x \in \mathbb{X}$ has a neighbourhood that is homeomorphic to an open subset of $\mathbb{R}^m$.

Definition A.1.5 (compact surface)**.** We call a 2-manifold a *compact surface*.

Definition A.1.6 ((Abelian) group)**.** A *group* $(G, *)$ is a set G together with a binary operation $* : G \times G \to G$ satisfying the following properties:

i) The binary operation is associative on G: $a*(b*c) = (a*b)*c$ for all $a, b, c \in G$.
ii) There is a neutral element $e \in G$ such that $e * a = a * e = a$ for all $a \in G$.
iii) For every $a \in G$ there is an inverse element a^{-1} such that $a * a^{-1} = e$ for any $a \in G$.

If the binary operation is in addition commutative on G, i.e. $a * b = b * a$ for all $a, b \in G$, we say the group is *Abelian*.

If the binary operation is additive, the inverse element of $a \in G$ is usually denoted as $-a$.

Definition A.1.7 (subgroup)**.** A subset G' of G is a *subgroup* if it fulfils the following properties:

i) G' is closed under the group operation: $a', b' \in G'$ implies $a' * b' \in G'$.
ii) $e \in G'$.
iii) $a' \in G'$ implies $a'^{-1} \in G'$.

Definition A.1.8 (normal subgroup)**.** A subgroup N of G is said to be *normal* if for every element $a \in N$ and every $b \in G$ it holds that

$$b * a * b^{-1} \in N. \tag{A.1}$$

Definition A.1.9 (group homomorphism)**.** Let $(G, *)$ and $(H, \circ)$ be groups. A *group homomorphism* $\varphi : G \to H$ is a map such that

$$\varphi(a * b) = \varphi(a) \circ \varphi(b),$$

for all $a, b \in G$.

Definition A.1.10 (kernel)**.** Let $(G, *)$ and $(H, \circ)$ be groups and $e_H \in H$ the neutral element in H. The set

$$\ker \varphi = \{g \in G : \ \varphi(g) = e_H\}$$

is the kernel of the group homomorphism φ.

Proposition A.1.1. *The kernel of a group homomorphism $\varphi : G \to H$ is a normal subgroup of G.*

Definition A.1.11 (image)**.** Let $(G, *)$ and $(H, \circ)$ be groups. The set

$$\mathrm{Im}\varphi = \{h \in H : \ \varphi(g) = h \text{ for some } g \in G\}$$

is the *image* of the group homomorphism φ.

Proposition A.1.2. *The image of a group homomorphism $\varphi : G \to H$ is a normal subgroup of H.*

Definition A.1.12 (coset)**.** Let A be a subgroup of G and $g \in G$. We say the set

$$gA = \{g * a : \ a \in A\} \tag{A.2}$$

is the *left coset* of A by g. Right cosets Ag are defined analogously.

Proposition A.1.3. *If A is a normal subgroup of G, it holds that $gA = Ag$.*

Definition A.1.13 (quotient group)**.** Let A be a normal subgroup of G. We call the family of all cosets G/A of A together with the operation

$$gA * hA = (g * h)A \tag{A.3}$$

the *quotient group* of G modulo A.

Proposition A.1.4. *G/A is a group.*

Definition A.1.14 (set of generators of a group)**.** For a group G, we call a subset $X = \{g_1, \ldots, g_k\} \subseteq G$ a *set of generators* of G if every element in G can be written as a product of elements from X. We write $\langle X \rangle = G$.

Definition A.1.15 (ring)**.** A *ring* $(R, *, \circ)$ is a set R together with two binary operations $+$ and $*$ such that:

i) $(R, +)$ is an Abelian group.
ii) The operation $*$ is associative: $(a * b) * c = a * (b * c)$,
iii) $a * (b + c) = (a * b) + (a * c)$,
iv) $(b + c) * a = (b * a) + (c * a)$,

for all $a, b, c \in R$.

Appendix B

Appendix

B.1 Further general barcodes

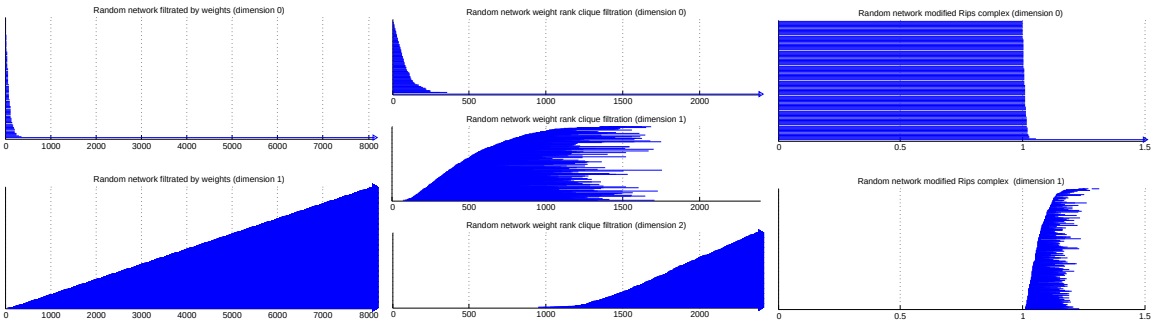

Figure B.1: Barcodes of a filtration by weights (left), a weight rank clique filtration (middle) and a modified Rips complex (right) performed on a network of 128 nodes with edge weights drawn uniformly from the interval $(0, 1)$.

B.2 Further Kuramoto barcodes

B.2.1 Kuramoto filtration by weights barcodes for five time layers

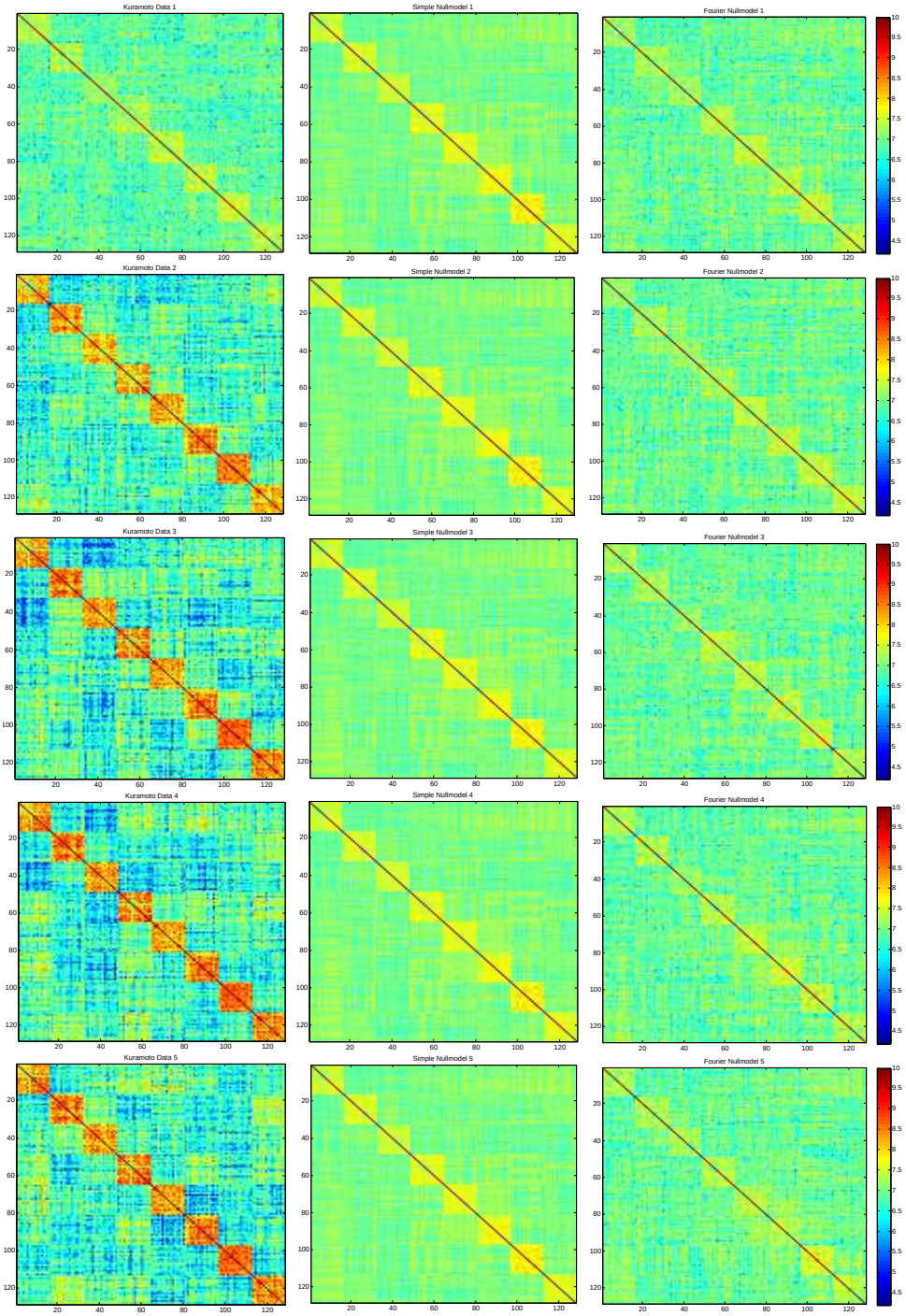

Figure B.2: Functional networks generated from the Kuramoto model, the simple null model, and the Fourier null model. Row 1: Networks based on time steps 1–100. Row 2: Networks based on time steps 101–200. Row 3: Networks based on time steps 201–300. Row 4: Networks based on time steps 301–400. Row 5: Networks based on time steps 401–500.

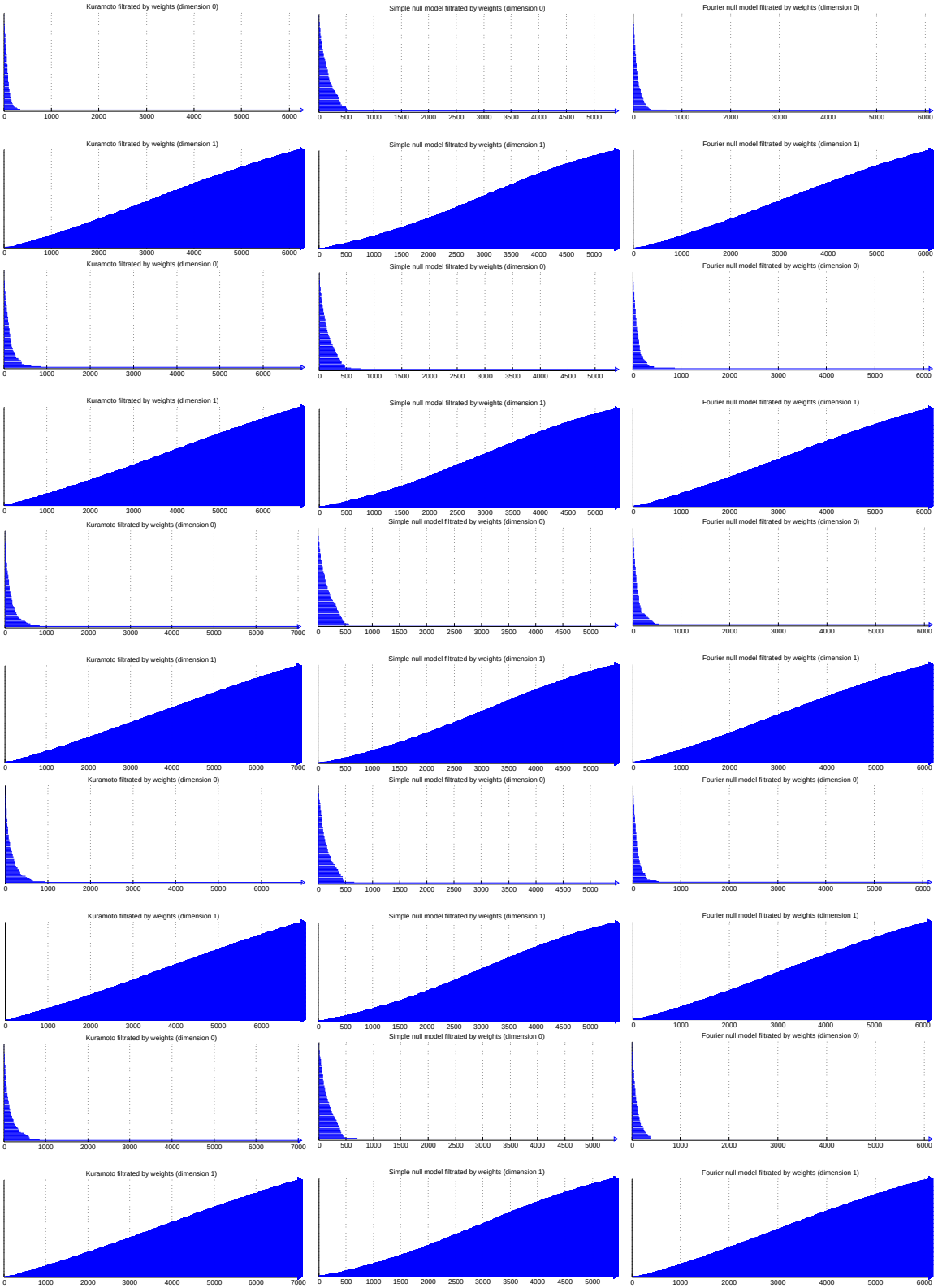

Figure B.3: Barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. Row 1: Barcodes generated from functional matrices based on time steps 1-100. Row 2: Barcodes generated from functional matrices based on time steps 101–200. Row 3: Barcodes generated from functional matrices based on time steps 201–300. Row 4: Barcodes generated from functional matrices based on time steps 301–400. Row 5: Barcodes generated from functional matrices based on time steps 401–500.

B.2.2 Kuramoto filtration by weights barcodes for fixed natural frequencies and two time layers

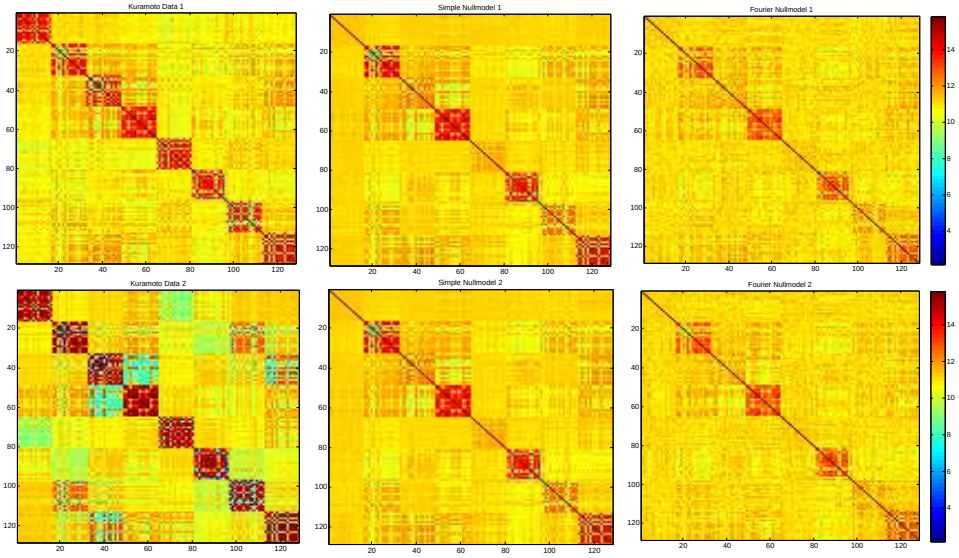

Figure B.4: Functional matrices for fixed natural frequencies and two time layers of the Kuramoto model, the simple null model, and the Fourier null model.

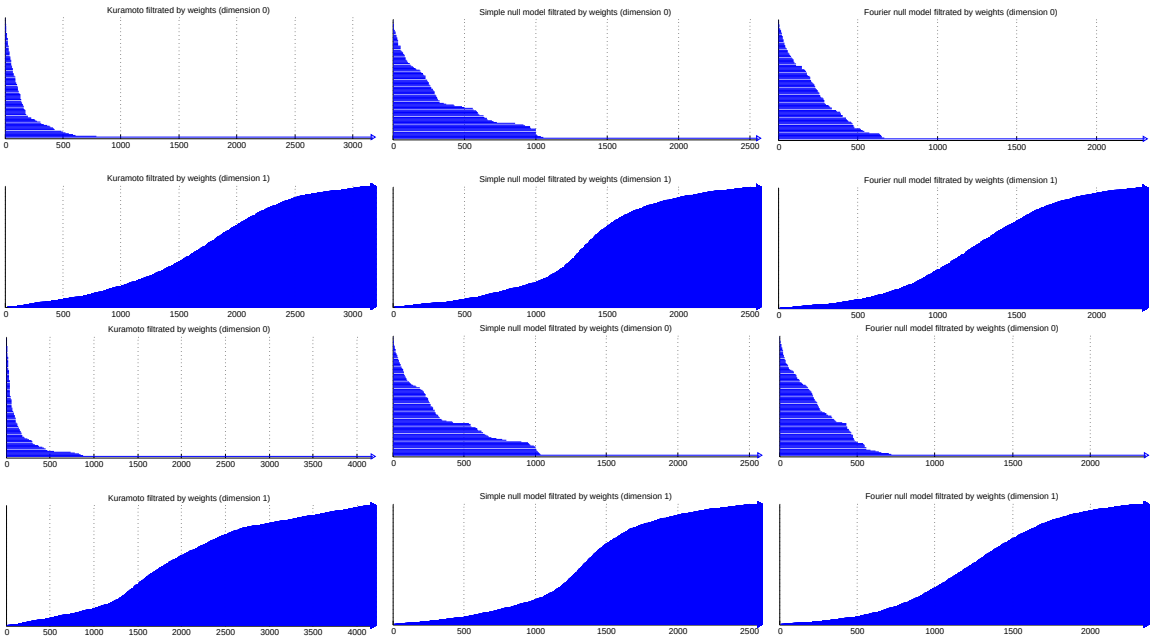

Figure B.5: Filtration by weights barcodes for fixed natural frequencies and two time layers of the Kuramoto model, the simple null model, and the Fourier null model.

B.2.3 Kuramoto modified Vietoris-Rips complex barcodes

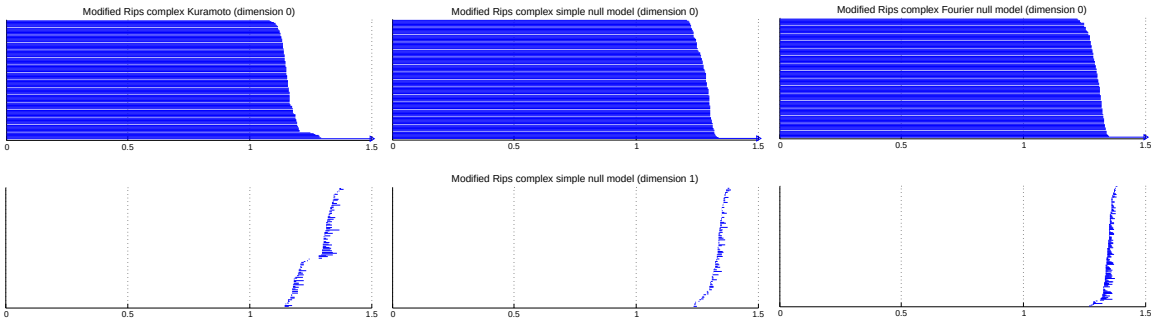

Figure B.6: Modified Vietoris-Rips complex barcodes generated from the Kuramoto model, the simple null model, and the Fourier null model. The functional networks were based on all time steps.

B.3 Further data barcodes

B.3.1 Filtration by weights

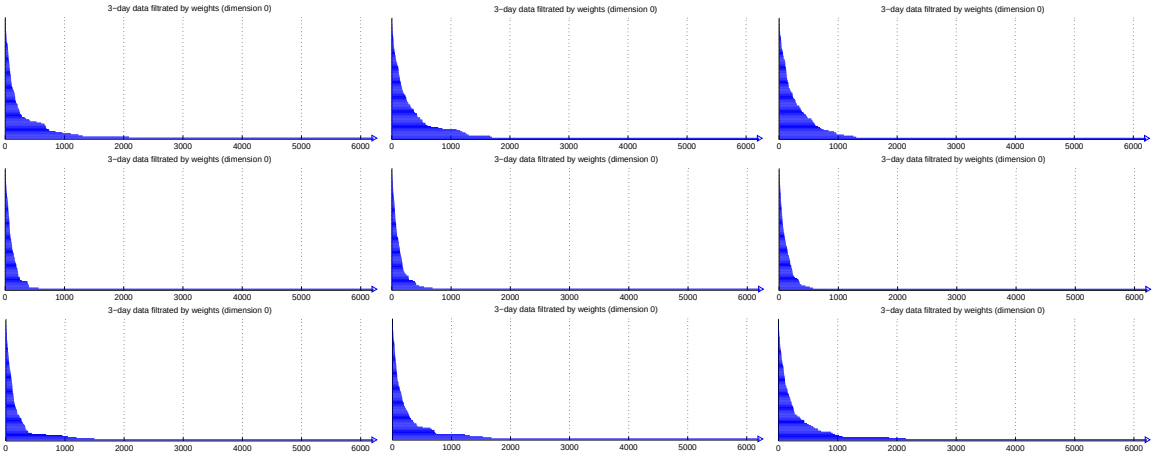

Figure B.7: Further examples of filtration by weights barcodes generated bases on data from subjects 6, 9, and 15. The barcodes are shown in temporal order from left to right.

B.4 Motor and visual modules in the human brain

The motor and visual modules are listed according to [8], where experiments were conducted using the same brain areas as in the data analysed in this dissertation.

Sensimotor component	node	Primary visual component	node
Left precentral gyrus	7	Left intracalcrine cortex	24
Right precentral gyrus	55	Right intracalcrine cortex	72
Left postcentral gyrus	17	Left cuneus cortex	32
Right postcentral gyrus	65	Right cuneus cortex	80
Left superior parietal lobule	18	Left lingual gyrus	36
Right superior parietal lobule	66	Right lingual gyrus	84
Left supramarginal gyrus, anterior	19	Left supracalcrine cortex	47
Right supramarginal gyrus, anterior	67	Right supracalcrine cortex	95
Left supplemental motor area	26	Left occipital pole	48
Right supplemental motor area	74	Right occipital pole	96
Left parietal operculum cortex	43		
Right supramarginal gyrus posterior	68		