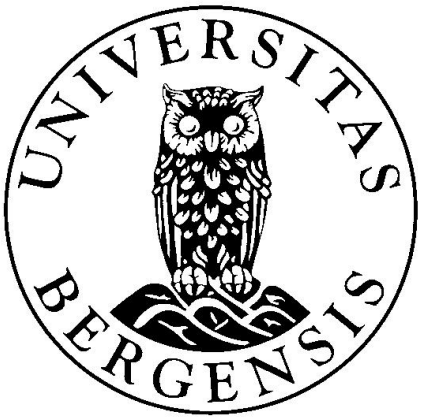
Using the bottleneck distance to compare homological methods of detecting atmospheric rivers

Kim Alexander Haugland

January 17, 2018



Acknowledgements

I wish to thank my supervisors Morten Brun and Bjørn Ian Dundas for their patience, advice and input, as well as student adviser Kristine Lysnes for her encouragement and support. I also wish to thank Kristian Alfsvåg and Clemens Spensberger for their assistance regarding python and technical matters. Lastly, I wish to thank Thomas Spengler for introducing me to the project in the first place.

Contents

1 Introduction

On the 13th of September 2005, the extreme weather "Kristin" hit the west coast of Norway, leading to floods, landslides and the loss of three lives [Stohl]. This event came about due to a transport of moisture across the Atlantic; an atmospheric river.

An atmospheric river is defined as a long and narrow region of intense vertically integrated water transport in the atmosphere (see [Rutz]). In his Master's thesis, Kristian Alfsvåg attempted to detect these rivers using persistent homology. The data he used is called the *total column water* (or *tcw* for short), which is defined as the integral of the water density from a point on the Earth's surface directly upwards to the top of the atmosphere (given in kg/m^2). This data came from the ERA-Interim project, which has recorded arrays of meteorological data from 1979 until today [Dee et al.].

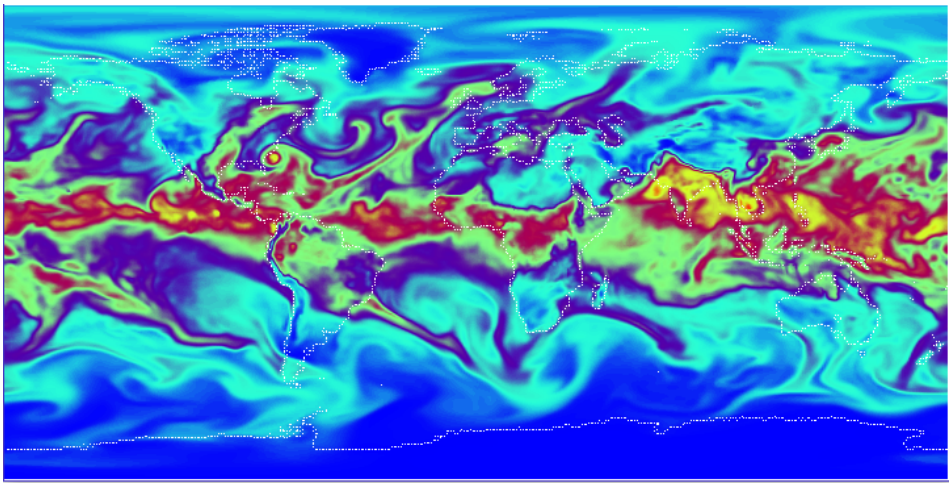


Figure 1: Here we see a visualization of the *tcw* data from the ERA-Interim project showing the atmospheric river responsible for "Kristin" as it hits Bergen

Using this data, he created two algorithms to detect atmospheric rivers hitting Bergen, one involving two-dimensions and one involving three-dimensions. His two-dimensional algorithm, called the *bec* algorithm, appeared to be very successful at its task. He then moved on to add a time dimension, in the hopes of detecting more general plumes of humidity being transported across the Atlantic to Bergen which are not connected to the equator at any fixed time-step (henceforth only referred to as "plumes", so as to distinguish them from rivers). This three-dimensional algorithm he called the *tbec* algorithm. The results of these calculations were more ambiguous, and it is not clear

whether they detect what we want them to or something else entirely. In this thesis, I continued where Alfsvåg left off and compared the two methods more thoroughly using the bottleneck distance of persistent homology, in an attempt to uncover what exactly the *tbec* is detecting. I then created a new three-dimensional algorithm called the *altbec* in an attempt to fix what I perceived to be the problem. As we shall see, the results were surprising, but still useful.

I was a bit worried at first that such an experimental thesis might lack the theoretical depth expected of a Master's thesis in mathematics. However, as I read about the bottleneck distance, it became clear to me that I could simplify an important proof somewhat by working at the right "level of generality". Thus, the thesis turned out to have quite a large theoretical component, and it is effectively split into two parts: A theory part exploring the bottleneck distance and proving the Isometry theorem (Sections 4 and 5), and a part where we use the bottleneck distance to compare the different methods of detecting rivers and plumes (Sections 6 and on). Before we get that far though, we need to go through some preliminaries (Section 2), and explain the *bec* and *tbec* in more detail (Section 3).

2 Preliminaries

2.1 Persistent homology and persistence diagrams

Definition 2.1. A filtered complex $\mathbb{X}$ is a CW-complex X equipped with a filtration of subcomplexes

$$\emptyset = X^0 \subset X^1 \subset X^2 \subset \dots$$

If X is a finite CW-complex (so that $X = X^N = X^{N+1} = \dots$ for some N), then we say that $\mathbb{X}$ is a finite filtered complex.

Note that the X^n are *subcomplexes*, not to be confused with n-skeleta.

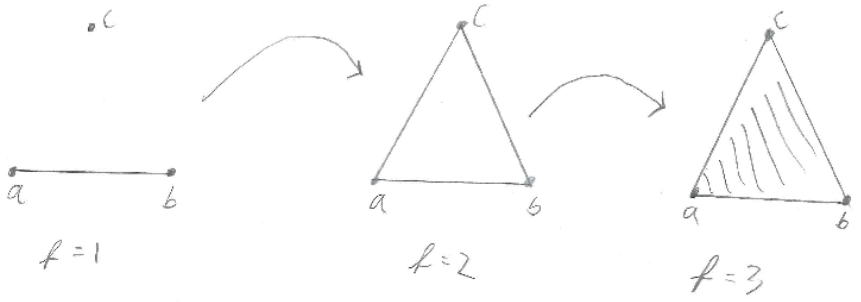
From this point on when we refer to filtered complexes, we mean finite filtered complexes unless otherwise stated. Also, in this thesis we only concern ourselves with homology calculated with field coefficients (for reasons which will become clear later).

Definition 2.2. Given a filtered complex $\mathbb{X}$, we define the *k-persistent homology* $H_k(\mathbb{X})$ to be the sequence:

$$\emptyset = H_k(X^0) \to H_k(X^1) \to \dots \to H_k(X^N) = H_k(X)$$

where the arrows are induced by the inclusions.

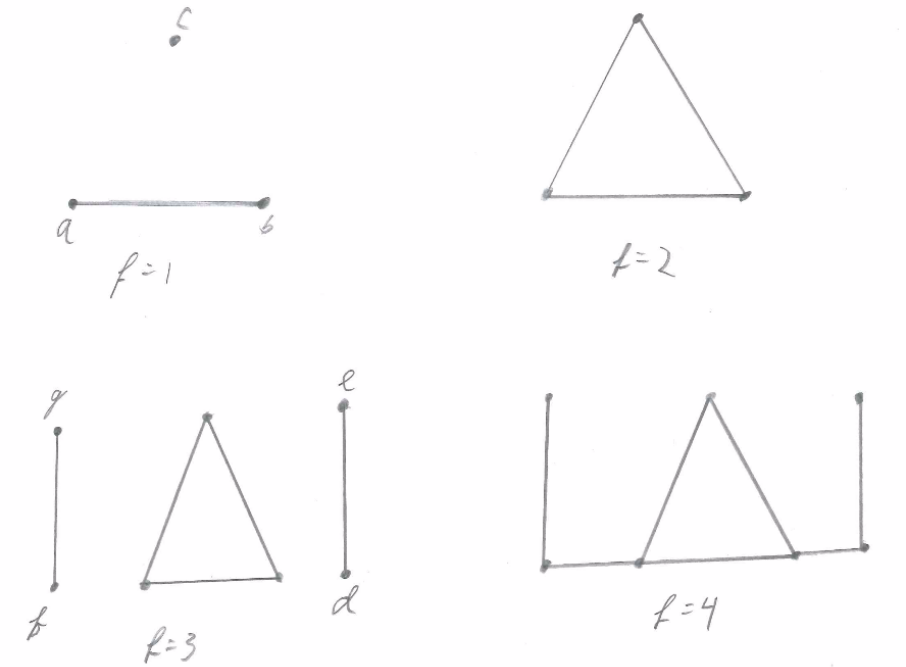
Example 2.3. Consider a 2-simplex X (a filled triangle) with a filtration $X^1 \subset X^2 \subset X^3 = X$ constructed in the following manner:
All of the cells are simplices (see [Z-C, p. 254] for the extra conditions required for a CW-complex to be a simplicial complex),
The simplices $[a], [b], [c], [ab]$ have filtration degree 1,
the simplices $[ac], [bc]$ have filtration degree 2,
and the simplex $[abc]$ has filtration degree 3.



In this case, the 0-persistent homology will have one cycle that lasts through all filtration degrees (we can for instance choose $[a]$ as the generator), and one cycle that starts in filtration degree 1 and is killed off in filtration degree 2 (if $[a]$ was chosen as the generator of the other cycle, then this cycle must necessarily be generated by $[c]$). The 1-persistence homology has a single cycle that is born in filtration degree 2 and dies in filtration degree 3. This cycle represents the hole in our triangle that occurs in filtration degree 2 and is filled in the next degree. This complex has no higher dimensional homology.

In general, cycles that last through many filtration degrees represent persistent topological features of our filtered complex. Here is a more detailed example:

Example 2.4. as before, let the simplices $[a], [b], [c], [ab]$ have filtration degree 1,
and let the simplices $[ac], [bc]$ have filtration degree 2.
Now, introduce simplices $[d], [e], [de], [f], [g], [fg]$ of filtration degree 3,
and finally the simplices $[bd][af]$ of filtration degree 4.



Considering the 0-persistence homology, we see that we ultimately have one connected component at filtration degree 5 and above, but in lower degrees we have several. In particular, we have a cycle lasting through all filtration degrees and a cycle lasting from filtration degree 1 to filtration degree 2 as before, and two cycles lasting from filtration degree 3 to filtration degree 4. For the 1-persistent homology, we have a single cycle that is born in filtration degree 2 and is never killed. This long cycle represents the fact that our space has a persistent one-dimensional hole.

Note that the information obtained by calculating persistent homology with field coefficients can be described by sets of intervals, with each interval starting at the time of birth and ending at the time of death of its corresponding cycle. The homology of a filtered complex changes a finite number of times, and in when calculating persistent homology we are only interested in the time of birth, the time of death, and the length of the cycles. So for simplicity, we can write all intervals as open without losing information that is relevant to us. For instance, the 0-persistent homology of the above example can be written as:

$$\{(1,\infty),(1,2),(3,4),(3,4)\}$$

Such a list of intervals is called a *bar code*.

Note that a bar code is not in general a set, since the same interval can occur several times in the same bar code as the interval $(3, 4)$ does in the example above. However, a bar code can be described as a *multiset*.

Definition 2.5. A multiset M is a pair (S, m) consisting of a set S and a multiplicity function $m : S \to \mathbb{N}_{\geq 1}$. If $a \in S$, then we call $m(a)$ the multiplicity of a.

Thus, the bar code above can be given by the set $\{(1, \infty), (1, 2), (3, 4)\}$ together with a multiplicity function m such that $m((1, \infty)) = 1$, $m((1, 2)) = 1$, and $m((3, 4)) = 2$. We define $\mathbb{B}$ to be the set of all bar codes and we write $\mathbb{B}(\mathbb{X}_k)$ to be the bar code obtained by taking the k-persistent homology of a filtered complex $\mathbb{X}$.

There are other ways we can represent persistent homology. In this thesis, the concept of *persistence diagrams* will play an important role. The idea behind a persistence diagram can easily be understood by noting that we can represent an open, real interval by a point in the *extended half-plane*:

$$\hbar = \{(a, b) \in \mathbb{R}^2 \,|\, a < b\} \cup \{-\infty\} \times \mathbb{R} \cup \mathbb{R} \times \{\infty\} \cup \{(-\infty, \infty)\}$$

Note that when it comes to persistent homology, all of the intervals (a, b) will have $a \geq 0$. Also note that we have defined the extended half-plane to not include the diagonal. This is so we don't have to consider the multiplicity of points on the diagonal.

Definition 2.6. Let $\mathbb{X}$ be a filtered complex and let $\mathbb{X}_k$ denote thhe k-persistent homology of $\mathbb{X}$. The *persistence diagram* of $\mathbb{X}_k$ is a multiset $dgm(\mathbb{X}_k) = \{(a, b) \in \mathbb{R}^2 \,|\, 0 \leq a < b, (a, b) \in \mathbb{B}(\mathbb{X}_k)\}$, where the multiplicity $m_{\mathbb{X}}((a, b))$ of a point is the multiplicity of the corresponding interval in the bar code.

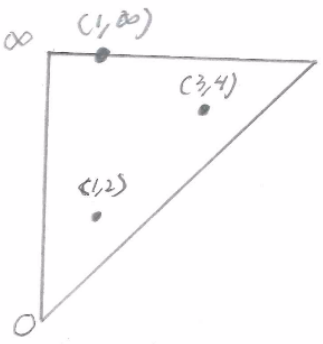


Figure 2: The 0-persistence diagram of the filtered complex in example 2.4

2.2 Calculating persistent homology

Let $\mathbf{k}$ be a field and $\mathbb{X}$ be a filtered complex. Then $H_k(\mathbb{X}, \mathbf{k})$ denotes the k-persistence homology with coefficients in $\mathbf{k}$. However, we shall for the sake of brevity simply write $H_k(\mathbb{X})$. We can define a graded module $\alpha(H_k(\mathbb{X}))$ over $\mathbf{k}[t]$ by $\alpha(H_k(\mathbb{X})) = \bigoplus_{i=0}^{\infty} H_k(X^i)$. The grading is given by filtration degree, so multiplying by t simply shifts the elements up one filtration degree.

Since $X^N = X^{N+1}$ for some N and each X^n is a finite CW-complex, this module is finitely generated. Since $\mathbf{k}$ is a field, then $\mathbf{k}[t]$ is a principal ideal domain. By the structure theorem for finitely generated, graded modules over a principal ideal domain ([Z-C][Theorem 2.1]), we get that:

$$\alpha(H_k(\mathbb{X})) \cong (\bigoplus_{i=1}^{n} t^{\alpha_i}\mathbf{k}[t]) \oplus (\bigoplus_{j=1}^{m} t^{\beta_j}\mathbf{k}[t]/t^{\gamma_j})$$

Let $\mathbf{B} \in \mathbb{B}$ be a bar code, and let m_i^j denote the multiplicity of an interval $(i, j) \in \mathbf{B}$. We have a bijection Q from the set of bar codes to the set of isomorphism classes of finitely generated, graded $\mathbf{k}[t]$-modules given by:

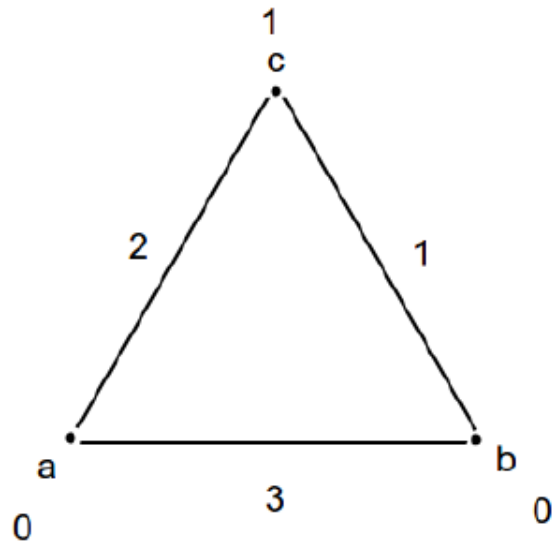
$$Q(\mathbf{B}) = \begin{cases} \bigoplus_{(i,j)\in\mathbf{B}} (t^i\mathbf{k}[t]/t^j)^{\oplus m_i^j} & \text{if} j \neq \infty \\ \bigoplus_{(i,j)\in\mathbf{B}} (t^i\mathbf{k}[t])^{\oplus m_i^j} & \text{if} j = \infty \end{cases}$$

Here, the m_i^j's encode that if you have several copies of the same interval, these are each sent to a different copy of the corresponding module (and vice versa).

We see that the target modules are of the same form as the ones occurring in the decomposition of $\alpha(H_k(\mathbb{X}))$. Hence all the information we need to represent the k-persistent homology with field coefficients is contained in the α_i's, β_j's and γ_j's. Specifically, the α_i's tell us the birth values of cycles which go to infinity, and the β_j's and γ_j's tell us the respective birth and death values of cycles with a finite lifespan.

To obtain these values, we start with the standard matrix reduction algorithm for computing homology [Z-C, p. 256]. The twist is that we also need to keep track of the filtration values of the chains. This is done by setting the degree of the element in the (i, j)-th square of a matrix to be the filtration degree of the corresponding column basis element minus the filtration degree of the corresponding row basis element. By induction, we can assume that we have a homogeneous basis for the cycles $Z_{k-1} \subset C_{k-1}$, and a matrix A_k representing the map $\tilde{\delta}_k : C_k \to Z_{k-1}$ (the initial case is trivial, since $Z_0 = C_0$). These cycles are sorted from the highest to the lowest filtration degree.

In the case where several cycles appear at the same filtration degree, these are sorted arbitrarily. We obtain the matrix A_{k+1} by using column operations to reduce A_k to column-echelon form. The column basis elements corresponding to zeroed-out columns after this reduction give a homogeneous basis for Z_k.



$$\begin{array}{c} \\ c \\ b \\ a \end{array}\begin{array}{c} \begin{array}{ccc} bc & ac & ab \end{array} \\ \begin{pmatrix} 1 & t & \\ t & & t^3 \\ & t^2 & t^3 \end{pmatrix} \end{array} \sim \begin{array}{c} \\ c \\ b \\ a \end{array}\begin{array}{c} \begin{array}{ccc} bc & ac+tbc & ab+tac+t^2bc \end{array} \\ \begin{pmatrix} 1 & & 0 \\ t & t^2 & 0 \\ & t^2 & 0 \end{pmatrix} \end{array}$$

Figure 3: This figure illustrates the matrix reduction process discussed above used to calculate the 0-persistence homology of a simple filtered complex. Figure taken from [Alfsvåg]

Considering the matrix $\tilde{A}_k$, the column-echelon form of the matrix A_k, we note that because of our sorting of the row basis elements, the degree of the entries in each column are monotonically increasing as we move down the matrix. This means that we can further reduce the matrix to normal form without affecting the degree of the pivot elements or the row basis elements. Hence, there is no need to reduce the matrix to normal form at all! All the information we require is given by the degrees of the pivot elements and the degrees of the row basis elements.

For instance, if row i has the pivot $\tilde{A}_k(i,j) = t^n$ (a pivot element of degree n), and its row basis element is of degree d_i, then it contributes the module $t^{d_i}\mathbf{k}[t]/t^{d_i+n}$ to the description of $H_{k-1}(\mathbb{X})$. That is, the matrix shows us that there is a cycle that is born at degree d_i and dies at degree $d_i + n$. Non-pivot rows contribute modules of the form $t^{d_i}\mathbf{k}[t]$. The intervals in the bar code for $H_{k-1}(\mathbb{X})$ corresponding to pivot and non-pivot rows are $(d_i, d_i + n)$ and (d_i, ∞), respectively.

This algorithm ensures that if a cell c appears and kills a cycle, the filtration degree of the cycle getting killed will be the highest filtration degree of the cells in the boundary of c. In other words, in the event that several cycles could be killed by a cell appearing, it is always the youngest cycle which is killed (youngest in the sense that it is born at the highest filtration degree). This is called *the elder rule*, and it is vital for calculating persistent homology (since we want to detect the cycles which persist through many filtration degrees).

To study this algorithm in more detail and learn how to implement it on a computer, see [Z-C].

3 What has been done

In his master's thesis (see [Alfsvåg]), Kristian Alfsvåg used the total column water (*tcw*) data from the ERA-Interim project to construct two different filtered complexes, one two dimensional and one three dimensional. Then he calculated the persistent homology of these filtered complexes with with coefficients in $\mathbb{Z}/2$. The idea was that the information obtained about the "shape" of these spaces would be enough to detect atmospheric rivers (and plumes, in the 3D case). These algorithms he named the *bec* and the *tbec* algorithm, respectively.

Before we go into the *bec* and *tbec* algorithms in more detail, we discuss how to construct a filtered complex from *tcw* data in general.

The *tcw* is sampled along a *grid* on the Earth's surface. Alfsvåg used a grid with whole-degree intervals (longitude and latitude). A grid with half-degree intervals was available, but a coarser grid was chosen in order to save computation time. In general, a two dimensional grid G can be defined as:

$$G = \coprod_{i=1}^{N} \{0, 1, \ldots, n_{i-1}\}$$

We can think of this grid G as having N rows of varying length n_i.

If G is three dimensional, it can be defined as:

$$G = \coprod_{i=1}^{N} (\{0, 1, \ldots, n_{i-1}\} \times \{0, 1, \ldots, m_{i-1}\})$$

Here it is useful to visualize G as a stack of "discretized planes" (that is, as points evenly spaced on horizontal planes), each with length n_i and width m_i.

In the data Alfsvåg and myself are using, the *tcw* is sampled in six hour intervals. For the *tbec* algorithm, a time dimension based on this sampling is included as the grid's third dimension.

Using the points of a grid as vertices, we now want to construct a filtered complex. To do this, we introduce a function $f : G \to \mathbb{R}$ that assigns a value to each grid point. In our case, this function will give the *tcw* value at each point. There are two ways we can construct a filtered complex from this information: the *top-down* and the *bottom-up* filtration. In both cases, the filtration value of each grid point will be its function value.

The top-down filtration adds the points with the highest filtration value first, and assigns the filtration value of a cell between two adjacent grid points to be the minimum of the filtration value of said points. For higher dimensional cells, the filtration value is given by the minimum filtration value of the cells in its boundary. In practice, the top-down filtration fills in the most humid areas first, and then the entire space is filled in as drier and drier points are allowed. This is the filtration that Alfsvåg used for his *bec* and *tbec* calculations.

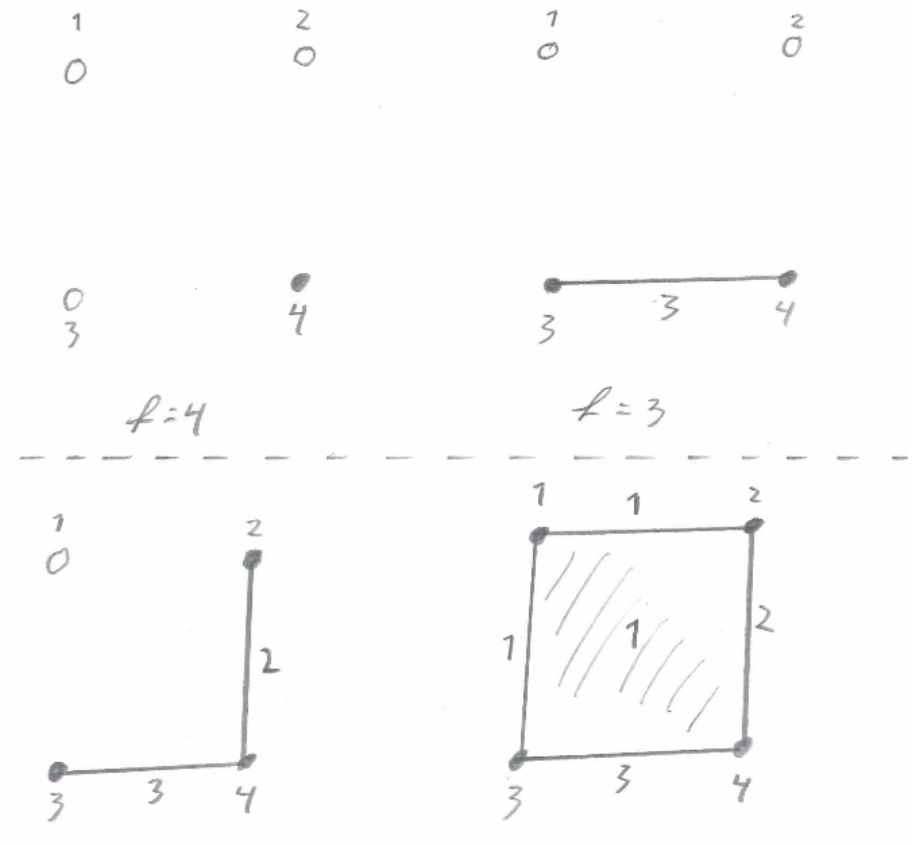


Figure 4: The top-down filtration of a 2x2 grid.

The bottom-up filtration mirrors the top-down filtration. It adds the points with the lowest filtration value first (the driest points), and then gives the filtration value of 1-cells and higher dimensional cells by taking the maximum filtration value of the cells in their boundaries. As one might expect, this filtration fills in the driest areas first, and the entire space is filled when the most humid areas are added. In section 7, we shall introduce an alternative (or supplementary) algorithm to the *tbec* that uses a bottom-up filtration.

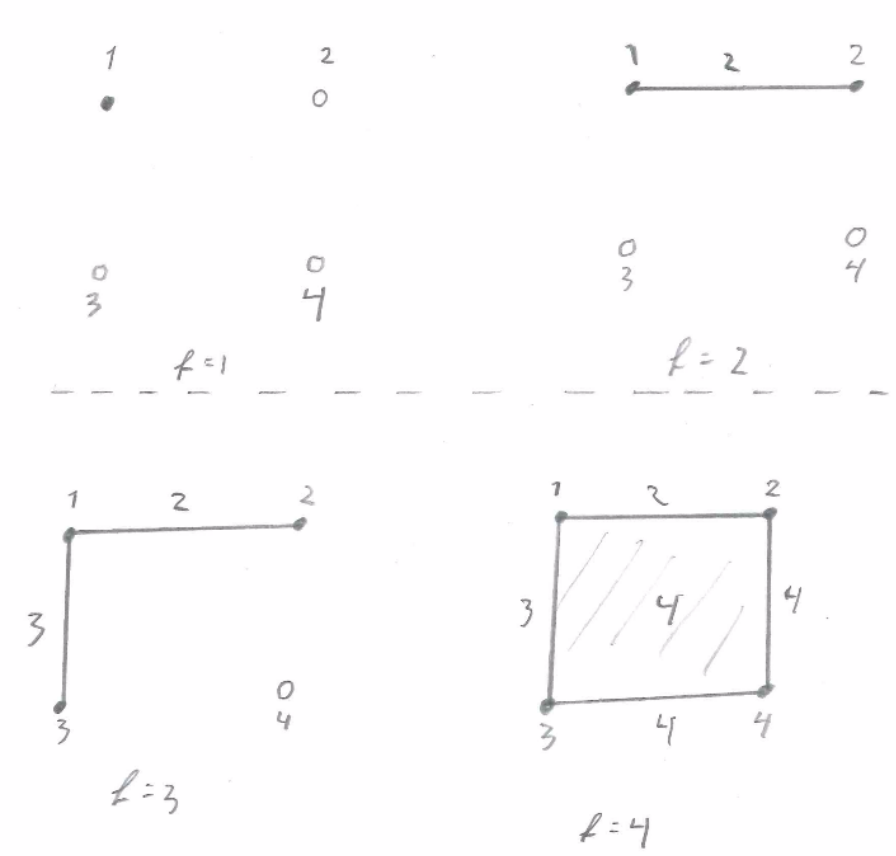


Figure 5: The bottom-up filtration of the same 2x2 grid as above.

We now describe the *bec* algorithm in more detail. Since we are interested in atmospheric rivers hitting Bergen, there is no need to include the southern hemisphere in our grid, as atmospheric rivers always come from humid areas around the equator. Additionally, because of the Coriolis effect and because of the large, dry landmasses to the south and south-east of Norway, atmospheric rivers that hit Bergen almost always originate in the south-west. Because of this, Alfsvåg chose a grid with the southern part of Norway in the upper right corner, and the Gulf of Mexico in the lower left corner.

Specifically, the grid he chose was

$$G = \{-95°E, -94°E, \ldots, 15°E\} \times \{0°N, 1°N, \ldots, 65°N\}$$

Figure 6: The area covered by the grid G

The *bec* algorithm works by first adding a point with a 1-cell connecting it to Bergen. We "artificially" give this point the maximal filtration value. We also set the filtration value of points along the equator to be more than the rest of the points in the filtration, but less than the added point connected to Bergen. We now compute the persistent 0-homology of the filtered complex resulting from taking the top-down filtration. Because of our prior modification of the filtration values, the cell generated by the added vertex will be the oldest cell in the filtration, and the cell generated by the equator will be the second-oldest. By the elder rule, we then have that Bergen is connected to the equator when the second-oldest cell is killed. We call the filtration value at which this occurs the *bec*, short for "Bergen equator connected" (hence why it is called the *bec* algorithm). If the *bec* is high, this means that there is a path of high humidity from the equator to Bergen: a possible atmospheric river.

When we test this algorithm and compare it to a visualization of the meteorological data, we see that it is indeed a good algorithm for detecting atmospheric rivers. However, due to its two dimensional nature, it cannot detect plumes (as plumes are not connected to the equator in the two dimensional case). We need to introduce a time dimension to our filtration in order to do that.

To move to three dimensions, we select a time interval $\{t_i, t_{i+1}, \ldots, t_{i+m}\}$ starting at some time step t_i, and where the t_i and t_{i+1} are six hours apart (recall that the *tcw* is sampled every six hours, so for a non-leap year, $i = 1, 2, \ldots, 1460$). We want to detect rivers and plumes hitting Bergen during this interval. However, we must first add the interval $\{t_{i-n}, \ldots, t_{i-1}\}$ to our space as well, in order to allow rivers and plumes originating earlier than t_i to be detected. In the calculations done so far, a month was selected as the desired interval for detection, and rivers/plumes originating up to 10 days earlier than the start of the month were allowed. For example, for the month of February in any given year which isn't a leap year, the grid of vertices would be:

$$G = \{-95°E, -94°E, \ldots, 15°E\} \times \{0°N, 1°N, \ldots, 65°N\} \times \{t_{85}, t_{86}, \ldots, t_{236}\}$$

since the *tcw* is sampled in six hour intervals.

However, in this case we are only interested in detecting rivers and plumes that hit Bergen after t_{124} (that is, in February). To achieve this, we artificially set the filtration value of the vertices at $\{t_{85}, t_{86}, \ldots, t_{124}\}$ to be the minimum. Since we are doing a top-down filtration, that means that these points won't be added until the final stage of the filtered complex. In the general case, we do this to vertices in the interval $\{t_{i-n}, \ldots, t_{i-1}\}$.

Before we build our filtered complex, we once again attach an extra vertex to Bergen as we did in the two dimensional case. However, we must now do this at every time step. In addition, we attach a line from a point at the equator to this added vertex. We give the equator, the line, and the vertex the maximum filtration value, so that they are included in the first step of the filtration. Note that we now have the potential for the same river or plume to generate many persistent cycles, since they may hit Bergen at several time steps. To counteract this, we take the lines from Bergen to the added vertex to be boundaries of 2-simplexes. These 2-simplexes are included in our filtered complex, and given the "expected" filtration values (the minimum filtration values of their boundaries).

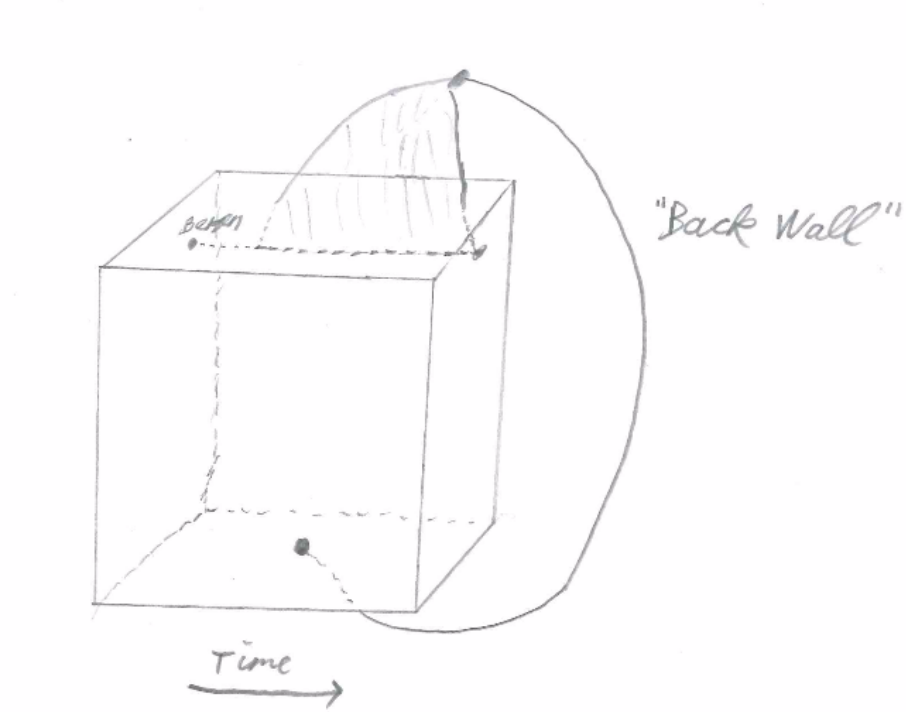


Figure 7: An illustration of the 3D grid used in the *tbec* algorithm with an attached back wall connecting Bergen to the equator.

Now that we have built our filtered complex, we move on to the persistent homology calculation. Since we are only interested in cycles going from the equator to Bergen, we discard all cycles which do not travel along "the back wall". Specifically, a cycle is discarded if it does not include the line from the equator to the added vertex.

In order to compare the *bec* and *tbec* algorithms, we need to calculate the *bec* for an entire month at a time. To do that, we use a height function on monthly graphs of the *bec*, and then calculate the 0-persistent homology given by this height function.

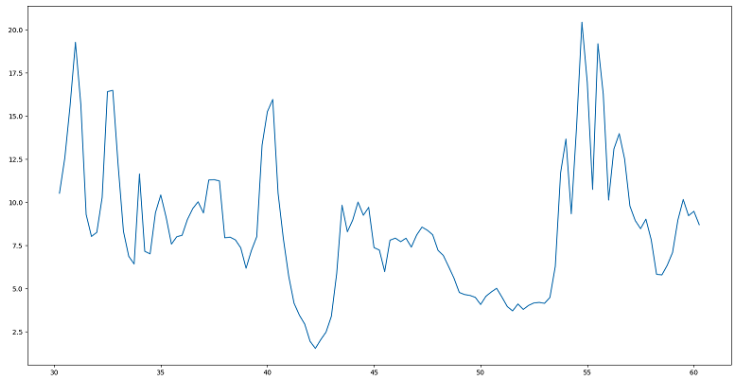


Figure 8: The graph of the *bec* in February of 2011

By visually comparing the bar codes produced by the *bec* and *tbec* calculations for the same month, we see that the atmospheric rivers detected by the *bec* calculation are also detected by the *tbec* calculation. However, the *tbec* calculation does in general produce many more cycles than the *bec* calculation, and some of these cycles can be rather persistent. We wish to find out whether or not the persistent cycles which do not appear in the *bec* calculation correspond to plumes.

Before sifting through the weather data, it is a good idea to compare the two methods using something more rigorous than visual comparison, so that we can pinpoint where the potential plumes or unwanted cycles are. The C++ library Dionysus offers a method for measuring how similar two persistence diagrams (i.e. two sets of bar codes) are to one another. This measure is called the *bottleneck distance*.

Upon reading articles regarding the bottleneck distance, it is apparent that by using the concepts of persistence modules and interleavings as presented in "The Structure and Stability of Persistence Modules" by Chazal, de Silva, Glisse and Oudot (see [Chazal et al.]), one could easily obtain stronger results than is given in other articles (at least the articles my supervisors and I were aware of before starting this thesis). However, in order to prove the desired results for a more general class of persistence diagrams (which includes infinite persistence diagrams), the authors had to define many new concepts, and introduce a lot of novel notation for simplicity.

In order to give the same results for the persistence diagrams which occur in practice (i.e. finite persistence diagrams), many of these concepts are not needed, and we can shorten the process of obtaining important results considerably. Thus, we shall endeavour to simplify the available literature, and prove the results (most notably the Isometry Theorem, which includes the Stability Theorem) provided in Chazal et al. (see [Chazal et al., p. 49]) for finite persistence diagrams only.

4 The Bottleneck Distance and the Interleaving Distance

Now we delve into the more theoretical parts of the thesis. Readers who simply wish to see the comparison of the *bec* and *tbec* calculations without this theoretical detour may skip to section 6.

In this section, we introduce the bottleneck distance, which is the metric we will use to compare the results of the *bec* and *tbec* calculations. The bottleneck distance is a measure of how close two sets of points lying above the diagonal in $\mathbb{R}^2$ are. We relate this to persistent homology through the concept of persistence modules.

4.1 Persistence modules

Definition 4.1. A persistence module over a partially ordered set T (henceforth called a T-persistence module) consists of an indexed family of vector spaces $\{V_t \,|\, t \in T\}$ and linear maps $\{v_s^t \,|\, s,t \in T\}$ for which $v_r^t = v_s^t \circ v_r^s$ whenever $r \leq s \leq t$.

Definition 4.2. A homomorphism Φ between two T-persistence modules $\mathbb{U}$ and $\mathbb{V}$ is a collection of linear maps $\{\phi_t : u_t \to v_t \,|\, t \in T\}$ such that

$$\begin{array}{ccc} U_s & \xrightarrow{u_s^t} & U_t \\ \downarrow{\scriptstyle \phi_s} & & \downarrow{\scriptstyle \phi_t} \\ V_s & \xrightarrow{v_s^t} & V_t \end{array}$$

commutes for all $s \leq t$.

Example 4.3. Let $\mathbb{X}$ be a filtered complex. If homology is taken with field coefficients, then the family: $H_n(X^0,k) \to H_n(X^1,k) \to H_n(X^2,k) \to \cdots$ determine a persistence module (the maps are induced by inclusions).

We now define interval modules. Note that, contrary to the norm, we define interval modules using open intervals only. This is due to that when calculating persistent homology we only concern ourselves with one type of interval. This lack of ambiguity regarding the topology of the intervals will also simplify some proofs for us.

Definition 4.4. An interval module $\mathbb{I}(a, b)$ where (a, b) is an interval in T is a T-persistence module with vector spaces

$$\mathbb{I}_t = \begin{cases} \mathbf{k} & \text{if } t \in (a, b) \\ 0 & \text{if } t \notin (a, b) \end{cases}$$

and maps i_c^d which are the identity if $c, d \in (a, b)$ and zero otherwise.

Definition 4.5. We say that a persistence module is decomposable if it can be decomposed into interval modules, i.e. written as a sum $\mathbb{V} = \bigoplus_j \mathbb{I}(a_j, b_j)$ for some indexing set J. If it can be written as a finite sum of interval modules, we say that it is finitely decomposable.

A theorem by Krull, Remak, Schmidt and Azumaya gives us that such a decomposition is unique up to permutation ([Chazal et al., Theorem 1.3, p. 12]).

As we have seen in Example 4.3, the persistent homology of a cellular complex has the structure of a persistence module, and we know that it can be represented by a sum of intervals (the bar code). Hence, persistent homology can be described by a decomposable interval module. Consider the filtered complex in Example 2.4. If we describe the 0-persistence homology of this complex by a persistence module $\mathbb{V}$, we can write: $\mathbb{V} = \mathbb{I}(1, \infty) \oplus \mathbb{I}(1, 2) \oplus \mathbb{I}(3, 4) \oplus \mathbb{I}(3, 4)$

As with the specific case of persistent homology, the intervals in a decomposable T-persistence module where $T \subset \mathbb{R}$ can be represented by points that lie in the extended half-plane.

Definition 4.6. If $\mathbb{U}$ is a decomposable persistence module over $\mathbb{R}$, that is: $\mathbb{U} = \bigoplus_j \mathbb{I}(a_j, b_j)$ for some indexing set J, we define the persistence diagram of $\mathbb{U}$ to be the multiset $dgm(\mathbb{U}) = \{(a_j, b_j) | j \in J\} - \Delta$ where Δ denotes the diagonal in $\mathbb{R}^2$.

Note that (as before) this is a multiset since intervals in the interval decomposition can occur with multiplicity, and hence so can points in the diagram.

Definition 4.7. Let $\mathbb{U} = \bigoplus_{j \in J} \mathbb{I}(a_j, b_j)$ be a finitely decomposable persistence module. Then the cardinality $|dgm(\mathbb{U})|$ is equal to $|J|$. Remark: since J indexes each individual copy of an interval module, we can also write this cardinality as $\Sigma_{j \in J} m_{\mathbb{U}}(a_j, b_j)$.

4.2 The bottleneck distance

Recall: the l^∞-metric in the plane, which is given by $d^\infty((a,b),(c,d)) = max\{|a-c|, |b-d|\}$.

Henceforth in this section, if a point α in a persistence diagram has multiplicity n, we index the different "instances" of α by writing $\alpha_1, \alpha_2 \cdots \alpha_n$ so we can work with sets instead of multisets.

Definition 4.8. A partial matching between two persistence diagrams $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$ is a collection of pairs $\mathbf{M} \subset dgm(\mathbb{U}) \times dgm(\mathbb{V})$ such that
1) For every $\alpha \in dgm(\mathbb{U})$ there is at most one $\beta \in dgm(\mathbb{V})$ such that $(\alpha, \beta) \in \mathbf{M}$.
2) For every $\beta \in dgm(\mathbb{V})$ there is at most one $\alpha \in dgm(\mathbb{U})$ such that $(\alpha, \beta) \in \mathbf{M}$.

Definition 4.9. We say that a partial matching is a δ-matching if all of the following are true:
1) If $(\alpha, \beta) \in \mathbf{M}$ then $d^\infty(\alpha, \beta) \leq \delta$
2) If $\alpha \in dgm(\mathbb{U})$ is unmatched, then $d^\infty(\alpha, \Delta) \leq \delta$
3) If $\beta \in dgm(\mathbb{V})$ is unmatched, then $d^\infty(\Delta, \beta) \leq \delta$

Definition 4.10. The bottleneck distance between $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$ is defined to be:
$d_b(dgm(\mathbb{U}), dgm(\mathbb{V})) = inf\{\delta \mid \exists \text{ a } \delta\text{-matching between } dgm(\mathbb{U}) \text{ and } dgm(\mathbb{V})\}$

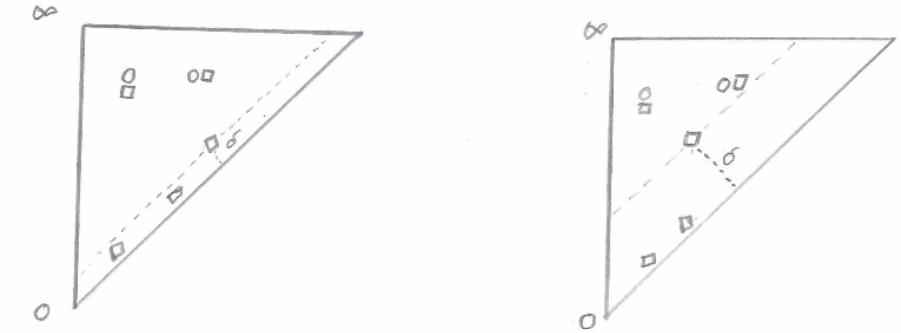


Figure 9: Here we see two instances of persistence diagrams being compared. On the left, we have a small bottleneck distance. On the right, an unmatched square far away from the diagonal gives a large bottleneck distance.

Note that the bottleneck distance is indeed a metric. The only thing that is not obvious is that it satisfies the triangle inequality.

Proposition 4.11. *For any three persistence diagrams A,B,C, the triangle inequality*

$$d_b(A,C) \leq d_b(A,B) + d_b(B,C)$$

holds.

Proof. Let $\delta = \delta_1 + \delta_2$ where M_1 is a δ_1-matching between A and B, and M_2 is a δ_2-matching between B and C. To show that the triangle inequality holds, we must show that there exists a δ matching M between A and C.

Let $M = \{(\alpha,\gamma) \mid \exists \text{ a } \beta \in B \text{ such that } (\alpha,\beta) \in M_1 \text{ and } (\beta,\gamma) \in M_2)\}$.

Now, if $(\alpha,\gamma) \in M$, then $d^\infty(\alpha,\gamma) \leq d^\infty(\alpha,\beta) + d^\infty(\beta,\gamma) \leq \delta_1 + \delta_2 = \delta$.

If $\alpha \in A$ is unmatched in M (the proof is symmetric for γ), then either α is unmatched in M_1, or α is matched with an element $\beta \in B$, in which case β must be unmatched in M_2.
In the first case, we have that $d^\infty(\alpha,\Delta) \leq \delta_1 \leq \delta$.
In the second case, we have that

$$d^\infty(\alpha,\Delta) \leq d^\infty(\alpha,\beta) + \leq d^\infty(\beta,\Delta) \leq \delta_1 + \delta_2 = \delta.$$

Hence, the desired δ-matching exists. □

4.3 The interleaving distance

It might seem intuitive why the bottleneck distance is a useful tool for comparing instances of persistent homology. But it is always useful to be wary of when we think of two related things as "the same". In the next section, we shall prove *the Isometry theorem*, which states that

$$d_b(dgm(\mathbb{U}), dgm(\mathbb{V})) = d_i(\mathbb{U},\mathbb{V})$$

The distance on the right-hand side will be covered shortly. It is called *the interleaving distance* and measures how close two persistence modules are to being isomorphic. What this theorem tells us is that working with the bottleneck distance on persistence diagrams gives us the same result as working directly with modules. Nothing is lost or added in the jump from module to diagram in the sense that the "difference" between two modules is preserved exactly as it was.

We now explain what we mean by an interleaving, and define the interleaving distance.

An isomorphism between two persistence modules is defined in the way one would expect. That is, two persistence modules $\mathbb{U}$ and $\mathbb{V}$ are isomorphic if there are maps $\Phi \in Hom(\mathbb{U}, \mathbb{V})$ and $\Psi \in Hom(\mathbb{V}, \mathbb{U})$ such that $\Psi \circ \Phi = 1_{\mathbb{U}}$ and $\Phi \circ \Psi = 1_{\mathbb{V}}$. However, when working with persistent homology, this criterion is too strict to be the one we use for when we want to think of two modules as essentially "the same". Even if we work with the same dataset, there will at least be slight differences in the output if we build our cellular complexes using different functions. Therefore, a weaker relation is needed.

Definition 4.12. Let $\mathbb{U}, \mathbb{V}$ be $\mathbf{R}$-persistence modules. Let $\delta \in \mathbf{R}$. A homomorphism of degree δ is a collection Φ of linear maps
$\phi_t : U_t \to V_{t+\delta} \; \forall \, t \in \mathbf{R}$
such that

$$\begin{array}{ccc} U_s & \xrightarrow{u_s^t} & U_t \\ \downarrow{\scriptstyle \phi_s} & & \downarrow{\scriptstyle \phi_t} \\ V_{s+\delta} & \xrightarrow{v_s^t} & V_{t+\delta} \end{array}$$

commutes for all $s \leq t$

We write $Hom^{\delta}(\mathbb{U}, \mathbb{V})$ for homomorphisms $\mathbb{U} \to \mathbb{V}$ of degree δ.

Definition 4.13. Let $\delta \geq 0$.
Two persistence modules $\mathbb{U}, \mathbb{V}$ are said to be δ-interleaved if there are maps $\Phi \in Hom^{\delta}(\mathbb{U}, \mathbb{V})$ and $\Psi \in Hom^{\delta}(\mathbb{V}, \mathbb{U})$ such that for any $t \in \mathbb{R}$:
$(\Psi \circ \Phi)_t = u_t^{t+2\delta}$ and $(\Phi \circ \Psi)_t = v_t^{t+2\delta}$.

Example 4.14. Let X be a filtered complex with sublevelsets given by $(X, f)^t = \{x \in X \mid f(x) \leq t\}$, where $f : X \to \mathbb{R}$.
Alternatively, we can use a different function $g : X \to \mathbb{R}$ to build the sublevelsets of X. If $\| f - g \|_{\infty} \leq \delta$, then the corresponding persistence modules obtained by calculating the persistent homology are δ-interleaved.

This is due to the fact that we have inclusions

$$(X, f)^t \subseteq (X, g)^{t+\delta} \subseteq (X, f)^{t+2\delta}$$

and

$$(X, g)^t \subseteq (X, f)^{t+\delta} \subseteq (X, g)^{t+2\delta}$$

which induce interleaving maps of degree δ when we take the persistent homology.

Observe: if two PMs are δ-interleaved, then they are also $(\delta+\epsilon)$-interleaved for all $\epsilon > 0$.

The maps $\Phi' = \Phi \circ 1_U^\epsilon = 1_V^\epsilon \circ \Phi$ and $\Psi' = \Psi \circ 1_V^\epsilon = 1_U^\epsilon \circ \Psi$
give the required interleavings.
We say that they are δ^+-interleaved if they are $(\delta + \epsilon)$-interleaved for all $\delta > 0$.

Definition 4.15. The interleaving distance between two persistence modules $\mathbb{U}, \mathbb{V}$ is defined as
$d_i(\mathbb{U}, \mathbb{V}) = inf\{\delta \mid \mathbb{U}, \mathbb{V} \text{ are } \delta\text{-interleaved}\} = min\{\delta \mid \mathbb{U}, \mathbb{V} \text{ are } \delta^+\text{-interleaved}\}$.
If there is no value δ giving a δ-interleaving between $\mathbb{U}$ and $\mathbb{V}$, then $d_i(\mathbb{U}, \mathbb{V}) = \infty$.

Remark: it is easily checked that the interleaving distance satisfies the triangle inequality, but it is not a true metric in general. This is due to the fact that $d_i(\mathbb{U}, \mathbb{V}) = 0$ does not imply that $\mathbb{U} \cong \mathbb{V}$. For instance, if we allowed the occurence of closed interval modules, the interleaving distance between $\mathbb{I}[a, b]$ and $\mathbb{I}(a, b)$ is equal to zero, but they are obviously not isomorphic as persistence modules.

5 The Isometry theorem

We now arrive at the main theorem assessed in this thesis: the Isometry theorem.

Theorem 5.1. *Let $\mathbb{U}, \mathbb{V}$ be finitely decomposable persistence modules. Then*

$$d_i(\mathbb{U}, \mathbb{V}) = d_b(dgm(\mathbb{U}), dgm(\mathbb{V}))$$

In this thesis, we have chosen to specialize the theorem to finitely decomposable modules. This is due to the fact that these are the only modules that occur when calculating persistent homology (since for all practical applications the filtered complexes must be finite). However, proving the theorem in the general case requires significant effort. So by specializing in this way, we can obtain this important result for all the persistence modules which occur in practice with much less hassle.

The Isometry theorem is split into two parts, the *Stability theorem*:

Theorem 5.2. *Let $\mathbb{U}, \mathbb{V}$ be finitely decomposable persistence modules. Then*

$$d_i(\mathbb{U}, \mathbb{V}) \geq d_b(dgm(\mathbb{U}), dgm(\mathbb{V}))$$

and the *Converse Stability theorem*:

Theorem 5.3. *Let $\mathbb{U}, \mathbb{V}$ be finitely decomposable persistence modules. Then*

$$d_i(\mathbb{U}, \mathbb{V}) \leq d_b(dgm(\mathbb{U}), dgm(\mathbb{V}))$$

5.1 The Stability theorem

We first prove the Stability theorem, and we start by noting that it can also be stated in the following fashion:

Theorem 5.4. *Let $\mathbb{U}, \mathbb{V}$ be finitely decomposable persistence modules which are δ-interleaved. Then there exists a δ-matching between $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$.*

Note that since $\mathbb{U}$ and $\mathbb{V}$ are finitely decomposable, it is easy to see that if there is an η-matching between their respective persistence diagrams for every $\eta > \delta$, then there is also a δ-matching between them. Since there are finitely many points, there is a finite amount of ways to match them. Hence the amount of η-matchings is closed, and so attains its minimum. This gives us that proving the theorem under the weaker condition that $\mathbb{U}$ and $\mathbb{V}$ are δ^+-interleaved implies that we have also proved it under the stronger condition that they are δ-interleaved.

The proof of the Stability theorem requires two lemmas: the Box lemma and the Interpolation lemma.

5.1.1 The Box lemma

Lemma 5.5. *Let $\mathbb{U}, \mathbb{V}$ be δ-interleaved persistence modules. For $a < b \leq c < d$ such that a, b, c, d lie above the diagonal, let R be the rectangle $[a, b] \times [c, d]$ and let R_δ be the thickened rectangle $[a - \delta, b + \delta] \times [c - \delta, d + \delta]$. Then $|dgm(\mathbb{U})|_R| \leq |dgm(\mathbb{V})|_{R_\delta}|$ and $|dgm(\mathbb{V})|_R| \leq |dgm(\mathbb{U})|_{R_\delta}|$.*

Proof. We show that $|dgm(\mathbb{U})|_R| \leq |dgm(\mathbb{V})|_{R_\delta}|$ (the proof of the other inequality is symmetrical).

Due to the δ-interleaving, we may construct the following module:

$$\mathbb{W} = V_{a-\delta} \to U_a \to U_b \to V_{b+\delta} \to V_{c-\delta} \to U_c \to U_d \to V_{d+\delta}$$

Note that all points lying in R_δ (if thought of as intervals) contain the interval $(b + \delta, c - \delta)$, which is represented by the lower right corner of R_δ. Hence, the cardinality of $dgm(\mathbb{V})$ in R_δ (which we have denoted by $|dgm(\mathbb{V})|_{R_\delta}|$) is equal to the sum $\Sigma_{\substack{x \leq b+\delta \\ y \geq c-\delta}} m_{\mathbb{V}}(x, y) \, \forall \, (x, y) \in |dgm(\mathbb{V})|_{R_\delta}|$, which we denote by $m_{\mathbb{V},R_\delta}(b+\delta, c-\delta)$. We will use this notation for the rest of the proof.

We see that $m_{\mathbb{U},R}(b, c) = m_{\mathbb{W},R}(b, c)$ by the construction of $\mathbb{W}$ and the fact that interleaving maps commute with module maps. Also by the construction of $\mathbb{W}$, we have that $m_{\mathbb{V},R_\delta}(b+\delta, c-\delta) = m_{\mathbb{W},R_\delta}(b+\delta, c-\delta)$. Note that $m_{\mathbb{W},R_\delta}(b + \delta, c - \delta)$ is equal to $m_{\mathbb{W},R}(b, c)$ plus some additional terms (given by the other intervals that restrict to $(b + \delta, c - \delta)$).

By omitting said terms, we see that $m_{\mathbb{W},R}(b,c) \leq m_{\mathbb{W},R_\delta}(b+\delta, c-\delta)$.

Combining these observations, we obtain the inequality:

$$|dgm(\mathbb{U})|_R| = m_{\mathbb{U},R}(b,c) = m_{\mathbb{W},R}(b,c) \leq m_{\mathbb{W},R_\delta}(b+\delta,c-\delta) = m_{\mathbb{V},R_\delta}(b+\delta,c-\delta) = |dgm(\mathbb{V})|_{R_\delta}|$$

This gives us that $|dgm(\mathbb{U})|_R| \leq |dgm(\mathbb{V})|_{R_\delta}|$

□

5.1.2 The Interpolation lemma

Having proved one of the two lemmas we need, we turn our attention to the Interpolation lemma. Our proof of the Interpolation lemma uses the idea of shifted diagonals in the plane.

Definition 5.6. For any real number x,we define the diagonal shifted by x as $\Delta_x = \{(a,b) \in \mathbb{R}^2 \mid b - a = 2x\}$

There is an isomorphism $r \to (r-x, r+x)$ between the real line and Δ_x. If we use the standard partial ordering of the plane, we can use this isomorphism to identify persistence modules over $\mathbb{R}$ with persistence modules over Δ_x.

Lemma 5.7. *Let $\mathbb{U}, \mathbb{V}$ be δ-interleaved persistence modules. Then for $x \in [0,\delta]$ there exists a family of persistence modules $\mathbb{U}_x$ such that $\mathbb{U}_0 = \mathbb{U}$ and $\mathbb{U}_\delta = \mathbb{V}$, and such that for every $x, y \in [0,\delta]$, $\mathbb{U}_x$ and $\mathbb{U}_y$ are $|y-x|$-interleaved.*

Proof. The proof consists of four steps:

1) Showing that two persistence modules $\mathbb{U}, \mathbb{V}$ are $|y-x|$ interleaved if and only if there exists a persistence module $\mathbb{W}$ over $\Delta_x \cup \Delta_y$ such that $\mathbb{W}|_{\Delta_x} = \mathbb{U}$ and $\mathbb{W}|_{\Delta_y} = \mathbb{V}$

2) Using step 1 to show that proving the lemma can be reduced to showing that any persistence module $\mathbb{W}$ over $\Delta_0 \cup \Delta_\delta$ extends to a persistence module $\bar{\mathbb{W}}$ over the diagonal strip $\Delta_{[0,\delta]} = \{(a,b) \in \mathbb{R}^2 \mid 0 \leq b-a \leq 2\delta\}$.

3) Defining a candidate for such an extension.

4) Showing that the candidate we defined above is in fact the extension we were after, i.e. that $\bar{\mathbb{W}}|_{\Delta_0} \cong \mathbb{U}$ and $\bar{\mathbb{W}}|_{\Delta_\delta} \cong \mathbb{V}$.

Step 1:

We assume without loss of generality that $x < y$.
Given interleaving maps

$$\phi_t : U_t \to V_{t+y-x}$$

and

$$\psi_t : V_t \to U_{t+y-x}$$

To construct the module $\mathbb{W}$ we need maps $\mathbb{W}|_{\Delta_x} \to \mathbb{W}|_{\Delta_x}$, $\mathbb{W}|_{\Delta_y} \to \mathbb{W}|_{\Delta_y}$, $\mathbb{W}|_{\Delta_x} \to \mathbb{W}|_{\Delta_y}$ and $\mathbb{W}|_{\Delta_y} \to \mathbb{W}|_{\Delta_x}$, with the composition rule determined by the partial ordering on the plane. However, all of these maps are already given by the modules $\mathbb{U}, \mathbb{V}$ and the interleaving maps. The first two cases are obvious. For the last two, note that if $\mathbb{W}|_{\Delta_x} = \mathbb{U}$ and $\mathbb{W}|_{\Delta_y} = \mathbb{V}$, we can rewrite the interleaving maps in the following manner:

$$\phi_t : U_t = W_{(t-x,t+x)} \to W_{(t-x,t+2y-x)} = V_{t+y-x}$$

$$\psi_t : V_t = W_{(t-y,t+y)} \to W_{(t+y-2x,t+y)} = U_{t+y-x}$$

this gives us the maps needed to define $\mathbb{W}$ (it is easy to check that these maps agree with the composition rules for module and interleaving maps).

Assume we are given a module $\mathbb{W}$ on $\Delta_x \cup \Delta_y$ with $\mathbb{W}|_{\Delta_x} = \mathbb{U}$ and $\mathbb{W}|_{\Delta_y} = \mathbb{V}$. The module maps will be of the form $\omega_R^S : W_R \to W_S$ for $R \leq S$, satisfying the composition rule $\omega_R^S \circ \omega_T^R = \omega_T^S$ for $T \leq R \leq S$. We need to check that the cases of ω_R^S where $R \in \Delta_x$ and $S \in \Delta_y$ or vice versa give us the desired interleaving maps between $\mathbb{U}$ and $\mathbb{V}$. Assuming $R \in \Delta_x$ and $S \in \Delta_y$, then by taking $R = t$ and $S = t + y - x$ we obtain ϕ_t through the rewriting we did above. Similarly, we obtain ψ_t by starting with $R \in \Delta_y$ and $S \in \Delta_x$. It is easily checked that

$$\psi_{t+y-x} \circ \phi_t = u_t^{t+2(y-x)}$$

and

$$\phi_{t+y-x} \circ \psi_t = v_t^{t+2(y-x)}$$

as required.

Step 2:

By assumption, $\mathbb{U}, \mathbb{V}$ are δ-interleaved, which by step 1 implies that there exists a persistence module $\mathbb{W}$ over $\Delta_0 \cup \Delta_\delta$ such that $\mathbb{W}|_{\Delta_0} = \mathbb{U}$ and $\mathbb{W}|_{\Delta_\delta} = \mathbb{V}$. If we show that $\mathbb{W}$ extends to a persistence module $\bar{\mathbb{W}}$ over the diagonal strip $\Delta_{[0,\delta]}$, then for each pair $x, y \in [0, \delta]$ the restriction of $\bar{\mathbb{W}}$ to $\Delta_x \cup \Delta_y$ yields us (again, by step 1) $|y-x|$-interleaved persistence modules $\mathbb{U}_x$ and $\mathbb{U}_y$.

Step 3:

We wish to work with a module $\mathbb{W}$ defined over $\Delta_{-1} \cup \Delta_1$ instead of over $\Delta_0 \cup \Delta_\delta$, as this will make the proof more symmetrical and easy to follow. This poses no problem, as it is simply a matter of translation and rescaling in the plane.

Assume we are given such a persistence module $\mathbb{W}$ defined over $\Delta_{-1} \cup \Delta_1$. Since $\mathbb{W}|_{\Delta_{-1}} = \mathbb{U}$ and $\mathbb{W}|_{\Delta_1} = \mathbb{V}$, then by step 1 we have interleaving maps $\Phi \in Hom^2(\mathbb{U}, \mathbb{V})$ and $\Psi \in Hom^2(\mathbb{V}, \mathbb{U})$.

Note that since $\mathbb{U}$ and $\mathbb{V}$ are defined over Δ_{-1} and Δ_1 respectively, we have that $U_t = W(t+1, t-1)$ and $V_t = W(t-1, t+1)$.

We extend $\mathbb{W}$ to a module $\bar{\mathbb{W}}$ defined over the strip $\Delta_{[-1,1]}$ by constructing four persistence modules over $\mathbb{R}^2$ as well as module maps between them. The persistence modules are as follows:

$\mathbb{A}$: given by $A_{(p,q)} = U_{p-1}$ and $a^{(r,s)}_{(p,q)} = u^{r-1}_{p-1}$

$\mathbb{B}$: given by $B_{(p,q)} = V_{q-1}$ and $b^{(r,s)}_{(p,q)} = v^{s-1}_{q-1}$

$\mathbb{C}$: given by $C_{(p,q)} = U_{q+1}$ and $c^{(r,s)}_{(p,q)} = u^{s+1}_{q+1}$

$\mathbb{D}$: given by $D_{(p,q)} = V_{p+1}$ and $d^{(r,s)}_{(p,q)} = v^{r+1}_{p+1}$

We now define the module maps between them at each point (p, q):

$\mathbf{1}_{\mathbb{U}_{(p,q)}} : A_{(p,q)} \to C_{(p,q)}$ given by $u^{q+1}_{p-1} : U_{p-1} \to U_{q+1}$

$\mathbf{1}_{\mathbb{V}_{(p,q)}} : B_{(p,q)} \to D_{(p,q)}$ given by $v^{p+1}_{q-1} : V_{q-1} \to V_{p+1}$

$\Phi_{(p,q)} : A_{(p,q)} \to D_{(p,q)}$ given by $\phi_{p-1} : U_{p-1} \to V_{p+1}$

$\Psi_{(p,q)} : B_{(p,q)} \to C_{(p,q)}$ given by $\psi_{q-1} : V_{q-1} \to U_{q+1}$

Note that while Φ and Ψ are defined over the whole plane, $\mathbf{1}_{\mathbb{U}}$ and $\mathbf{1}_{\mathbb{V}}$ are only defined where $p - 1 \leq q + 1$ and $q - 1 \leq p + 1$ respectively. Hence all four of the module maps are defined over the diagonal strip $\Delta_{[-1,1]} = \{(p, q) \in \mathbb{R}^2 \mid -2 \leq q - p \leq 2\}$. This means that the matrix:

$$\Omega = \begin{bmatrix} \mathbf{1}_{\mathbb{U}} & \Psi \\ \Phi & \mathbf{1}_{\mathbb{V}} \end{bmatrix}$$

is also defined over $\Delta_{[-1,1]}$. We claim that $\bar{\mathbb{W}} = im(\Omega)$ is the extension we seek.

Step 4:

What is left to show is that $\bar{\mathbb{W}}|_{\Delta_{-1}} \cong \mathbb{U}$ and $\bar{\mathbb{W}}|_{\Delta_1} \cong \mathbb{V}$. We show first the former and then the latter.

On $\Delta_{-1} = \{(t+1, t-1)\}$ we have that:

$\mathbf{1}_{\mathbb{U}(t+1,t-1)} = u_t^t$, $\mathbf{1}_{\mathbb{V}(t+1,t-1)} = v_{t-2}^{t+2}$, $\Phi_{(t+1,t-1)} = \phi_t$ and $\Psi_{(t+1,t-1)} = \psi_{t-2}$.

Hence we have that:

$$\Omega|_{\Delta_{-1}} = \begin{bmatrix} u_t^t & \psi_{t-2} \\ \phi_t & v_{t-2}^{t+2} \end{bmatrix}$$

This matrix can be factorized as:

$$\Omega|_{\Delta_{-1}} = \begin{bmatrix} u_t^t \\ \phi_t \end{bmatrix} \begin{bmatrix} u_t^t & \psi_{t-2} \end{bmatrix}$$

For any point $(p, q) = (t+1, t-1)$, this factorization provides us with the following scenario:

$$U_t \bigoplus V_{t-2} \to U_t \to U_t \bigoplus V_{t+2}$$

where the map on the left is a surjection and the map on the right is an injection. Hence $\mathbb{U} \cong im(\Omega|_{\Delta_{-1}})$.

The proof that $\bar{\mathbb{W}}|_{\Delta_1} \cong \mathbb{V}$ follows the exact same procedure, but we write it out for the sake of completeness.

On $\Delta_1 = \{(t-1, t+1)\}$ we have that:

$\mathbf{1}_{\mathbb{U}(t-1,t+1)} = u_{t-2}^{t+2}$, $\mathbf{1}_{\mathbb{V}(t-1,t+1)} = v_t^t$, $\Phi_{(t-1,t+1)} = \phi_{t-2}$ and $\Psi_{(t-1,t+1)} = \psi_t$.

Hence we have the matrix:

$$\Omega|_{\Delta_1} = \begin{bmatrix} u_{t-2}^{t+2} & \psi_t \\ \phi_{t-2} & v_t^t \end{bmatrix}$$

which we can factorize as:

$$\Omega|_{\Delta_1} = \begin{bmatrix} \psi_t \\ v_t^t \end{bmatrix} \begin{bmatrix} \phi_{t-2} & v_t^t \end{bmatrix}$$

Now, given any point $(p, q) = (t-1, t+1)$, we have the factorization:

$$U_{t-2} \bigoplus V_t \to V_t \to U_{t+2} \bigoplus V_t$$

giving that $\mathbb{V} \cong im(\Omega\,|_{\Delta_1})$.

□

5.1.3 The proof

Finally, we have all the ingredients needed for our proof of the Stability theorem. Once again, we restate the theorem to make it easier to prove. Note that $\mathbb{U}$ and $\mathbb{V}$ being δ-interleaved implies the assumptions of the following restatement by the preceding lemmas.

Theorem 5.8. *Let $\{\mathbb{U}_x \,|\, x \in [0, \delta]\}$ be a one-parameter family of finitely decomposable persistence modules with $\mathbb{U}_0 = \mathbb{U}$ and $\mathbb{U}_\delta = \mathbb{V}$. Assume that for all $x, y \in [0, \delta], x \neq y$, the box inequality*

$$|dgm(\mathbb{U}_x)|_R| \leq |dgm(\mathbb{U}_y)|_{R^{|y-x|}}|$$

holds for all rectangles R whose $|y-x|$-thickening is in $\hbar$ (the extended half-plane). Then there exists a δ-matching between $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$.

Proof. The proof consists of 3 parts. First, we show that given an α far enough away from the diagonal, then there exists a β that is a potential match. Then, we show that there is *at most* one such β, so that we have a matching. Finally, we need to show that the two previous steps imply that $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$ are δ-matched.

1) If $\alpha \in dgm(\mathbb{U}_x)$ and $d^\infty(\alpha, \Delta) > |y-x|$, then $\exists\, \beta \in dgm(\mathbb{U}_y)$ with $d^\infty(\alpha, \beta) < |y-x|$. Symmetrically, if $\beta \in dgm(\mathbb{U}_y)$ and $d^\infty(\beta, \Delta) > |y-x|$, then $\exists\, \alpha \in dgm(\mathbb{U}_x)$ with $d^\infty(\alpha, \beta) < |y-x|$.

We prove only the first case, as the second is symmetrical.

Given $\alpha \in dgm(\mathbb{U}_x)$ such that $d^\infty(\alpha, \Delta) > |y-x|$, let $\epsilon > 0$ be small enough so that $d^\infty(\alpha, \Delta) > |y-x| + \epsilon$ is satisfied. We define the ϵ-square of α to be $\alpha^\epsilon = \{\gamma \in \hbar \,|\, d^\infty(\alpha, \gamma) \leq \epsilon\}$ (this is a square due to the l^∞ metric). By the box inequality of the Theorem, we have:

$$1 \leq |dgm(\mathbb{U}_x)|_{\alpha^\epsilon}| \leq |dgm(\mathbb{U}_y)|_{\alpha^{|y-x|+\epsilon}}|$$

Hence, there is at least one point of $dgm(\mathbb{U}_y)$ in the square $\alpha^{|y-x|+\epsilon}$. Clearly, the same inequality also holds for all $\epsilon' < \epsilon$. Since $dgm(\mathbb{U}_y)$ is finite, this means that there is at least one point $\beta \in dgm(\mathbb{U}_y)$ in the square $\alpha^{|y-x|}$.

2) Now we prove that there is at most one such β that is matched with each α.

Suppose we are given an $x \in [0, \delta]$. We index all the distinct points in $dgm(\mathbb{U}_x)$ by $\alpha_1, \ldots, \alpha_k$, each point with its respective multiplicity $m_1, \ldots, m_k$. For every such x we define a function $\rho(x)$ satisfying:

$$0 < \rho(x) < \frac{1}{2} d^\infty(\alpha_i, \alpha_j)$$

for $i \neq j$, and

$$0 < \rho(x) < \frac{1}{2} d^\infty(\alpha_i, \Delta)$$

for all i.

We claim that if $y \in [0, \delta]$ and $\rho(x) > |y - x|$, then $\mathbb{U}_x, \mathbb{U}_y$ are $|y - x|$-matched.

Let

$$\hbar^{|y-x|} = \{\alpha \in \hbar \,|\, d^\infty(\alpha, \Delta) \leq |y - x|\}$$

That is, $\hbar^{|y-x|}$ denotes all points in the extended half-plane that lie within some distance $|y - x|$ from the diagonal. From part 1 of the proof, it is clear that all points in $dgm(\mathbb{U}_y)$ lie in the union $\hbar^{|y-x|} \cup \alpha_1^{|y-x|} \cup \ldots \cup \alpha_k^{|y-x|}$. Due to how we have chosen $\rho(x)$, this union is disjoint. Also because of how we chose $\rho(x)$, for $\epsilon > 0$ small enough so that $2|y - x| + \epsilon < 2\rho(x) + \epsilon$ we have that

$$|dgm(\mathbb{U}_x)|_{\alpha_i^\epsilon}| = |dgm(\mathbb{U}_x)|_{\alpha_i^{2|y-x|+\epsilon}}|$$

From the box inequality we now obtain the following:

$$m_i = |dgm(\mathbb{U}_x)|_{\alpha_i^\epsilon}| \leq |dgm(\mathbb{U}_y)|_{\alpha_i^{|y-x|+\epsilon}}| \leq |dgm(\mathbb{U}_x)|_{\alpha_i^{2|y-x|+\epsilon}}| = m_i$$

Which gives us that $|dgm(\mathbb{U}_y)|_{\alpha_i^{|y-x|+\epsilon}}| = m_i$. Hence, counting multiplicity, there is exactly m_i points of $dgm(\mathbb{U}_y)$ in the square $\alpha_i^{|y-x|+\epsilon}$. This means that we can match each of the m_i copies of $\alpha_i \in dgm(\mathbb{U}_x)$ with a point in $dgm(\mathbb{U}_y)$, defining a $|y - x|$-matching.

3) As we have seen in the proof of Proposition 4.10 (the triangle inequality for the bottleneck distance), if $dgm(\mathbb{U}_0), dgm(\mathbb{U}_x)$ are x-matched, and $dgm(\mathbb{U}_x), dgm(\mathbb{U}_y)$ are $|y - x|$-matched, then $dgm(\mathbb{U}_0), dgm(\mathbb{U}_y)$ are y-matched.

Let $r = sup\{x \in [0,\delta] \,|\, dgm(\mathbb{U}_0)$ and $dgm(\mathbb{U}_x)$ are x-matched$\}$. Now choose an r' such that $dgm(\mathbb{U}_0), dgm(\mathbb{U}_{r'})$ are r'-matched and $\rho(r) \geq r - r'$. By the triangle inequality and step 2 of this proof, $dgm(\mathbb{U}_0)$ and $dgm(\mathbb{U}_r)$ must be r-matched. If $r < \delta$, then by the same logic there would exist an $\tilde{r} > r$ such that $dgm(\mathbb{U}_0)$ and $dgm(\mathbb{U}_{\tilde{r}})$ are $\tilde{r}$-matched. This cannot be the case, since it contradicts the definition of r. Hence $dgm(\mathbb{U}_0) = dgm(\mathbb{U})$ and $dgm(\mathbb{U}_\delta) = dgm(\mathbb{V})$ are δ-matched.

□

5.2 The Converse Stability theorem

We now move on to prove the Converse Stability theorem.

Proposition 5.9. *Let (a,b) and (c,d) be intervals (they may be infinite). Let $\mathbb{U} = \mathbb{I}(a,b)$ and $\mathbb{U} = \mathbb{I}(c,d)$ be interval modules. Then $d_i(\mathbb{U},\mathbb{V}) \leq d^\infty((a,b),(c,d))$*

Proof. We consider the case where a, b, d, c are all finite (the infinite cases are similar). We must show that if $\delta > max(|a-c|, |b-d|)$ then $\mathbb{U}, \mathbb{V}$ are δ-interleaved. To do this, we first define systems of linear maps

$$\Phi = \{\phi_t : U_t \to V_{t+\delta}\}$$

and

$$\Psi = \{\psi_t : V_t \to U_{t+\delta}\}$$

and show that they are homomorphisms of persistence modules.

For Φ to be a homomorphism of persistence modules, the diagram

$$\begin{array}{ccc} U_t & \longrightarrow & U_{t+\epsilon} \\ \downarrow{\scriptstyle \phi_t} & & \downarrow{\scriptstyle \phi_{t+\epsilon}} \\ V_{t+\delta} & \longrightarrow & V_{t+\delta+\epsilon} \end{array}$$

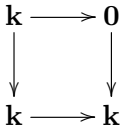
must commute for all t and for all $\epsilon > 0$ (the case for Ψ is symmetric).

Since $\mathbb{U}, \mathbb{V}$ are interval modules, the vector spaces are ether a copy of the ground field $\mathbf{k}$ or are 0. Hence, the only hypothetical cases where the diagram does not commute would be:

1)

$$\begin{array}{ccc} \mathbf{k} & \longrightarrow & \mathbf{k} \\ \downarrow & & \downarrow \\ \mathbf{0} & \longrightarrow & \mathbf{k} \end{array}$$

and 2)

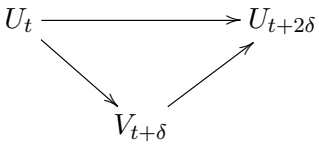


We show that these cannot occur.

1) In this case, $a \leq t$ since $U_t = \mathbf{k}$, and $t + \delta \leq c$ since $V_{t+\delta} = 0$ but $V_{t+\delta+\epsilon} = \mathbf{k}$. This is impossible since $\delta > c - a$

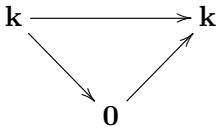
2) Here $b \leq t + \eta$ since $U_{t+\epsilon} = 0$ but $U_t = \mathbf{k}$, and $t + \delta + \epsilon \leq d$. This is also impossible since $\delta > d - c$.

To show that $\Phi \circ \Psi = (1_{\mathbb{U}})^{2\delta}$ we show that the diagram



commutes $\forall\, t$ (the proof is symmetric for $\Psi \circ \Phi$).

The only case where it does not commute is the case



Again we show that this cannot occur. The top row implies that $a \leq t$ and $t + 2\delta \leq b$. Since $\delta > c - a$ and $\delta > b - d$ we obtain the inequality

$$c < \delta + a \leq t + \delta \leq b - \delta < d$$

but this implies that $V_{t+\delta} = \mathbf{k}$.

□

Proposition 5.10. *Let* $(\mathbb{U}_\alpha \,|\, \alpha \in A)$ *and* $(\mathbb{V}_\alpha \,|\, \alpha \in A)$ *be families of persistence modules, and let*

$$\mathbb{U} = \bigoplus_{\alpha \in A} \mathbb{U}_\alpha$$

and

$$\mathbb{V} = \bigoplus_{\alpha \in A} \mathbb{V}_\alpha$$

Then $d_i(\mathbb{U}, \mathbb{V}) \leq sup(d_i(\mathbb{U}_\alpha, \mathbb{V}_\alpha) \,|\, \alpha \in A)$.

Proof. Given δ-interleavings for Φ_α and Ψ_α for each pair $\mathbb{U}_\alpha, \mathbb{V}_\alpha$, then the direct sum maps $\oplus\Phi_\alpha$ and $\oplus\Psi_\alpha$ give an interleaving of $\mathbb{U}$ and $\mathbb{V}$. Hence every upper bound for $\{d_i(\mathbb{U}_\alpha, \mathbb{V}_\alpha)\}_\alpha$ is greater than or equal to $d_i(\mathbb{U}, \mathbb{V})$. □

Proposition 5.11. *Let $\mathbb{U} = \mathbb{I}(a, b)$ be an interval module corresponding to the interval (a, b), and let $\mathbf{0}$ denote the zero persistence module. Then $d_i(\mathbb{U}, \mathbf{0}) = (b - a)/2$.*

Proof. Let $\delta \geq 0$. We have a δ-interleaving between $\mathbb{U}$ and the zero module when the interleaving maps are zero. In one direction this is trivial, so we only need to check that when $\Phi : \mathbb{U} \to \mathbf{0}$ and $\Psi : \mathbf{0} \to \mathbb{U}$, then $\Psi\Phi = (1_{\mathbb{U}})^{2\delta} = 0$. This holds when $\delta > (b - a)/2$ and fails when $\delta < (b - a)/2$ (a needs to be shifted past b). □

We now use these propositions to prove the converse stability theorem for decomposable persistence modules.

Proof. Let $\mathbf{M}$ be a δ-matching between $dgm(\mathbb{U})$ and $dgm(\mathbb{V})$. Each point in the diagrams correspond to an interval summand in its respective persistence module, so from $\mathbf{M}$ we obtain a pairing between the interval summands of $\mathbb{U}$ and $\mathbb{V}$. We say that two paired summands are matched.

We can therefore rewrite $\mathbb{U}$ and $\mathbb{V}$ in the form

$$\mathbb{U} = \bigoplus_{\alpha \in A} \mathbb{U}_\alpha$$

and

$$\mathbb{V} = \bigoplus_{\alpha \in A} \mathbb{V}_\alpha$$

such that each pair is one of the following:

1) A pair of matched intervals.

2) $\mathbb{U}_\alpha$ is an unmatched interval and $\mathbb{V}_\alpha = 0$, or $\mathbb{V}_\alpha$ is an unmatched interval and $\mathbb{U}_\alpha = 0$.

For case 1, we have that

$$d_i(\mathbb{U}_\alpha, \mathbb{V}_\alpha) \leq d^\infty((a, b), (c, d)) \leq \delta$$

by Proposition 5.9 and the definition of δ-matching.

For case 2, Proposition 5.11 gives us that

$$d_i(\mathbb{U}_\alpha, \mathbf{0}) = (b - a)/2 = d^\infty((a, b), \Delta) \leq \delta$$

. Now, by Proposition 5.10 we have that $d_i(\mathbb{U}, \mathbb{V}) \leq \delta$, and thus the proof is complete. □

This concludes our proof of the Isometry theorem.

6 Results of the first comparison

We now continue where we left off at section 3.

As these were the months which Alfsvåg considered in his thesis, we have chosen to compare the bottleneck distances of the *bec* and *tbec* algorithms for the entirety of 2011 and 2012 except for January, as well as for June from 2000 and onwards (comparing the results of the same month over many different years is useful because it allows us to disregard seasonal variance, as we shall soon see). The Dionysus method for computing bottleneck distances yields us three values: the actual bottleneck distance (which we have rounded to three decimal places), the number of cycles discovered by the *bec* algorithm, and the number of cycles discovered by the *tbec* algorithm.

For example, for February of 2011 the output is as follows:
Amount of cycles for bec: 24
Amount of cycles for tbec: 62
Bottleneck distance: 2.270

For June of the same year, we obtain:
Amount of cycles for bec: 24
Amount of cycles for tbec: 1108
Bottleneck distance: 5.458

As we can see, the number of cycles discovered by the *bec* algorithm is the same both for February and June, but the number of cycles discovered by the *tbec* algorithm is much greater for June than for February. We can also see that the bottleneck distance between the *bec* and *tbec* diagrams is much larger for June than for February. This leaves us with two questions: is the latter related to the former? And if so, what might be the cause of the discrepancy in the amount of cycles? Considering the visualization of the *tcw* data for February and June (provided by the ERA-Interim project), we see that the "band" of humidity at the equator appears to stretch much further north during June than during February.

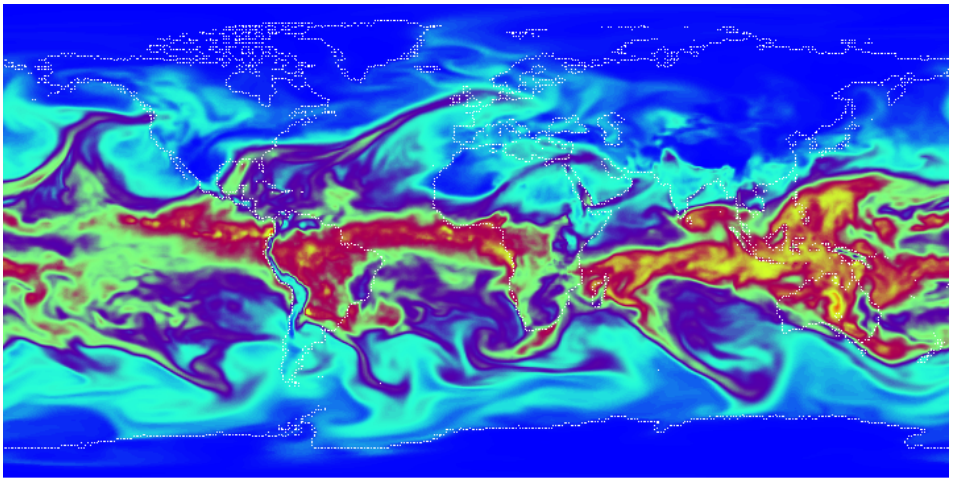


Figure 10: A typical "winter scene" during February 2011. Most of the activity is concentrated near the equator save for arms of humidity (rivers) stretching out towards the north.

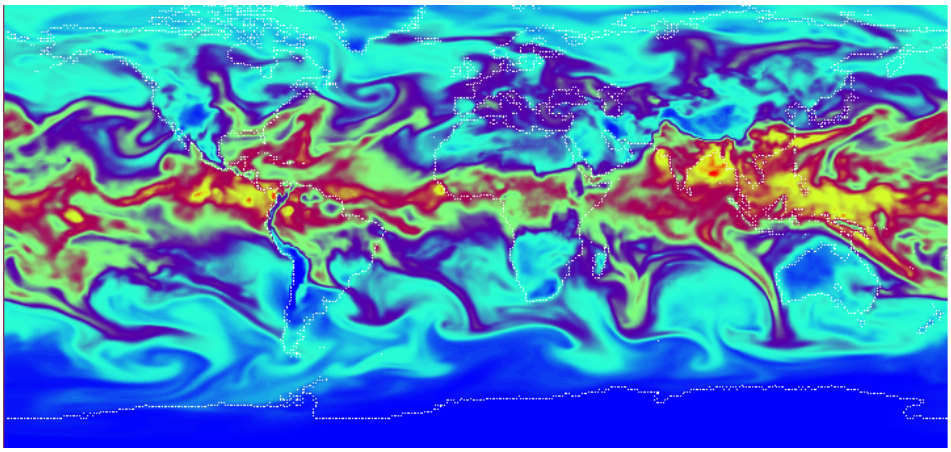


Figure 11: A frame from June 2011. As we can see, while the areas close to the equator are still the most humid, the areas to the north are now much more humid in general. This is typical for the summer months.

Considering the visualizations for the entirety of 2011 and 2012 apart from January, as well as for June from 2000 and on, we see that the situations seen above are part of a greater trend. In general, the humidity is concentrated near the equator in the earlier and later parts of the year (save for atmospheric rivers and plumes), while during the summer months, this humidity "expands" to cover areas further to the north. Looking at the bottleneck calculations, we see that this trend appears to be reflected in two-out-of-three output values. The amount of cycles discovered by the

bec algorithm is consistent throughout the year - with every month having between 21-32 cycles and with no apparent seasonal difference - whereas the *tbec* algorithm generally discovers a lot more cycles in the summer months. In addition, the bottleneck distance is generally higher in the summer months as well, even when no plumes appear to be present (one would expect an increase in plumes during the summer due to the generally high humidity, but this often seems to be difficult to confirm visually due to the sheer amount of activity).

Our hypothesis is therefore the following: the increased "activity" of humidity during the summer months causes the *tbec* algorithm to detect unwanted cycles. Such "fake" cycles may of course occur at any time, but they are especially likely to occur during summer, and they are responsible for the explosive increase in the amount of *tbec* cycles generally occurring at this time. We also hypothesize that some of these unwanted cycles may be persistent, hence they are also responsible for the larger bottleneck distances that often accompany this increase in *tbec* cycles.

Imagine a pincer-shaped region of humidity over the Atlantic, closing in on itself and surrounding a drier region. Now imagine that after some time, this "pincer" opens up again. In the *tbec* algorithm, which is based on a three-dimensional top-down filtration (as we discussed in section 3), this would appear as a cylindrically-shaped region of humidity, potentially causing paths from the equator to Bergen to split into two paths, thereby creating an unwanted cycle.

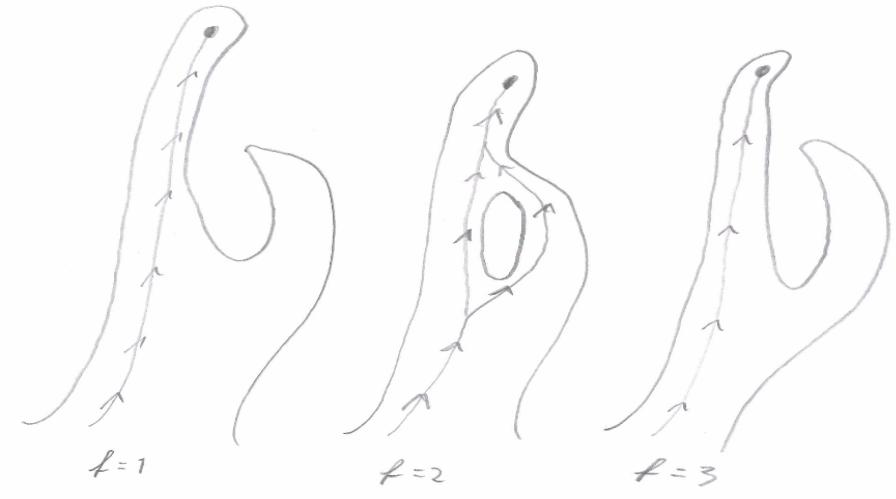


Figure 12: A "pincer" as described above.

This example illustrates what we fear might be a problem with the *tbec* algorithm. To confirm this manually, we look through the ERA-Interim visualizations of the *tcw* data to see when large bottleneck distances are caused by plumes, and when they might be caused by phenomena of the type we just discussed. We have looked at some months where the bottleneck distance is unusually large, as well as summer months where there are clear instances of the pincer-like phenomenon we discussed above (which will from now on be referred to simply as "pincers"). We refer to Appendix B for a complete list of calculated bottleneck distances. The regions we discuss above the pictures are highlighted by green circles.

June 2001
Amount of cycles for bec: 26
Amount of cycles for tbec: 562
Bottleneck distance: 9.403

This is the largest bottleneck distance we have calculated. There is a clear plume here which we have marked as a possible culprit, though it is difficult to tell how strongly it hits Bergen. Even if it is rather weak when it hits Bergen, it appears very "distinct" in the sense that it does not appear to be a part of a river which connects to Bergen at a lower filtration degree, meaning that it is not unthinkable for it to be the cause of this large bottleneck distance.

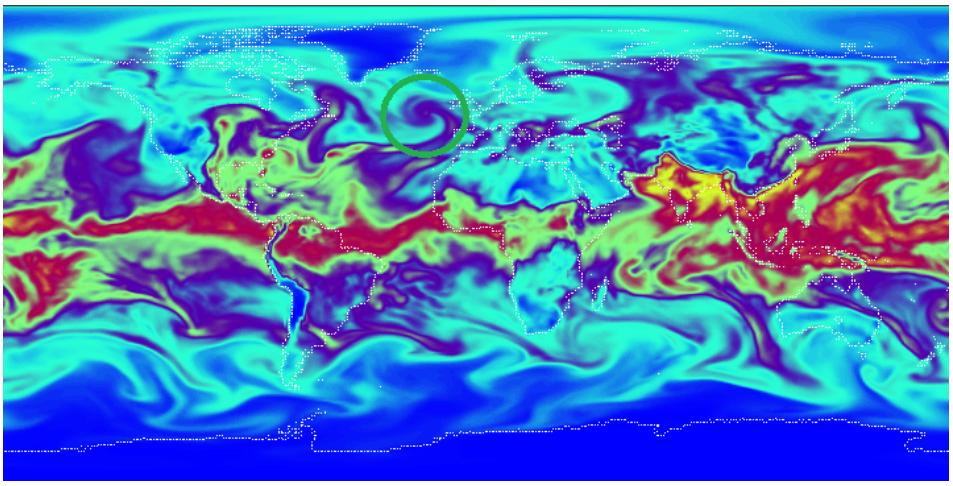
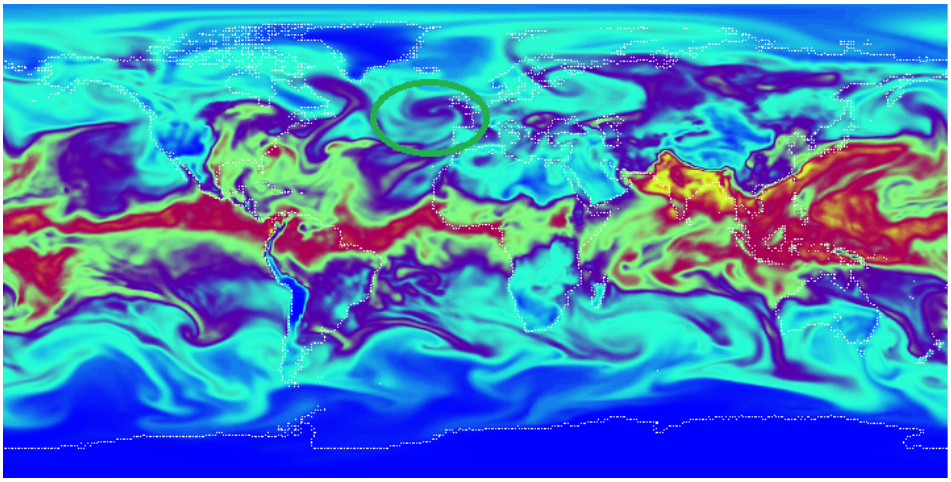
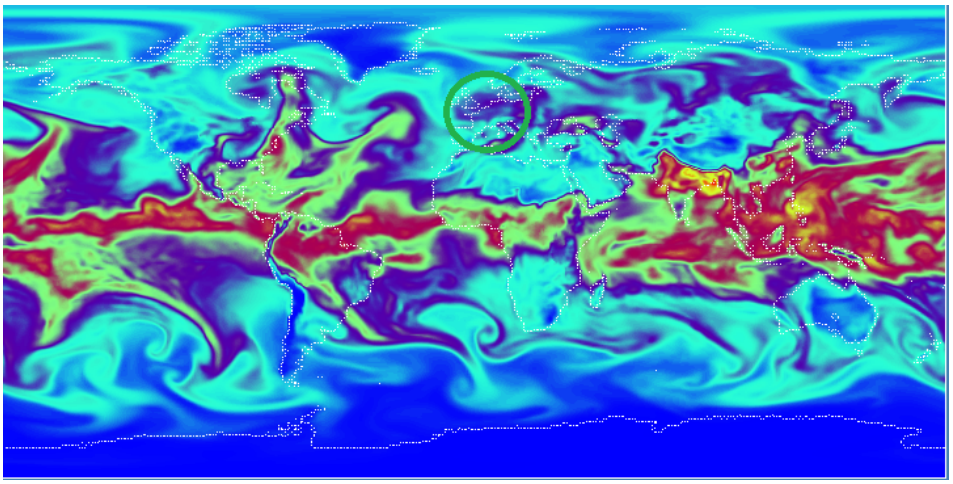


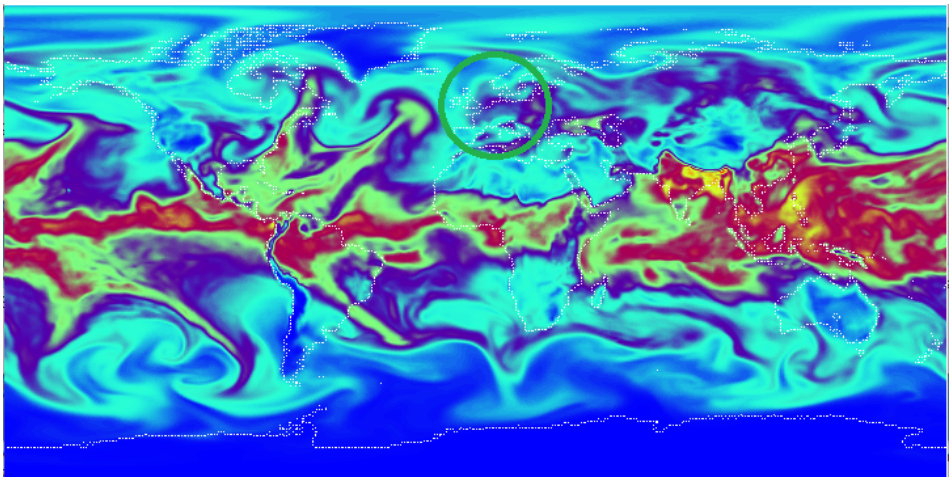
Figure 13: Frame 657 of 2001



Frame 659 of 2001



Frame 669 of 2001



Frame 671 of 2001

June 2002
Amount of cycles for bec: 29
Amount of cycles for tbec: 549
Bottleneck distance: 5.869

While not an unusually large bottleneck distance for a summer month, we do see a clear example of the phenomenon we fear leads to generally large bottleneck distances during the summertime. A relatively dry region is encircled by humidity, and after a while this pincer opens up again towards the south-west. We see that the humid region is attached to Bergen in the third and fourth frame.

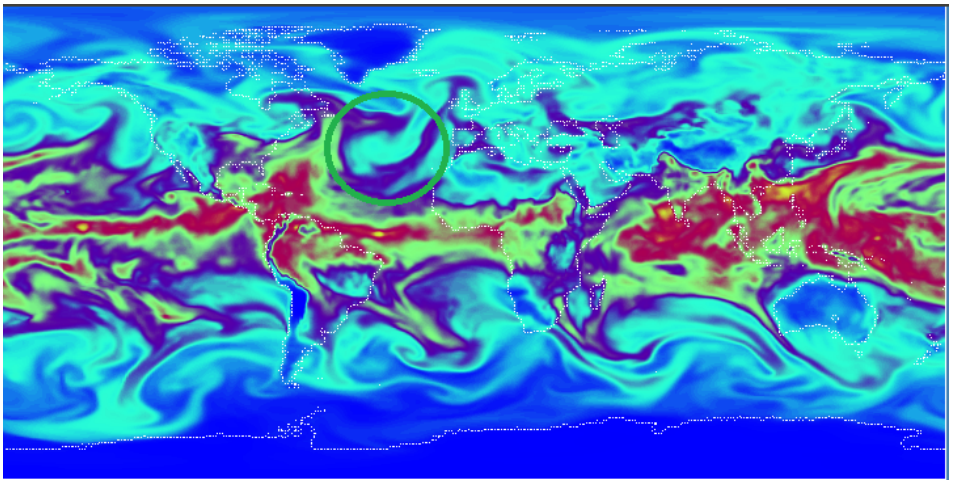
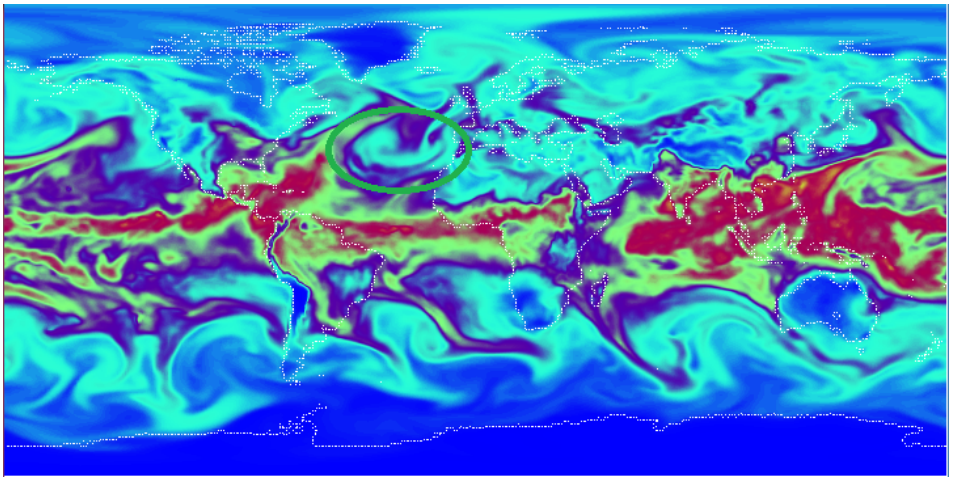
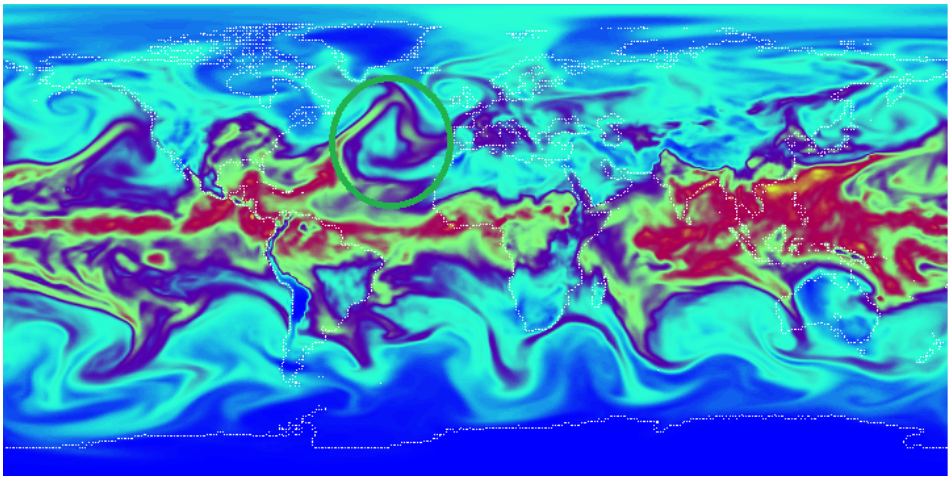


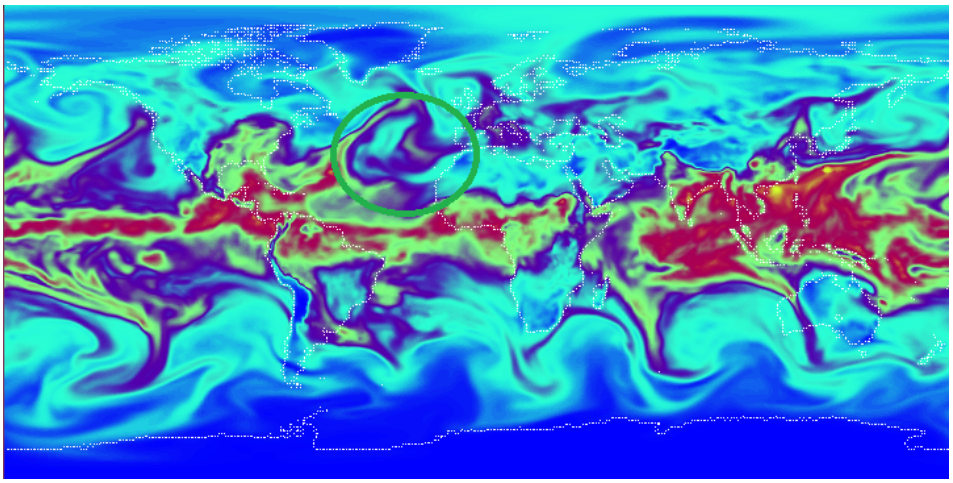
Figure 14: Frame 607 of 2002



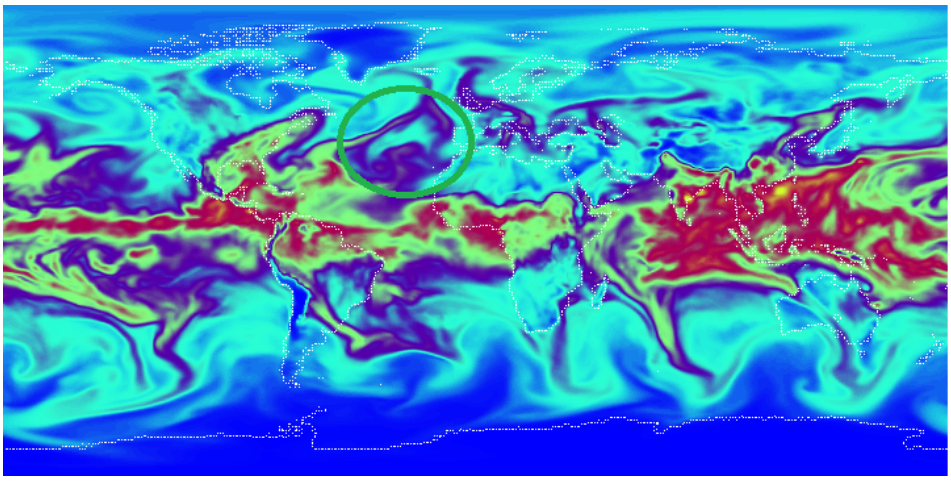
Frame 612 of 2002



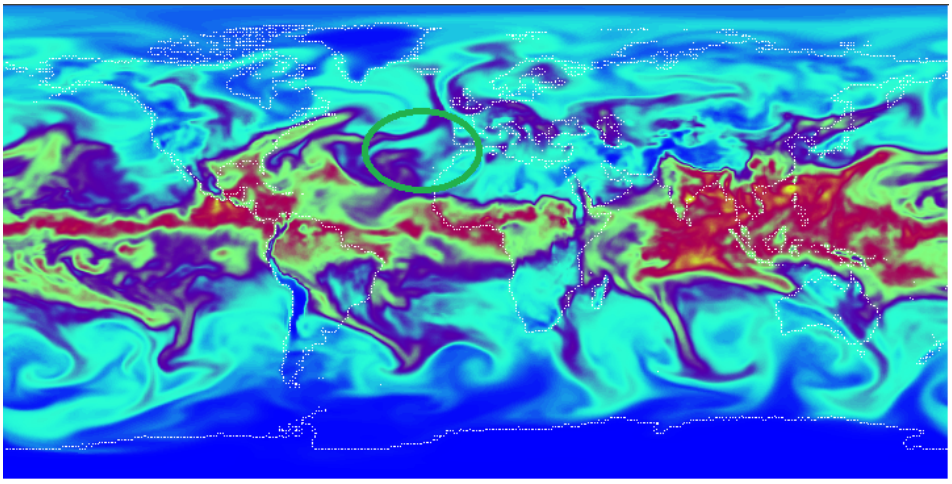
Frame 619 of 2002



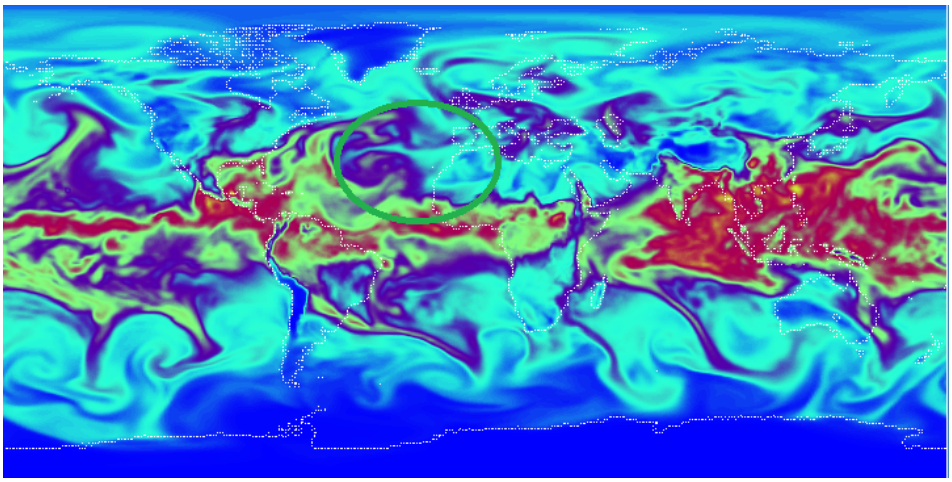
Frame 621 of 2002



Frame 625 of 2002



Frame 628 of 2002



Frame 632 of 2002

June 2003
Amount of cycles for bec: 27
Amount of cycles for tbec: 778
Bottleneck distance: 6.258

It is a bit harder to spot than the similar occurrence in June 2002, but here we see a small, dry region being encircled by a much more humid one. This pincer eventually opens up towards the north. The humid region is most clearly connected to Bergen in the second frame.

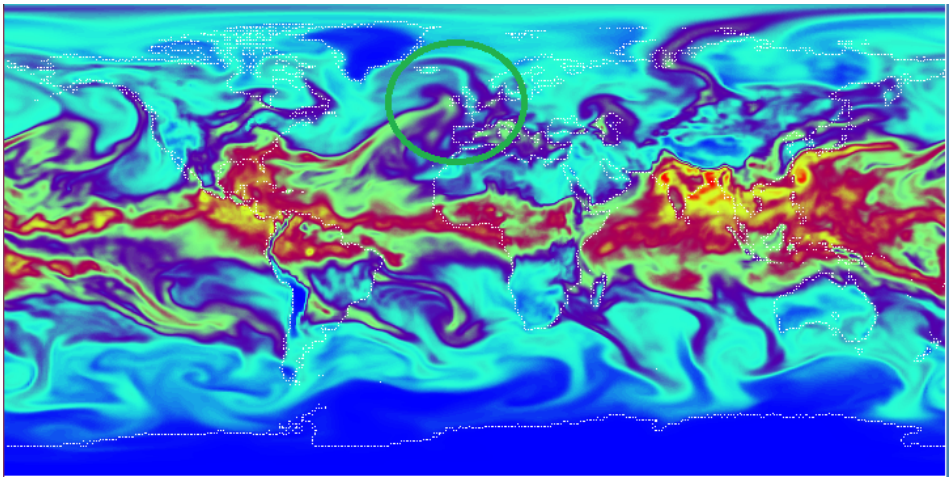
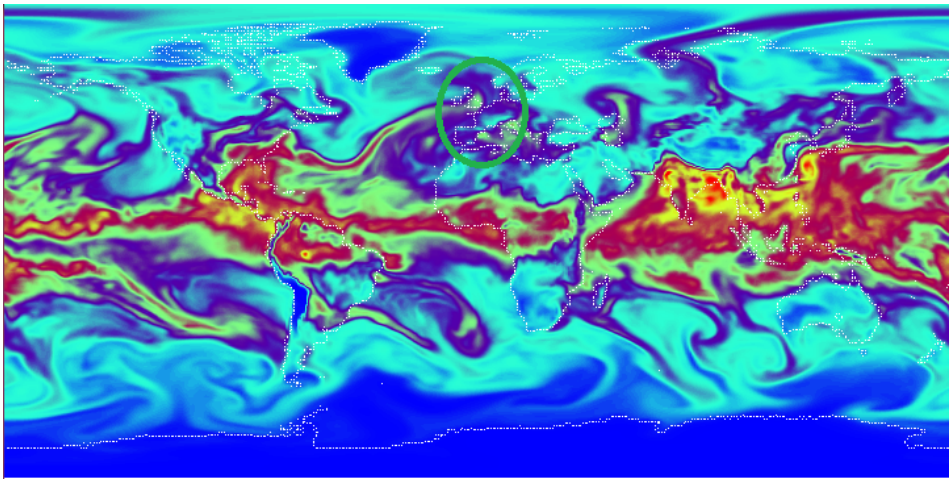
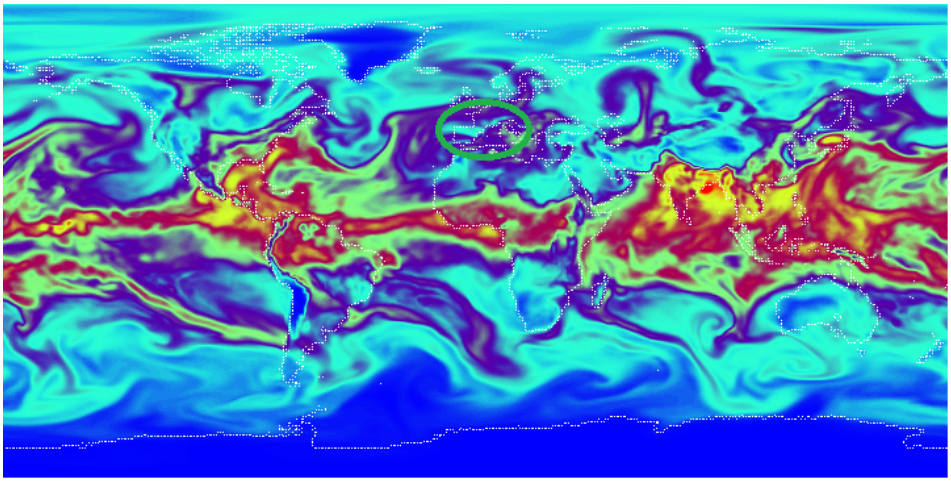


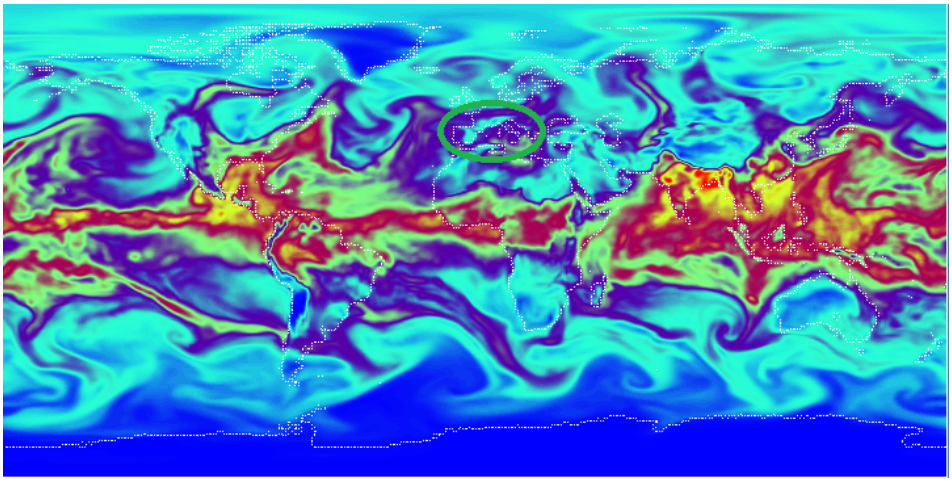
Figure 15: Frame 673 of 2003



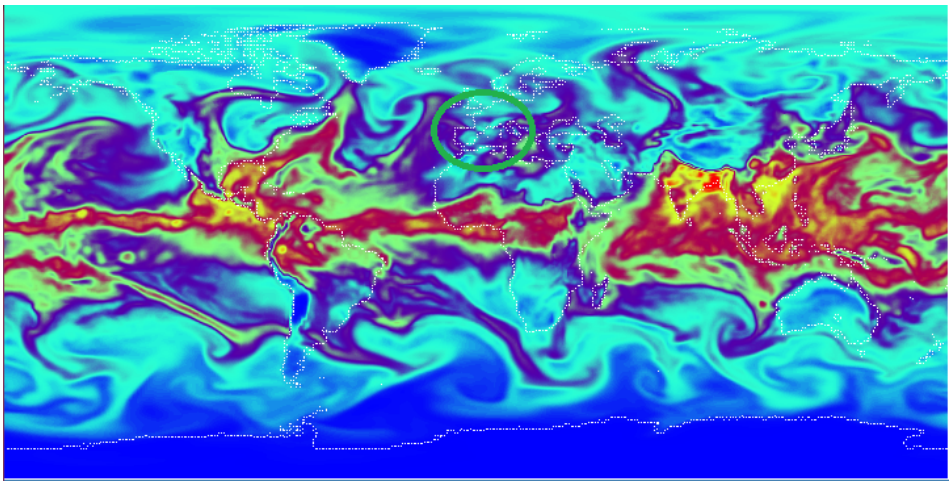
Frame 675 of 2003



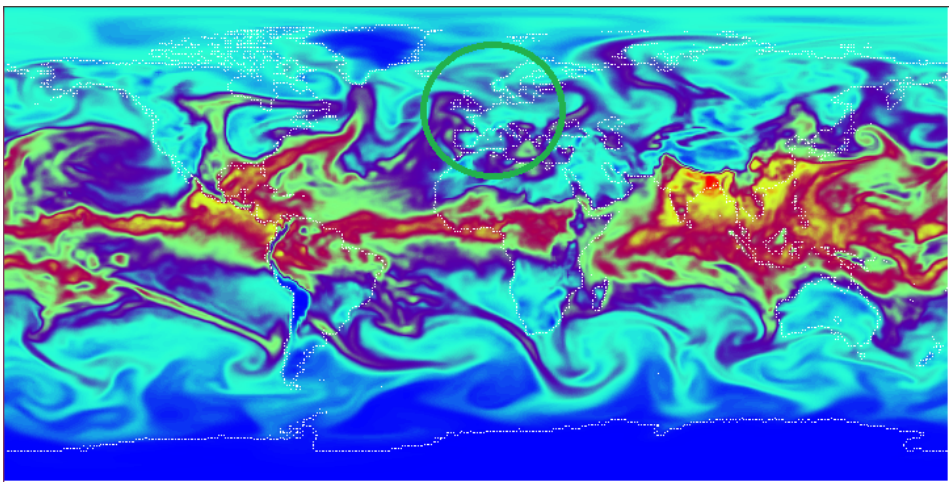
Frame 679 of 2003



Frame 682 of 2003



Frame 684 of 2003



Frame 686 of 2003

June 2010
Amount of cycles for bec: 26
Amount of cycles for tbec: 139
Bottleneck distance: 8.730

This is the second largest bottleneck distance we have calculated, even though there are very few *tbec* cycles for a summer month (the fewest of any summer month we've considered in this thesis). We have identified two instances of pincers which may be responsible for this bottleneck distance.

we see the first pincer to occur in June 2010. It closes in from above, and eventually also opens up towards the north. The humid region of the pincer is most clearly connected to Bergen in the second frame.

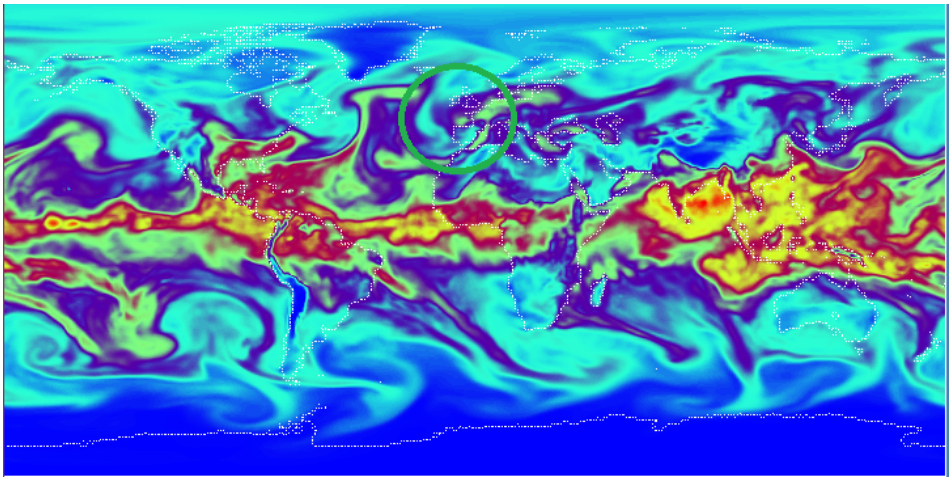
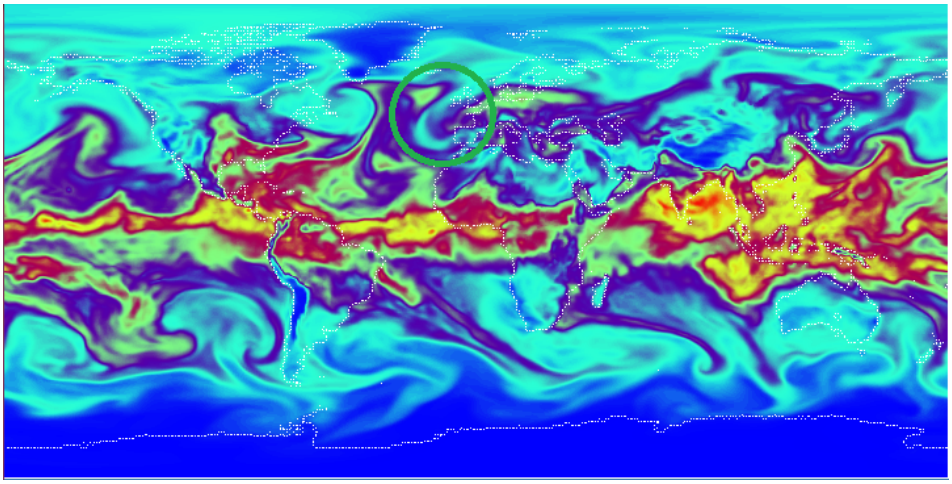
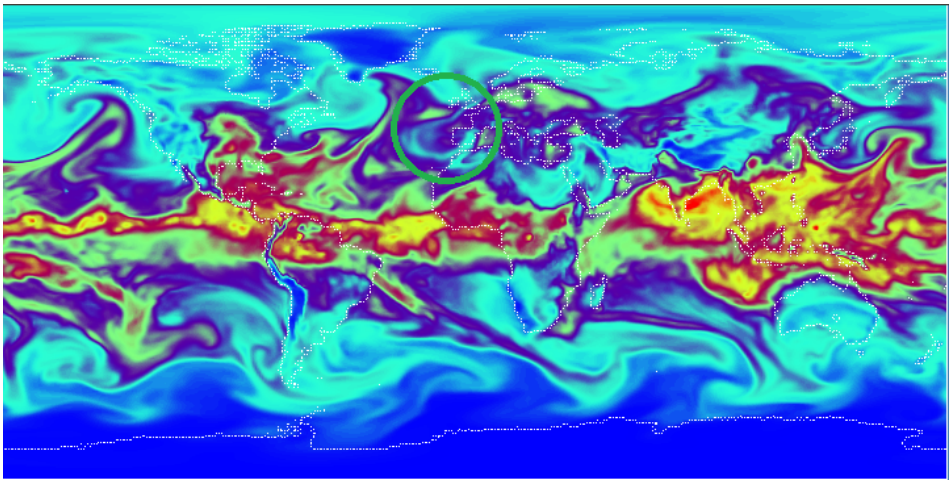


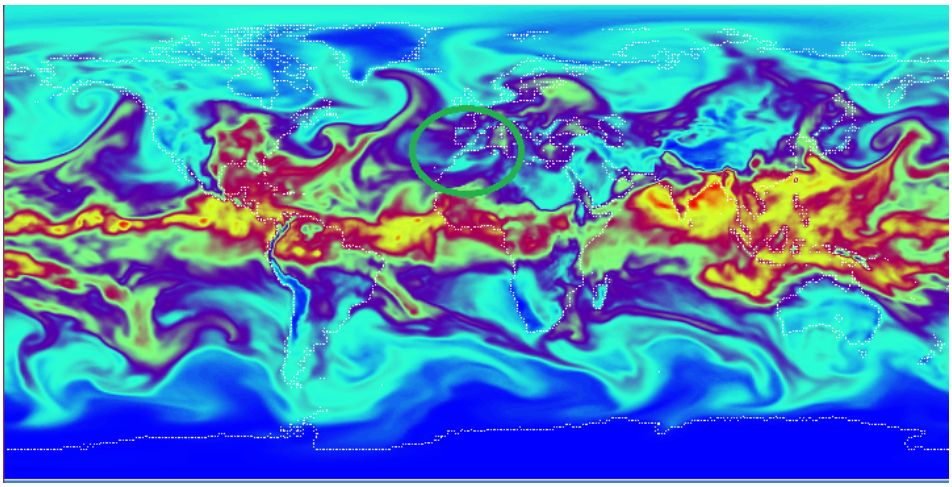
Figure 16: Frame 642 of 2010



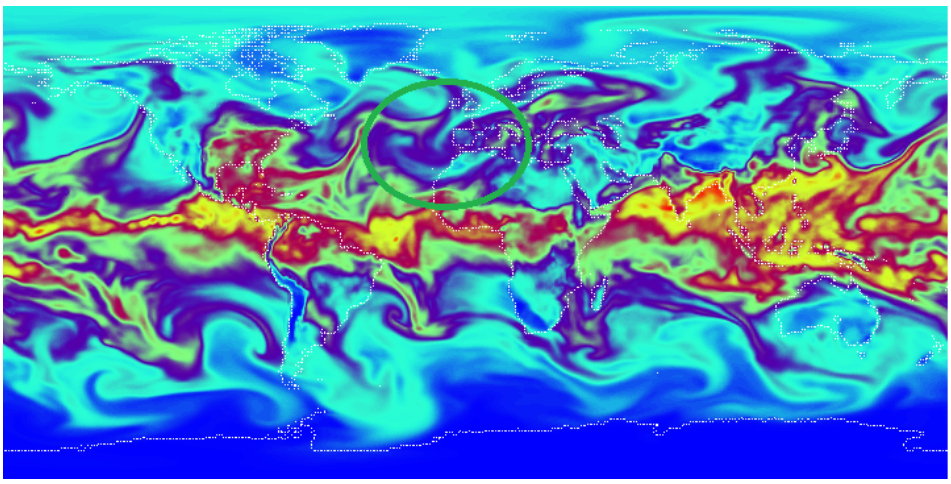
Frame 644 of 2010



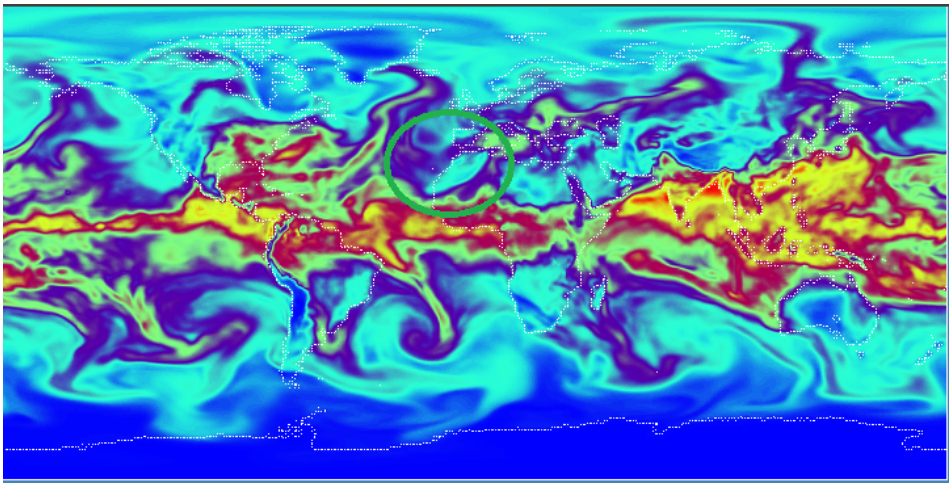
Frame 647 of 2010



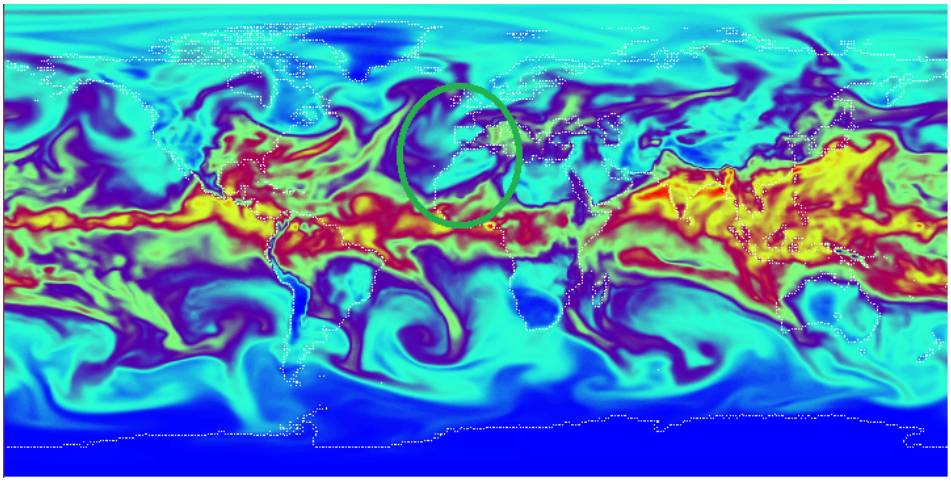
Frame 649 of 2010



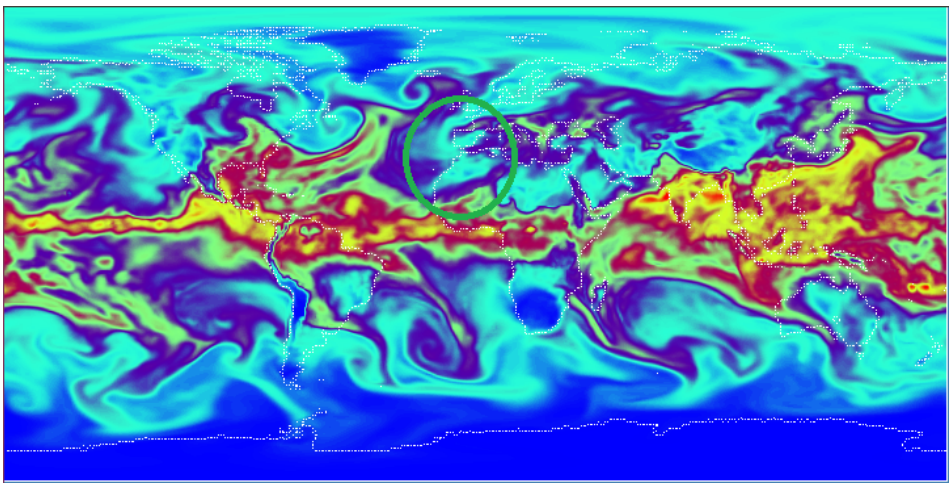
Frame 653 of 2010



Frame 660 of 2010



Frame 663 of 2010



Frame 666 of 2010

The pincer shown below is a bit more difficult to spot than the first one, but the spiral-like region of humidity captures a small, dry region inside of it until it appears to dissipate. The humid region of the pincer is most clearly connected to Bergen in the fourth frame.

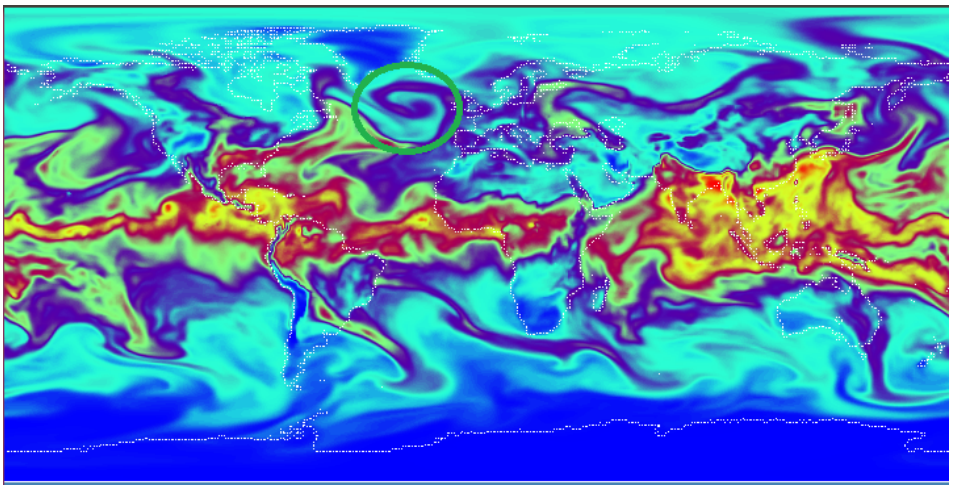
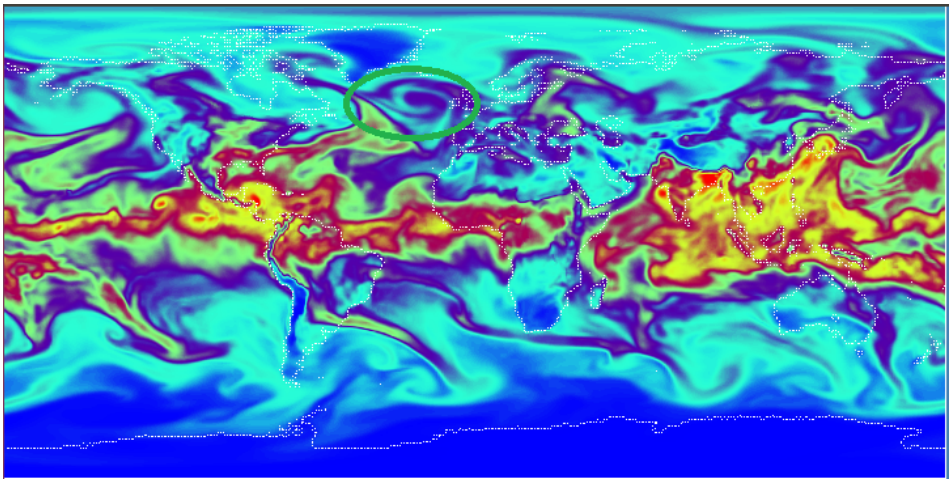
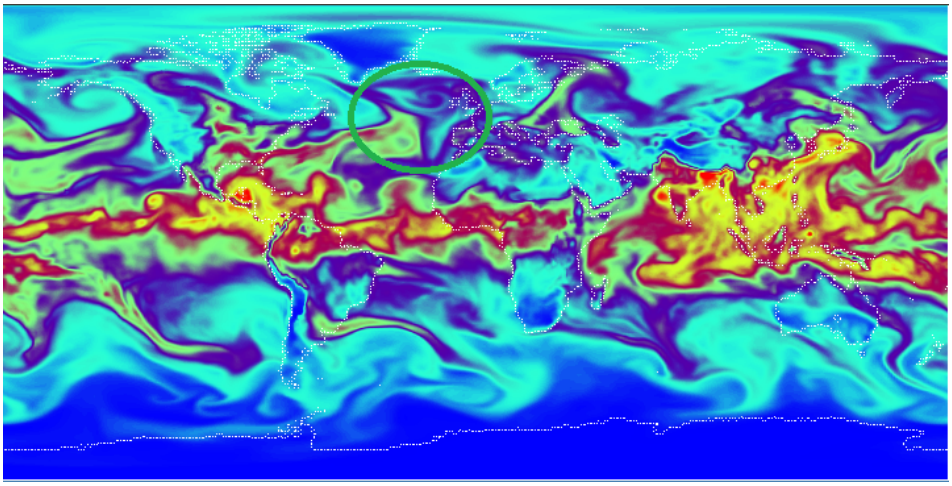


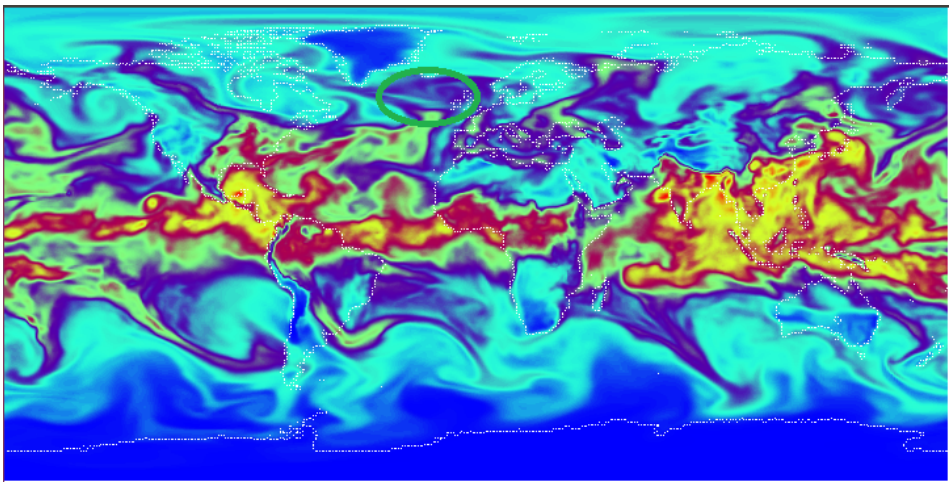
Figure 17: Frame 703 of 2010



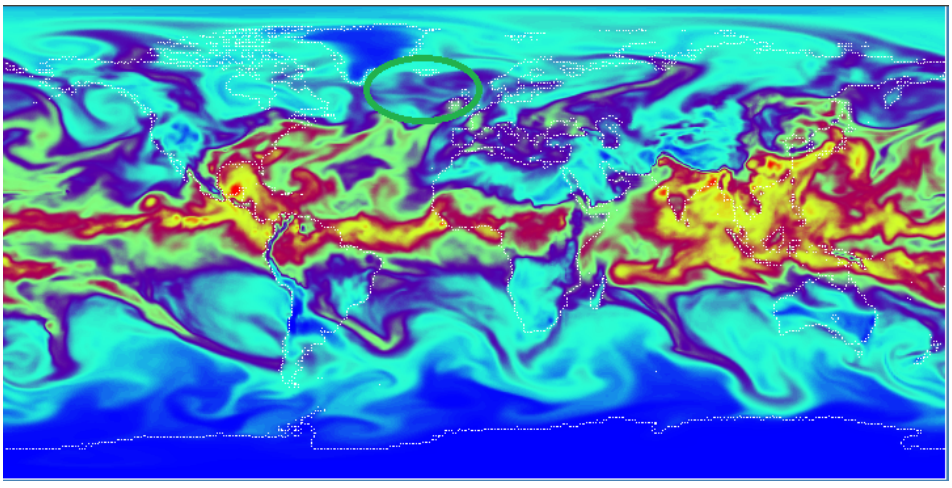
Frame 706 of 2010



Frame 709 of 2010



Frame 712 of 2010



Frame 715 of 2010

March 2011
Amount of cycles for bec: 30
Amount of cycles for tbec: 203
Bottleneck distance: 4.507

While smaller than the other bottleneck distances we've looked at so far, it is certainly large for March. It was difficult to find a possible explanation for this. In the frames below we've highlighted a thin, spiral-like region which may create a pincer, but it is so thin that it is hard to tell. There is obviously a river that reaches Bergen during this timespan.

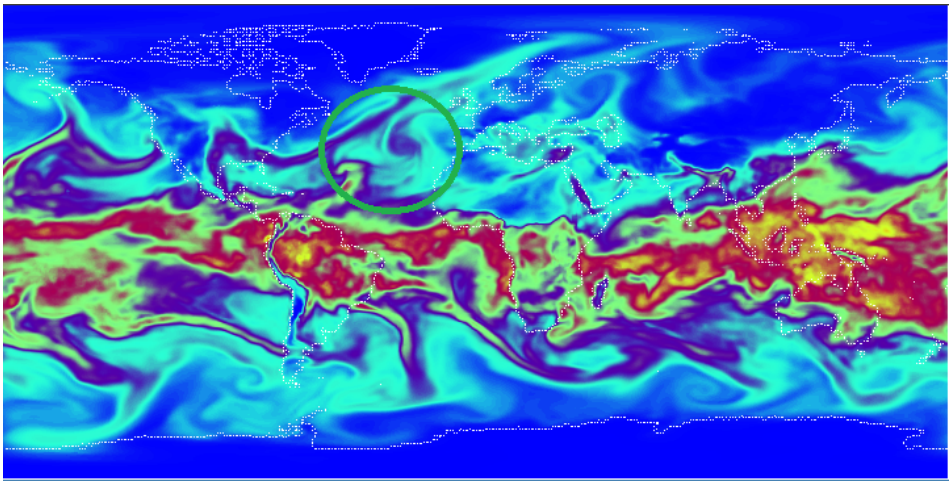
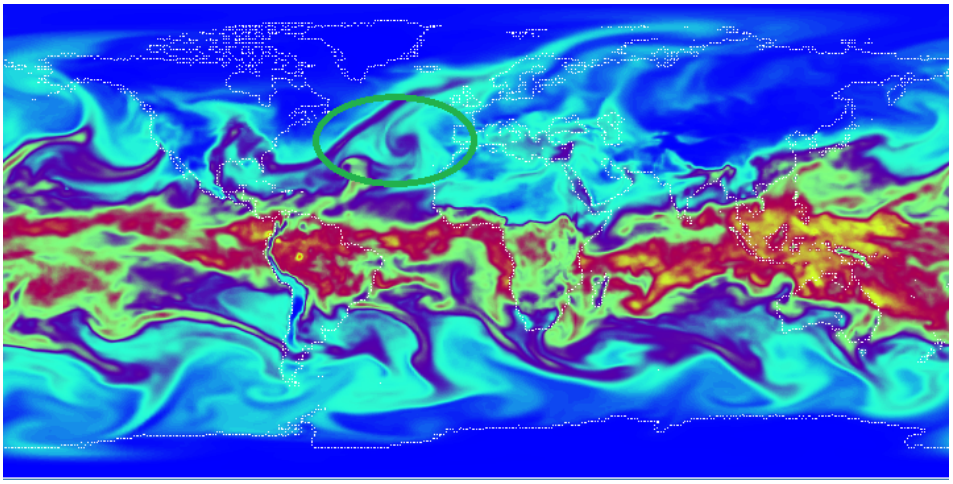
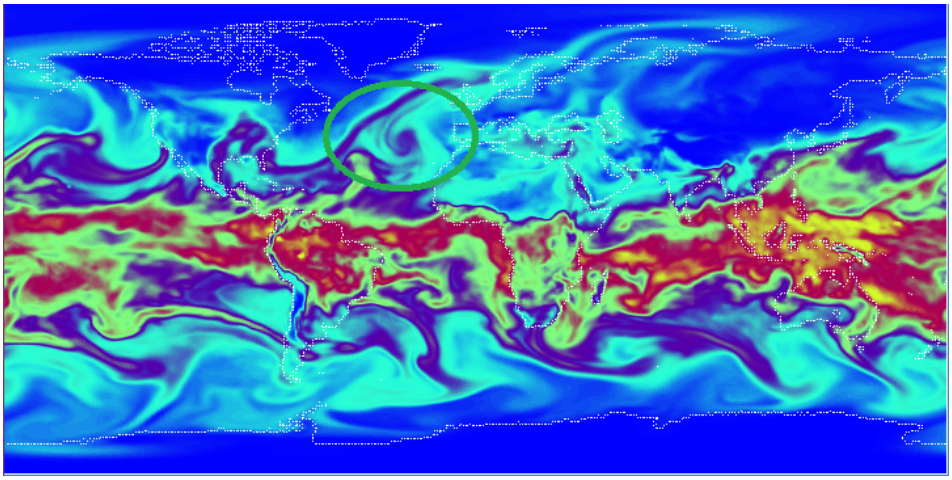


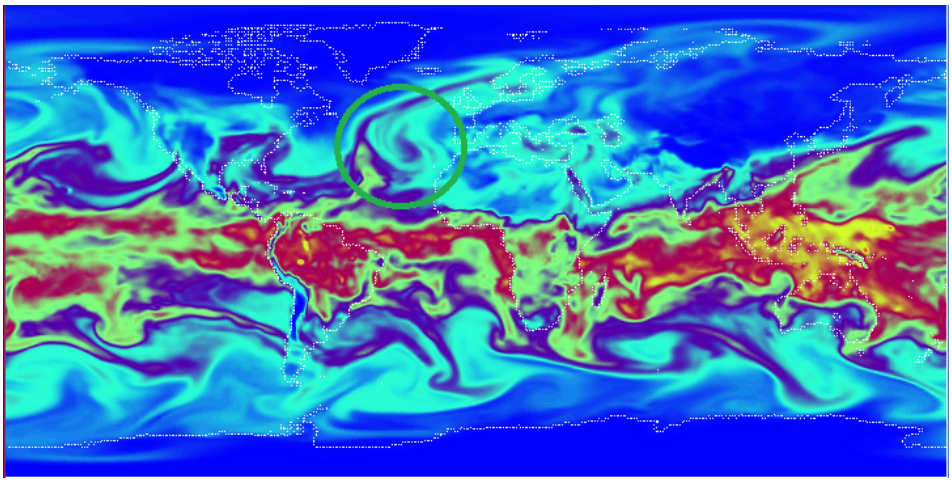
Figure 18: Frame 314 of 2011



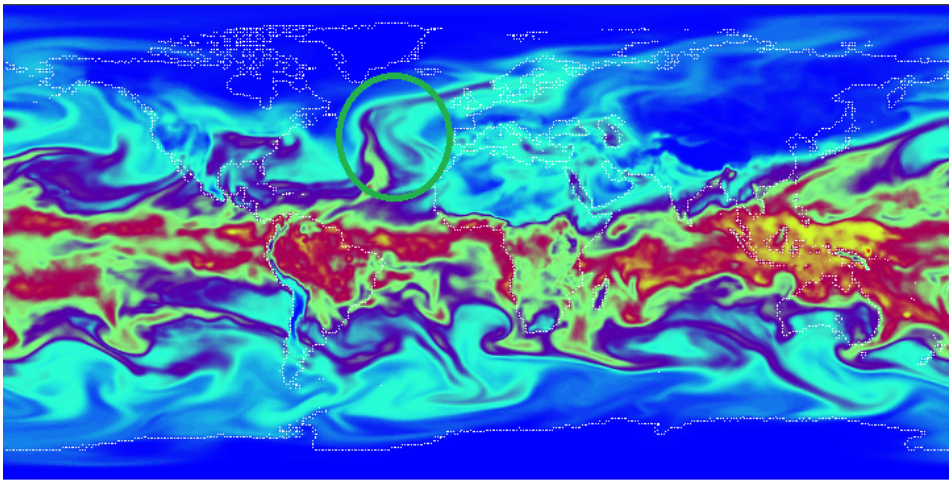
Frame 316 of 2011



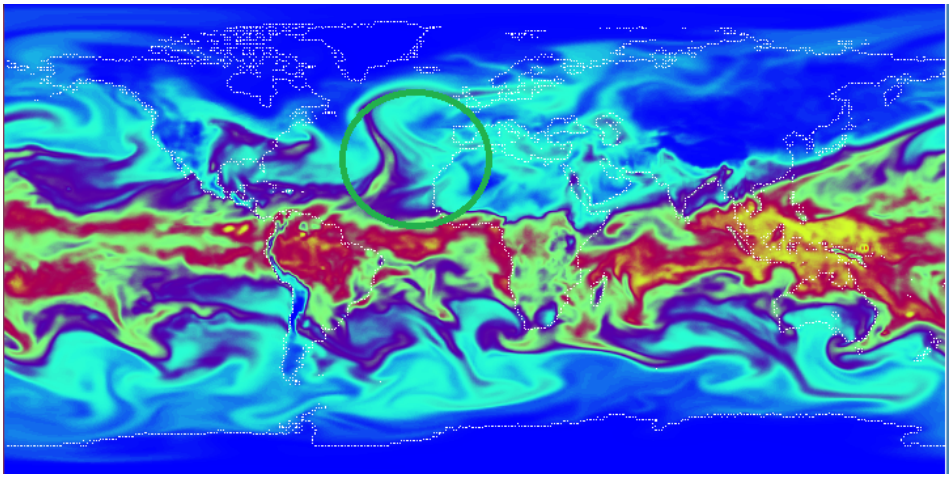
Frame 317 of 2011



Frame 319 of 2011



Frame 321 of 2011



Frame 323 of 2011

May 2011
Amount of cycles for bec: 27
Amount of cycles for tbec: 829
Bottleneck distance: 7.091

The third highest bottleneck distance we've calculated. Note the high amount of *tbec* cycles. For comparison, May of 2012 only has 78 *tbec* cycles and a bottleneck distance of 1.814. While it's hard to see why there are so many *tbec* cycles, we have found two events which may explain the large bottleneck distance. Below we see a quite humid plume clearly hitting Bergen. This is the most likely cause of the large bottleneck distance, (though there is an occurrence of a pincer that warrants attention).

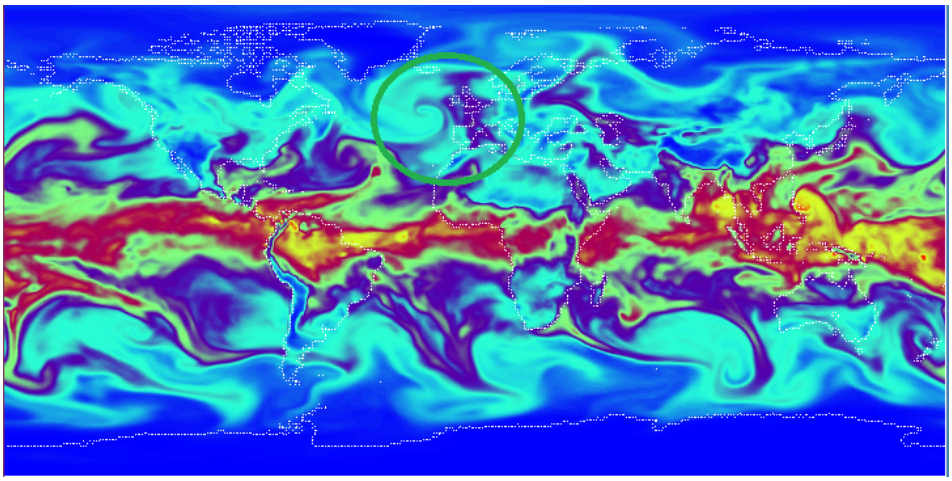
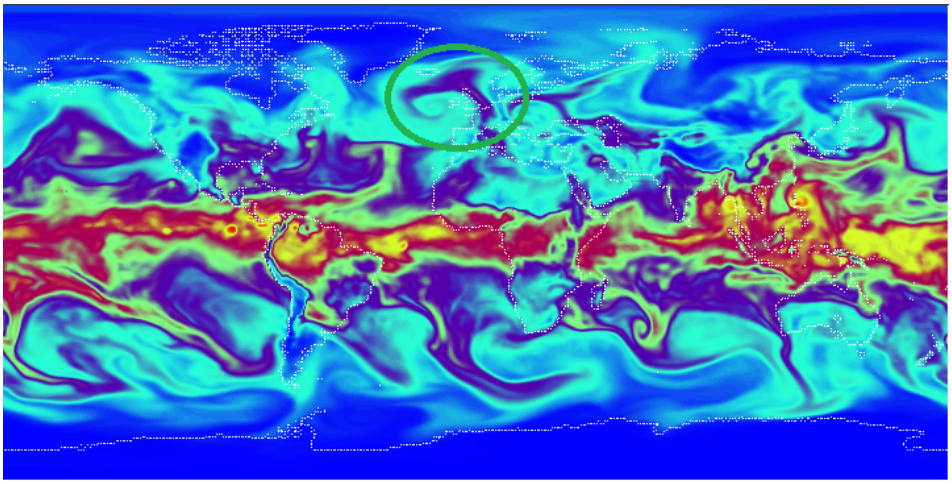
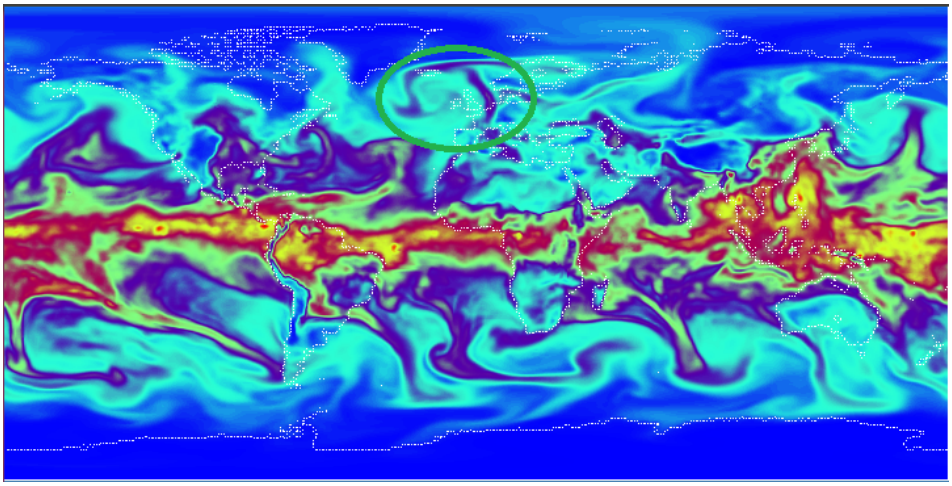


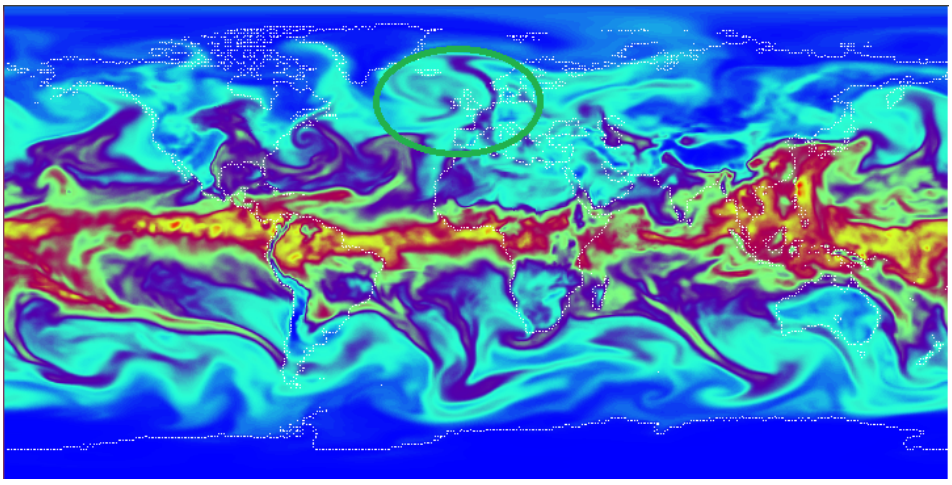
Figure 19: Frame 507 of 2011



Frame 511 of 2011



Frame 514 of 2011



Frame 517 of 2011

Here we see a dry region being encircled. The dry region is then transported westward, and opens up towards the north. The humid region of the pincer appears to be connected to Bergen at several time-steps.

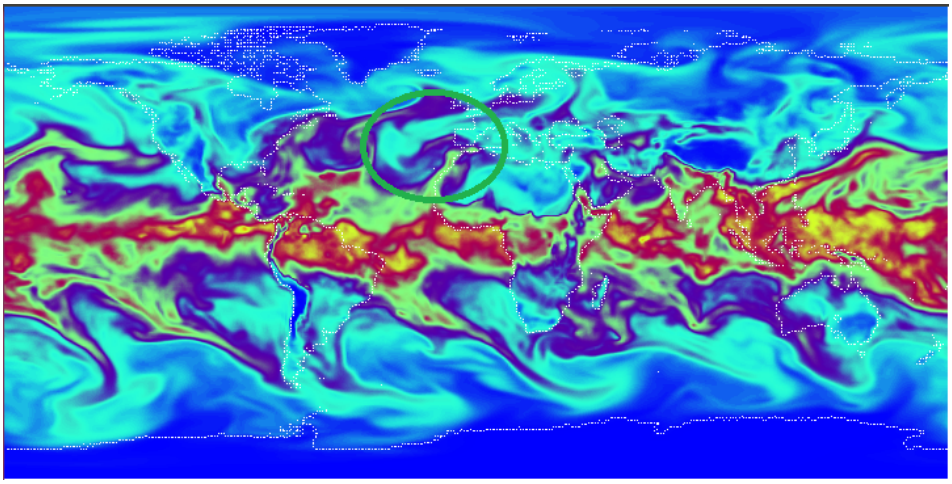
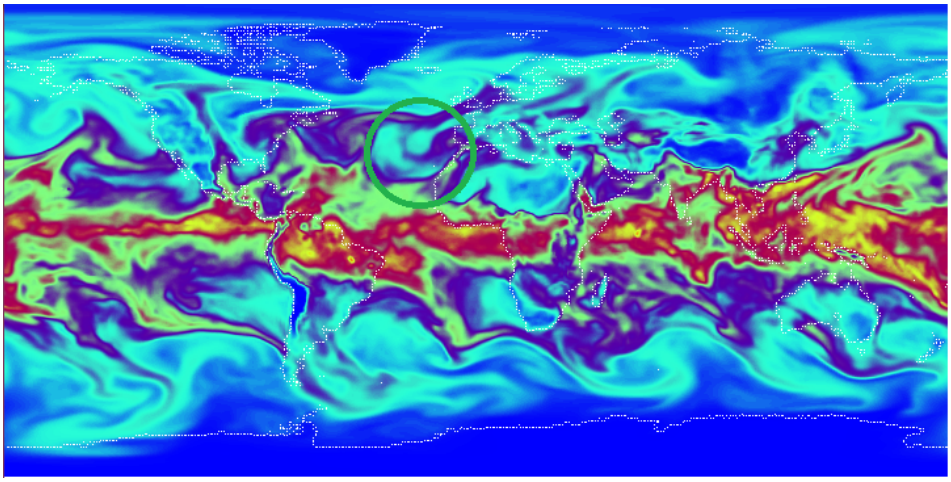
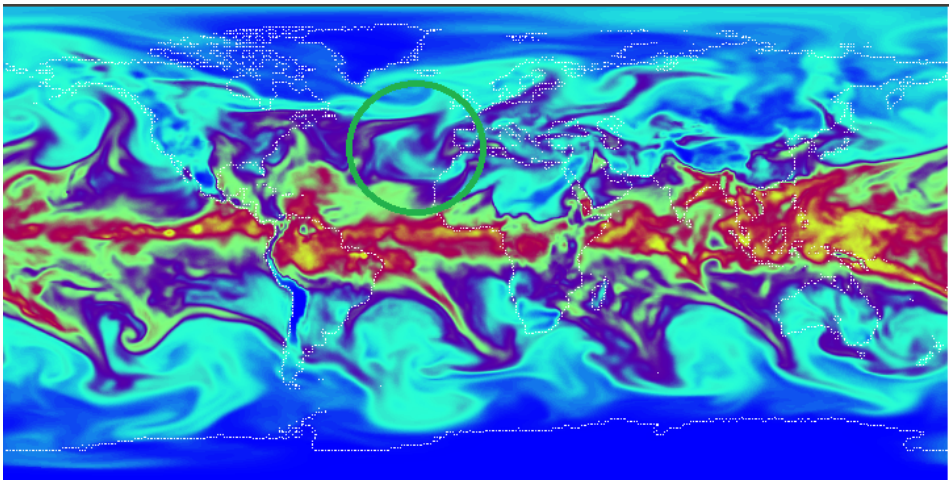


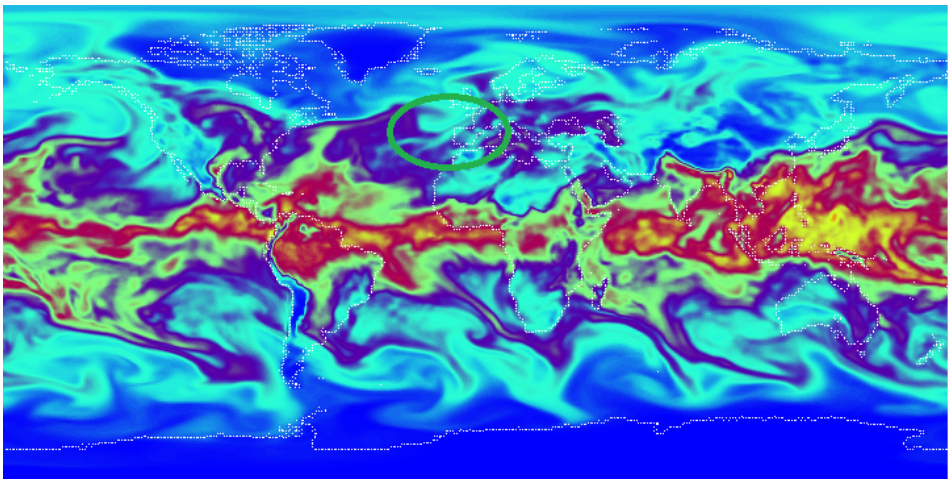
Figure 20: Frame 546 of 2011



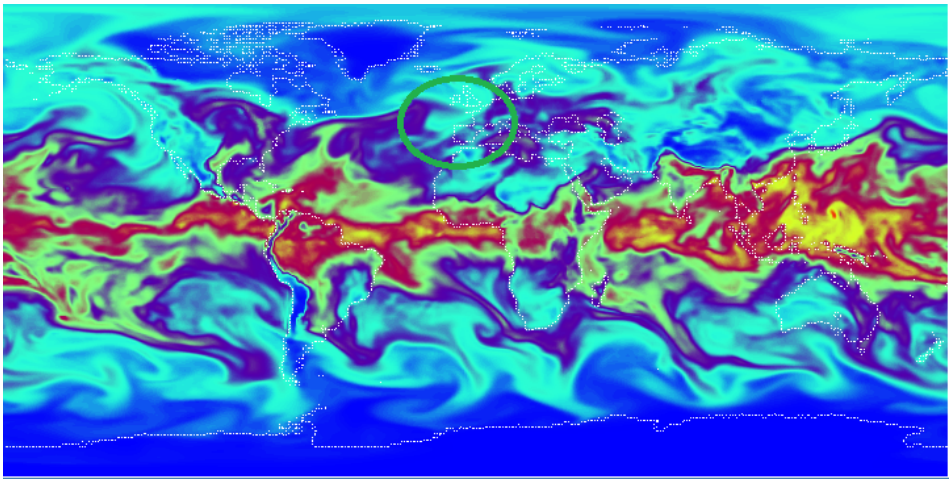
Frame 550 of 2011



Frame 555 of 2011



Frame 566 of 2011



Frame 567 of 2011

Now that we have identified a potential problem with the *tbec* algorithm, we move on to presenting an alternative algorithm in the hopes of solving it.

7 The altbec

Since this method was intended as an alternative to *tbec*, we have appropriately named it *the altbec*. We started with the idea of using a bottom-up filtration instead of a top-down filtration in order to build our 3-dimensional filtered complex. The thought was that rivers and plumes would be detected as holes in a 3D cubical complex, thus avoiding the need for "the back wall" of the *tbec* algorithm. This seemed to us to solve the problem posed by pincers, thus eliminating at least some of the unwanted cycles. A lot of the challenge of this thesis was devising a filtered complex with the properties we wanted, and we went through quite a few different concepts (and different versions of those concepts). While the method we ended using does have some drawbacks, it has some apparent advantages as well.

The *altbec* relies on two key concepts: the idea of calculating the 1-persistent homology of two "versions" of the same filtered complex, and the idea of "cross expanding". We describe the latter first.

To cross-expand a 3D grid, we first select a line through it along one of the axes of the grid (if we think of the grid as embedded in $\mathbb{R}^3$). In our case, we chose the line corresponding to Bergen going in the time-direction. For the sake of explanation, we shall call the direction of the line for "time", no matter the grid we're working with. So, moving in the time-direction, we "fatten" our line by creating duplicates of every point along it. These duplicates are added as adjacent points to the original one, in such a manner that the point we started with has now become a 3x3 grid along the two axes different from time. Note that the points that occupied these positions previously has not been replaced, but simply "pushed aside". The rest of the grid is then adjusted so as to not effect the persistent homology of the filtered complex.

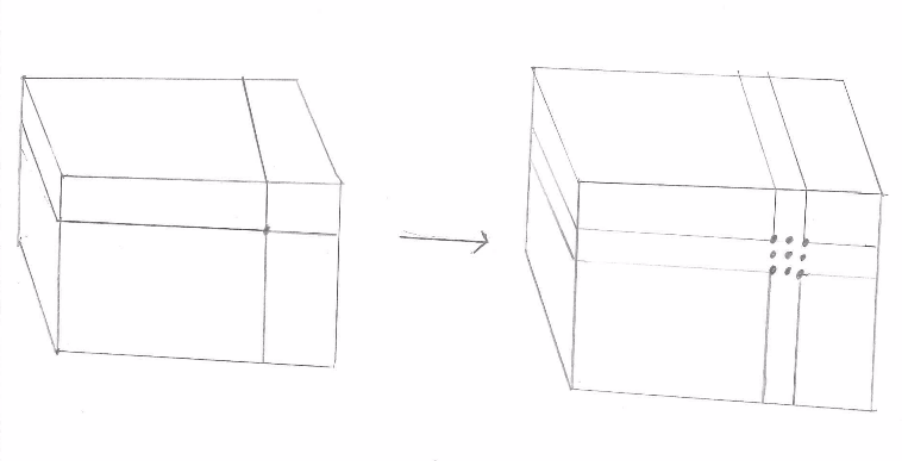


Figure 21: An illustration of a cubical grid being cross-expanded. The point in the center of the cross on the left becomes fattened up into the 3x3 grid on the right.

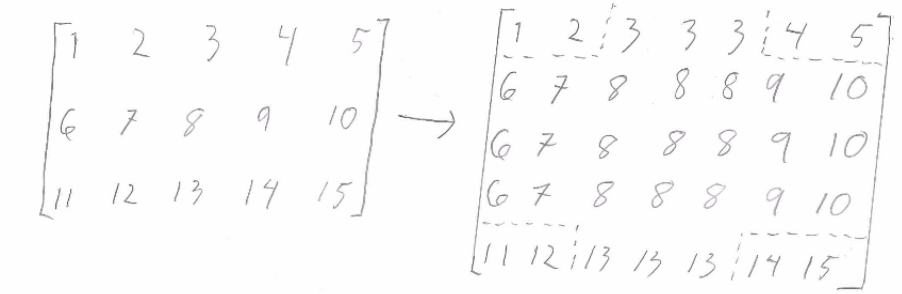


Figure 22: A 2D grid being cross-expanded at point number 8. This illustrates what happens at every fixed time-step.

As we can see, this method has been given the name cross-expand due to the fact that it takes a cross-shaped region of our grid (centered at our chosen line) and fattens it up by adding duplicate points. Now, these duplicate points will all be filled in at the same time as the original ones, so there will be no difference between the persistent homology of the original filtered complex and the cross-expanded one. So what then is the purpose? To understand, we must first go through the initial steps of the *altbec* algorithm.

When building the filtered complexes we need for the *altbec*, we start the same way as we would when building the filtered complex needed to calculate the *tbec*. We start with the grid:

$$G = \{-95°E, -94°E, \ldots, 15°E\} \times \{0°N, 1°N, \ldots, 65°N\} \times \{t_i, t_{i+1}, \ldots, t_n\}$$

and a function giving the *tcw* at each grid point. Then we deviate from the *tbec* by using a bottom-up filtration rather than a top-down filtration. This means that the driest points will be added first, and atmospheric rivers and plumes will be detected as holes in the complex. However, we are only interested in rivers and plumes hitting Bergen. This is where the second key concept comes in. We create two different versions of the same filtered complex - one with an artificially humid line going through Bergen and one without - and only include the cycles that occur in the former but not the latter, as they must be the ones involving the humid line (since the complexes are identical otherwise). The value along this humid line is arbitrary, so long as it is higher than the other *tcw* values in the filtration (we set it at 100, which is much higher than any recorded *bec* value in Bergen). We call the filtered complex with the humid line $\mathbb{X}$, and the the one without we call $\mathbb{Y}$. Since we are doing a bottom-up filtration, then at each filtration degree k, $\mathbb{X}_k \subseteq \mathbb{Y}_k$. These inclusions induce maps on homology. So mathematically speaking, what we're after is

$$ker(H_1(\mathbb{X}) \to H_1(\mathbb{Y}))$$

In practice, we find the desired cycles by removing the cycles in $H_1(\mathbb{X})$ which also occur in $H_1(\mathbb{Y})$.

However, we do not wish to replace the *tcw* values in Bergen with the artificially humid line. Doing this would mean we would be detecting rivers and plumes hitting points adjacent to Bergen. This is where we need the cross-expand concept. What we do is that when we build $\mathbb{X}$ and $\mathbb{Y}$, we cross-expand both of them along the line at Bergen. We then modify $\mathbb{X}$ so that the artificially humid line is located at the center of the "beam" we obtained by cross-expanding. The code for cross-expanding and modifying the value of the central line is can be found in Appendix A.

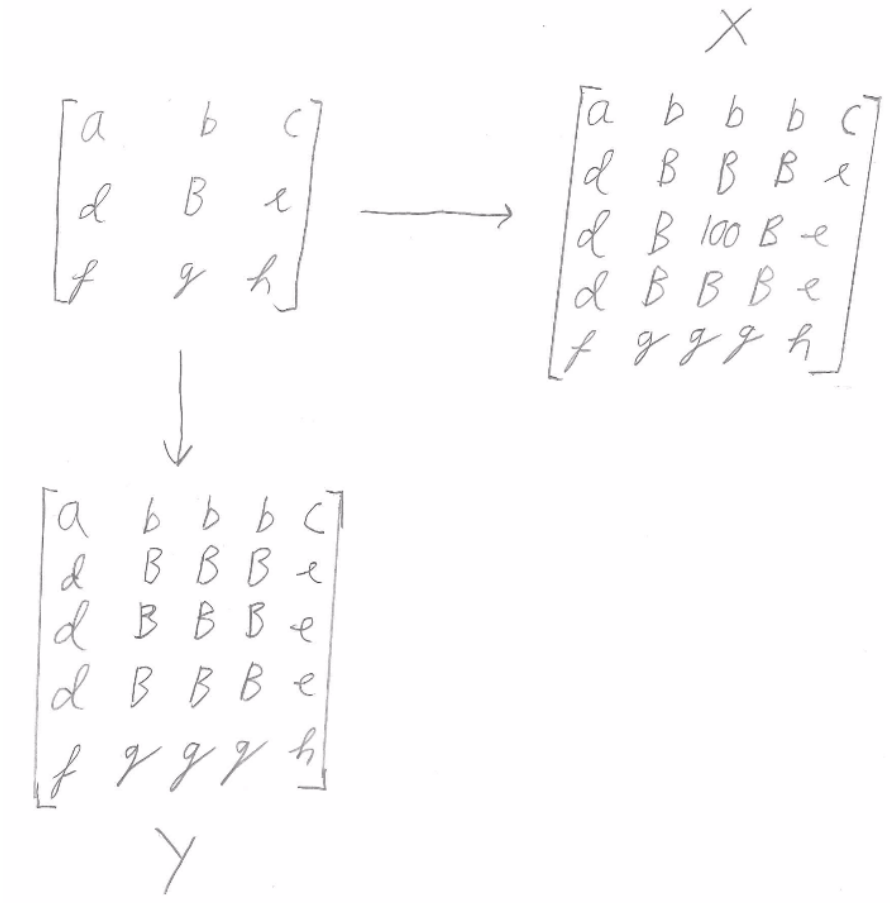


Figure 23: What we do at each fixed time-step. B denotes the value at Bergen.

We still have a few modifications left to do to our spaces. When we say "add" or "fill in" in the to-do list below, what we mean is to respectively add a layer of points with value zero, or to set the value of the relevant points to zero (so that they are included at the very start of the filtration).

- Add a front and a back wall in the time direction. For $\mathbb{X}$ we poke a hole through the back wall at the location of the artificially humid line through Bergen. This is so that the holes representing rivers and plumes can go through the complex, without there being a cycle living until filtration degree 100 going around Bergen. The back wall makes the longest cycle be born at degree 0, which is useful for comparing with *bec* and *tbec* (see the next section).

- Add a roof (latitudinal direction) and a side wall (longitudinal direction) next to Bergen to avoid cycles caused by humidity coming from above or to the east.
- Fill in the areas from Bergen and up for the first 40 time-steps of the filtration. This is because we add 10 days before the start of the month in order to detect rivers and plumes originating earlier than the start of said month, but we do not want to detect the rivers and plumes that actually hit Bergen during this time (we do the same thing in the *tbec* algorithm)

As with the cross-expansion, the code used for constructing the filtered complexes in the *altbec* algorithm is included in Appendix A.

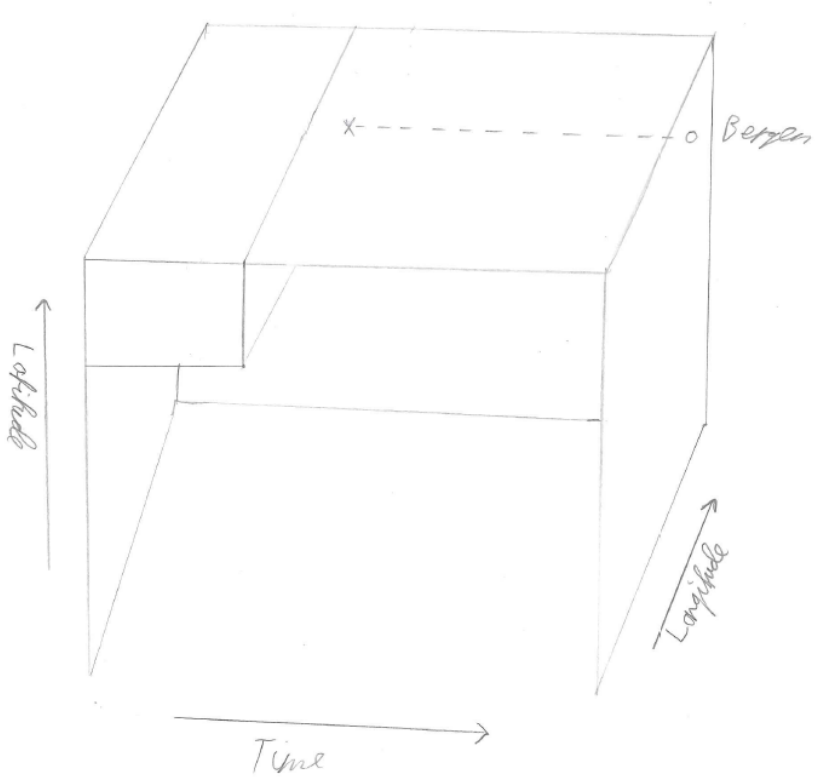


Figure 24: An illustration of $\mathbb{X}$ at filtration degree 0. The dotted line is the artificially humid line through the center of the "Bergen beam". The 'o' denotes where the line goes through the back wall, and the 'x' denotes where the line hits the already filled-in area at the first 40 time-steps.

Since we are calculating the persistent homology twice, and then sorting the cycles to select the cycles we want afterwards, the computation time is more than double that of the *tbec*. Each of the two persistent homology calculations produces cycles numbering in the tens of thousands, so it usually takes in excess of two hours per computation (that is, 2+ hours per month), even though we are using a powerful computer belonging to the Geophysical Institute of UiB (skd-cyclone). Luckily, the amount of cycles we're left with at the end of the selection process is much, much smaller.

8 Comparing the altbec with the previous methods

Recall that the filtered complexes in the *altbec* algorithm is built using a bottom-up filtration, whereas when we calculate the *bec* and *tbec* we use a top-down filtration. In order to compare the persistence diagrams produced by the former to those produced by the former, we must first swap the birth and death value of the *bec* and *tbec* cycles. For instance, if we plotted the diagrams of *altbec* and *bec* together without changing anything, their points would be on opposite sides of the diagonal. Thus, swapping the death values and the birth values of the points in the *bec* diagram amounts to simply mirroring the points through the diagonal, so they are on the same side as the points in the *altbec* diagram. In addition, the largest cycle in any given month for the *bec* and *tbec* goes to $-\infty$. We cut this off at 0, so that it corresponds with the same cycle in *altbec* (nothing changes below 0 anyway).

We now start by considering the same two examples we did in the start of section 3, namely February and June of 2011.

February 2011
Amount of cycles for bec: 24
Amount of cycles for tbec: 62
Amount of cycles for altbec: 30
Bottleneck distance tbec-bec: 2.270
Bottleneck distance altbec-bec: 2.291
Bottleneck distance altbec-tbec: 2.270

June 2011
Amount of cycles for bec: 24
Amount of cycles for tbec: 1108
Amount of cycles for altbec: 22
Bottleneck distance tbec-bec: 5.458
Bottleneck distance altbec-bec: 9.445
Bottleneck distance altbec-tbec: 4.578

As we can see, the $altbec - bec$ bottleneck distance is almost the same as the $tbec - bec$ bottleneck distance for February. We expected this, since February of 2011 is a winter month with few *tbec* cycles. In general, the $altbec - bec$ bottleneck distance is very similar the $tbec - bec$ bottleneck distance, which for other months is a somewhat unexpected result (more on this later). However, something interesting happens in June. The $altbec - bec$ distance is even larger than the $tbec - bec$ distance, despite the fact that *tbec* produces over 50 times the cycles that *altbec* does! In general, *altbec* produces a similar amount of cycles to *bec*, with no apparent seasonal variation. Usually a few more than *bec*, but sometimes one or two fewer, as is the case with June of 2011. In other words, the amount of cycles does not fluctuate dramatically from season to season as is often the case with *tbec*. This seems to be a huge advantage, but only if it means that it does not fail to detect something that it "should".

So what causes this large bottleneck distance, despite the low number of cycles? Since the humid line through Bergen goes through the back wall (which is completely filled in already in degree zero), we have that for each month there will be an *altbec* cycle born in degree zero. This will be the longest living cycle, and will "absorb" the most prominent river of that month. It is killed when the entire "Bergen beam" is filled in (save of course for the humid line with filtration degree 100). This means that the longest cycle in *altbec* only tells us about the largest *tcw* in Bergen during the month of interest. During most of the year, this poses little problem, as the most humid time-steps in Bergen almost always correspond with a river. However, during the summer, when it is very humid in general, this may not always be the case. Looking directly at the persistence diagrams, we see that the longest *altbec* cycle in June 2011 is $(0, 34.464)$ (rounded to three decimal places as with the bottleneck distance), whereas the longest *bec* cycle for this month is $(0, 25.019)$. The difference between the end values matches up exactly with the bottleneck distance of 9.445.

In order to check what would happen if this large difference in the longest cycle wasn't there, we modify the longest *altbec* cycle by giving it the end value of the longest *bec* cycles. Doing this gives us a bottleneck distance of 6.557, which is still quite large. There appear to be quite a few plumes during this once. For example the cycle $(11.782, 30.584)$ from the *altbec* has the exact same initial value as the cycle $(11.782, 18.252)$ from the *bec*, yet its end value is much higher. This "lengthening" of cycles is typical for plumes, as plumes are often part of weaker rivers which connect to Bergen at lower filtration degrees. Plumes like this would certainly explain the large bottleneck distance.

We will now revisit some of the examples from section 6. We refer to Appendix B for a full list of calculated bottleneck distances.

June 2001
Amount of cycles for bec: 26
Amount of cycles for tbec: 562
Amount of cycles for altbec: 23
Bottleneck distance tbec-bec: 9.403
Bottleneck distance altbec-bec: 9.403
Bottleneck distance altbec-tbec: 3.901

Here we have the exact same bottleneck distance between *altbec* and *bec* as between *tbec* and *bec*, which is not that unexpected since we did spot a plume during this month. Also note that the $altbec-tbec$ bottleneck distance is much smaller, further strengthening our belief that the other bottleneck distances are due to a plume. However, it is doubtful that this bottleneck distance is caused by the plume that we spotted. If we modify the longest *altbec* cycle $(0, 32.449)$ to set it equal to the longest *bec* cycle $(0, 23.046)$ as we did with June of 2011, we see that the difference in the end values is equal to 9.403. As we commented above Figure 12, the plume we observed does not appear to be part of a river which connects Bergen to the equator at some lower degree, therefore it is highly unlikely that this plume is the originator of the longest cycle. That being said, the $altbec - bec$ bottleneck distance we get after modifying the longest cycle is still very large at 7.721, and there are a couple of persistent cycles in *altbec* which do not appear to correspond to anything persistent in *bec*, for instance the cycle $(13.587, 29.591)$, so there is still a good chance that the plume in Figure 13 is detected.

Looking through the visualizations again, there does in fact appear to be a plume which might correspond to the lengthening of the longest cycle, and thus explain the large bottleneck distance. It was just a bit difficult to spot.

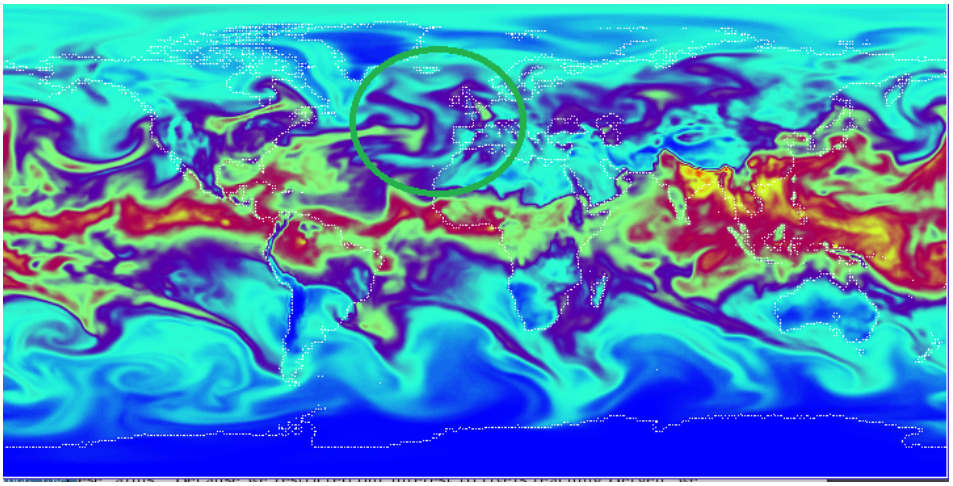
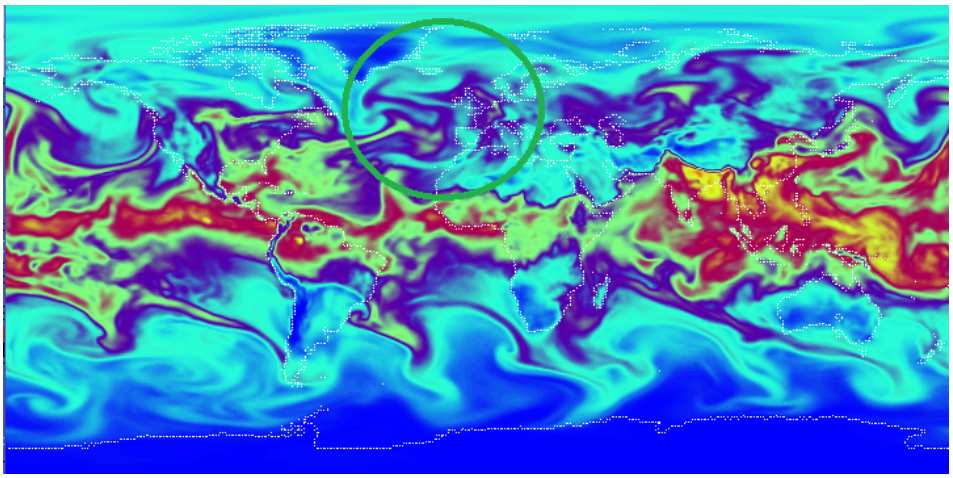
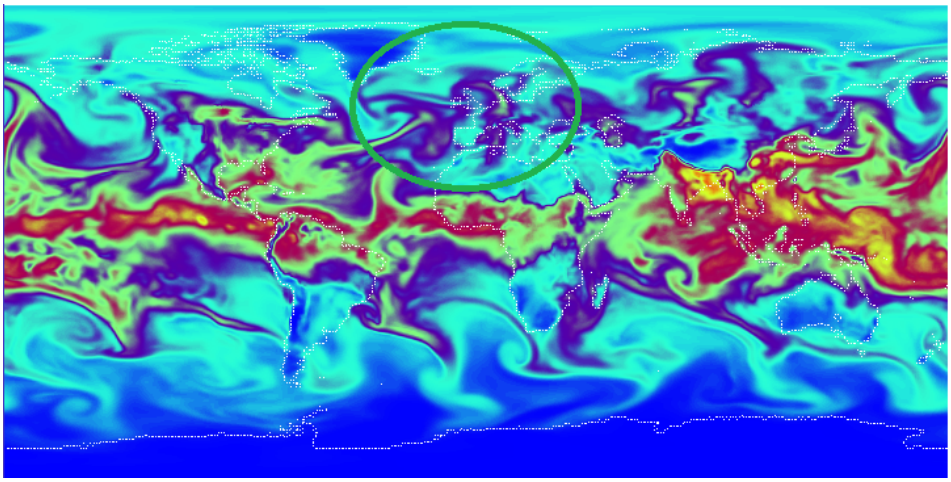


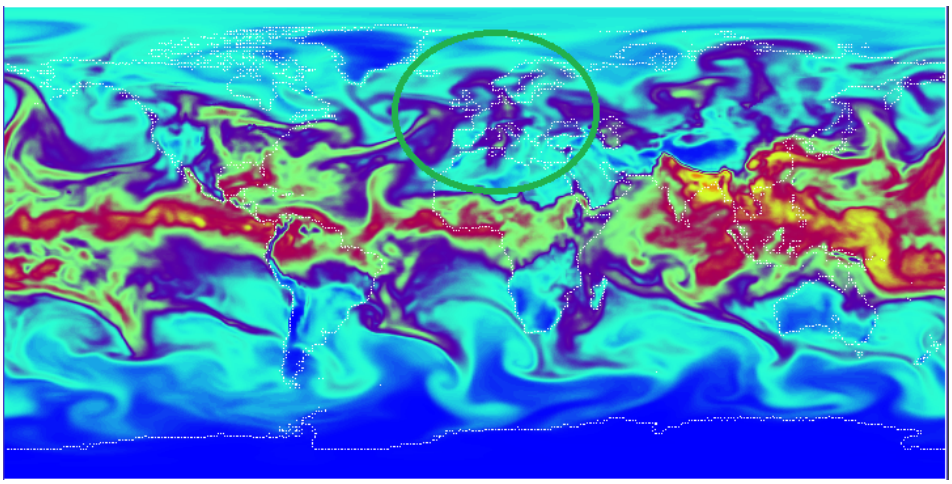
Figure 25: Frame 709 of 2001



Frame 711 of 2001. We see that there is now a gap between the plume and the humid region connected to the equator.



Frame 713 of 2001. It is not clear how connected the shape inside the circle is (there is a thin path to the left that might connect it to the equator), but the region that hits Bergen is clearly more humid than the rest of it.



Frame 714 of 2001

May 2011

Amount of cycles for bec: 27
Amount of cycles for tbec: 829
Amount of cycles for altbec: 40
Bottleneck distance tbec-bec: 7.091
Bottleneck distance altbec-bec: 7.091
Bottleneck distance altbec-tbec: 4.694

Here we have a similar situation as in June of 2001 where the $altbec - bec$ and $tbec - bec$ bottleneck distances are the same. While the $altbec - tbec$ distance is smaller in this case as well, the difference is not as large as in the previous example. This is an expected result, as we spotted both a plume and a pincer during this month. Also as is the case for June of 2001, we have that the bottleneck distance is caused by the difference in the longest cycle, which is $(0, 27.334)$ for *altbec* and $(0, 20.243)$ for *bec*. More surprising is the fact that we still get a rather large bottleneck distance of 5.883 once we modify the *altbec* diagram in the usual way. There does not appear to be any other distinct plumes occurring this month, but there are a few persistent cycles not corresponding to anything in *bec*. Two cycles in particular stand out: $(3.693, 24.812)$ and $(9.021, 24.628)$. Does our plume correspond to one of these cycles or to the lengthened longest cycle? Since it does not appear to be part of a weaker river, it is most likely to be one of the former. By design, the *altbec* should not detect pincers, so it is unclear what is causing the other cycle.

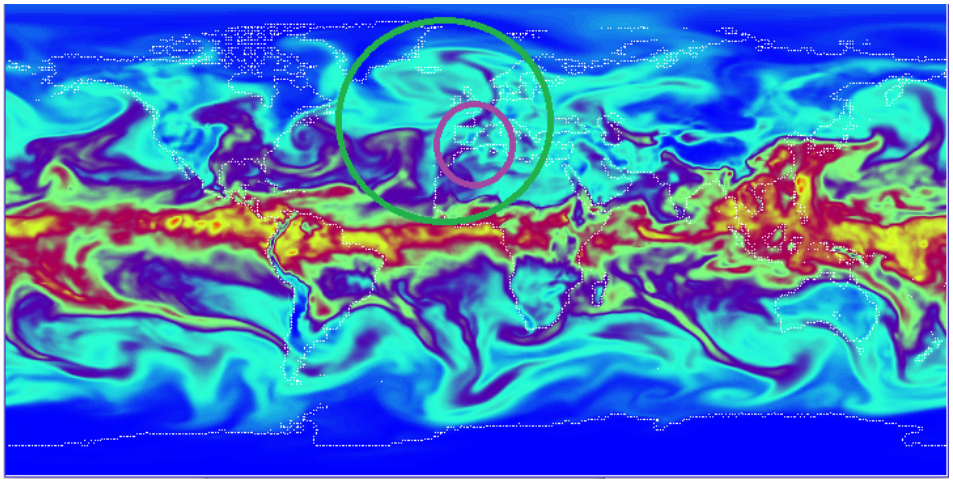


Figure 26: Frame 518 of 2011, when the plume is sitting right on top of Bergen. There appears to be a clear gap between the plume and the dark-blue region connected to the equator (look at the violet ellipse inside the green one), hence it is doubtful that this plume is part of a river which connects to Bergen at a slightly lower filtration degree.

March 2011
Amount of cycles for bec: 30
Amount of cycles for tbec: 203
Amount of cycles for altbec: 33
Bottleneck distance tbec-bec: 4.507
Bottleneck distance altbec-bec: 4.507
Bottleneck distance altbec-tbec: 2.535

This was perhaps the most surprising of all our results. At first glance, it seems like a similar situation to the ones we had at June of 2001 and May of 2011, but recall that the only thing we could spot that might cause this bottleneck distance was a thin spiral-like region that might form a pincer. However, the *altbec* should not be able to detect pincers, yet the bottleneck distance is exactly the same. Could this be caused by a plume that we missed? We check by modifying the longest cycle in *altbec* to correspond with the longest cycle in *bec*, but we still get the exact same bottleneck distance, so it is not the longest cycle that is causing it. Looking through the cycles manually, we do indeed spot what appears to be a plume: a cycle $(4.216, 21.105)$ in *altbec* that seems to correspond to a cycle $(4.216, 15.587)$ in *bec*. However, we also spot a cycle $(16.512, 21.920)$ that does not appear to correspond to any cycle in *bec*. Could this cycle be what both *altbec* and *tbec* picked up? If so, which phenomenon does it correspond to? Considering the visualizations, there does not appear to be anything immediately recognizable that would explain this.

9 Concluding remarks

The *altbec* method has the apparent advantage that it produces a much smaller number of cycles compared to the *tbec* method, thus making it much easier to compare with the *bec*. However, it also has its downsides. The most glaring one being that the longest cycle simply tells us what the highest *tcw* value in Bergen was. This has a tendency to cause trouble during the warmer months, as we could see with June of 2011. It also takes longer to compute than the *tbec*. Perhaps a future task could be to tweak the algorithm in order to overcome this problem, or eventually run *tbec* as a supplementary algorithm to *altbec* in order to accurately detect the longest cycle (though this would increase computation times even further).

I created the *altbec* in the hopes of eliminating cycles created by pincers, as I feared that these cycles could be rather persistent. Looking at the results of the bottleneck comparisons however, it becomes clear that most of the large bottleneck distances between *tbec* and *bec* are not caused by pincers, as they also occur when we compare *altbec* with *bec*. They appear to be due to something unforeseen, or more benignly, plumes that are hard to spot.

While pincers might certainly still pose an issue (as seen in the figure below) we can conclude that they are not the cause of the most conspicuous bottleneck distances. Whether what actually causes these distances are "features" or "bugs" is a topic for future discussion between topologists and meteorologists.

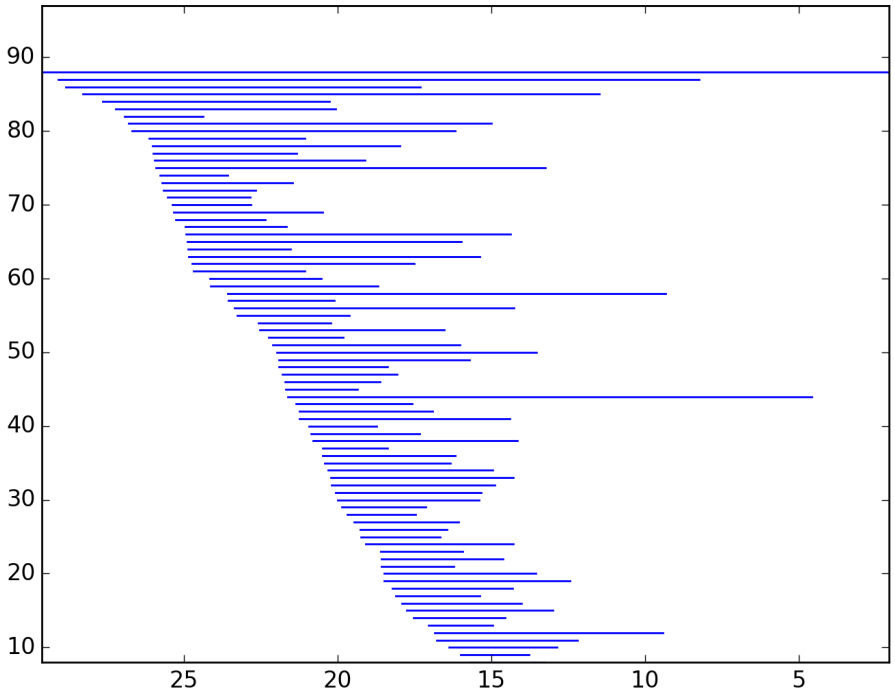


Figure 27: The most persistent *tbec* cycles in June of 2000. The sheer amount of cycles with a non-trivial length is a reason to still be concerned about pincers.

References

[Alfsvåg] K. Alfsvåg, *Detecting Atmospheric Rivers Using Persistent Homology* Universitetet i Bergen (2015)
Link: https://bora.uib.no/handle/1956/10335
Source code and documentation: http://org.uib.no/mi/master/kal045/main.html

[Dionysus] D. Morozov, *Dionysus*
Link: http://www.mrzv.org/software/dionysus/

[Chazal et al.] F. Chazal, V. de Silva, M. Glisse, S. Oudot,
The Structure and Stablity of Persistence Modules (2012)
Link: https://arxiv.org/abs/1207.3674

[Z-C] A. Zomorodian, G. Carlsson,
Computing Persistent Homology Discrete Computational Geometry February 2005, Volume 33, Issue 2 (2005)
Link: https://link.springer.com/article/10.1007/s00454-004-1146-y

[Hatcher] A. Hatcher, *Algebraic Topology* Cambridge University Press (2001)
Link: https://www.math.cornell.edu/ hatcher/AT/AT.pdf

[Fraleigh] J. B. Fraleigh,
A First Course in Abstract Algebra Seventh Edition. Pearson Education (2003)

[Rutz] J. J. Rutz, W. J. Steenburgh, and F. M. Ralph. *The inland penetration of atmospheric rivers over western north america: A lagrangian analysis* Monthly Weather Review (2015)
Link: http://journals.ametsoc.org/doi/abs/10.1175/MWR-D-14-00288.1

[Dee et al.] D. P. Dee and co-authors. *The ERA-Interim reanalysis: configuration and performance of the data assimilation system* Quarterly journal of the Royal Meteorological Society, 137(656,A):553-597, Apr. 2011. Link: http://onlinelibrary.wiley.com/doi/10.1002/qj.828/abstract

[Stohl] A. Stohl, C. Forster, H. Sodemann, *Remote sources of water vapor forming precipitation on the Norwegian west coast at 60N – a tale of hurricanes and an atmospheric river.* Journal of Geophysical Research Volume 113, Issue D5, 16 March 2008.
Link: http://onlinelibrary.wiley.com/doi/10.1029/2007JD009006/full

Appendix A: Code

It should be note that, as this was my first foray into python programming, these scripts very are ad-hoc in the sense that they are made only with our specific case in mind. They may not be suitable for use in a more general setting. For example, it might yield an error if one attempted to input a different grid into the ha.sliceArray() function called in altbec.py. It should also be noted that these scripts build on the framework already created by Kristian Alfsvåg. A link to his source code and its documentation can be found in the bibliography.

I wish to apologize for the weird indentation going on here. I don't know why the package pythonhighlight.sty is doing this, and I do not have time to fix it. Hopefully it is still readable.

Cross-Expand

```
import numpy as np

#cross-expanding a 2D array

def crossExpand(A, c, wetPoint = False):

#A: an array
#c: the coordinates of a point in A
#wetLine: whether or not to raise the value at c to 100

    n=c[0]
    e=c[1]
    N = A.shape[0]
    E = A.shape[1]
    B = np.zeros((N+2,E+2))

    expand = range(0,e+1) + [e] + range(e,E)

    for j in range(0,n+1):
        B[j,] = A[j,expand]

    B[n+1,] = A[n,expand]
    if wetPoint == True:
     B[n+1,e+1] = 100
    else:
     B[n+1,e+1] = A[n,e]


    for j in range(n+2,N+2):
       B[j,] = A[j-2,expand]

    return B
```

```
#cross-expanding a 3D array

def crossExpand3D(A, c, wetLine=False):

    X = np.zeros((A.shape[0],A.shape[1]+2,A.shape[2]+2),dtype=
                                    float)

    for j in range(0,A.shape[0]):

      X[j,:,:] = crossExpand(A[j,:,:],c,wetLine)

    return X
```

The altbec

```
import numpy as np
import copy

import arraymod
import ha
import cfc
from cells import CubicalCell

def altBec(month,year):

#month is a number from 2 to 12 (January is not included since
                                the algorithm relies upon
                                including the last 10 days of
                                the previous month).

    timeInterval = 122 #1464/12
    mint = (month-1)*timeInterval - 40 #-40 since we are
                                    including the previous 10
                                    days
  maxt = month*timeInterval
  bergen = (5,100) #the coordinates of the vertex at Bergen (it
                                's actually (60,5), but in our
                                grid the longitude starts at -
                                95 and the array for tcw starts
                                 counting the latitudes from
                                above)

  tcw = ha.loadTcw(year)

  A = ha.sliceArray(tcw,mint,maxt,1,0,65,2,-95,15,2)

  B = arraymod.crossExpand3D(A, bergen, True) #here, the line
                                at the centre of the cross has
                                its  filtration value set to
                                100
```

```
  C = arraymod.crossExpand3D(A, bergen, False) #here, the line
                                  at the centre of the cross has
                                      the same filtration value
                                  as Bergen

#we add a solid front wall, so that we prevent the cycle going
                                  around the artificially wet
                                  line through Bergen from living
                                   until degree 100

  W1 = np.zeros((B.shape[1],B.shape[2])) #B and C have the same
                                   shape
  front = W1.reshape((1,B.shape[1],B.shape[2]))
  B = np.vstack((front,B))
  C = np.vstack((front,C))

  #we also add a side wall and a roof to avoid unwanted cycles

  W2 = np.zeros((B.shape[0],B.shape[1]))
  side = W2.reshape((B.shape[0],B.shape[1],1))
  B = np.dstack((B,side))
  C = np.dstack((C,side))

  W3 = np.zeros((B.shape[0],B.shape[2]))
  roof = W3.reshape((B.shape[0],1,B.shape[2]))
  B = np.hstack((B,roof))
  C = np.hstack((C,roof))


#we also add a back wall to reduce the amount of cycles
                                  calculated and make the longest
                                   cycle be born at degree 0 (for
                                   the purpose of comparing with
                                  the bec)

  W4 = np.zeros((B.shape[1],B.shape[2]))
  back = W4.reshape((1,B.shape[1],B.shape[2]))
  B = np.vstack((B,back))
  C = np.vstack((C,back))

#we wish to for the grid points in the upper left corner to be
                                  filled in first, so that we do
                                  not detect rivers or plumes
                                  hitting Bergen before the start
                                   of the month

    for i in range (0, bergen[0]+3): #+3 due to cross-expand
                                       and the roof
      for j in range (0,40):
          B[j,i,:] = 0
            C[j,i,:] = 0

#we poke a hole through the back wall where the wet line is
```

```
  B[B.shape[0]-1,bergen[0]+2,bergen[1]+1] = 100

#we now build filtered complexes from the arrays B and C. The
                                input 'b' gives us complexes
                                built using a bottom-up
                                filtration.

    X = cfc.make2SkeletonFrom3Darray(B,'b')
  Y = cfc.make2SkeletonFrom3Darray(C,'b')

#finally we compute the persistent homology (with Z/2
                                coefficients) and obtain the 1-
                                persistence diagrams of X and Y

  X.computePersistentHomology(1)
  Y.computePersistentHomology(1)

#computePersistentHomology and getDiagram are defined on
                                instances of the class "
                                FilteredComplex" (defined in
                                filteredcomplex.py). The method
                                 make2SkeletonFrom3Darray made
                                X and Y into instances of this
                                class.

  x = X.getDiagram(1)
  y = Y.getDiagram(1)

  print('Persistent homology computed!')

#we remove the points in x that also occur in y

  z = copy.deepcopy(x)

  i = 0
  while i < (len(z.cycles)):
    if z.cycles[i] in y.cycles:
      z.cycles.remove(z.cycles[i])
    else:
      i += 1

#we sort the cycles of z so the longest ones appear first

  z.cycles.sort(key=lambda x: x.death-x.birth, reverse=True)

  return x,y,z
```

Bottleneck distance

```
import pickle
import inout, cfc
import dionysus
import numpy as np

def tbec_bec(month, year):
    tbec = inout.load('Time_diagrams/time_diagram_{:02d}_{:4d}'
                                        .format(month,year))

    bec = inout.load('bec/bec'+str(year))
    timeInterval = len(bec)/12
    bec = bec[(month-1)*timeInterval:month*timeInterval]

    #Compute persistent homology given by bec
    f = cfc.makeCubicalComplex(np.array(bec), 't')
    f.computePersistentHomology(0)
    y = f.getDiagram(0)

    #tbec
    t = [(cycle.birth, cycle.death) for cycle in tbec.cycles]
    tbec_pd = dionysus.PersistenceDiagram(1,t)

    #bec
    b = [(cycle.birth, cycle.death) for cycle in y.cycles]
    bec_pd = dionysus.PersistenceDiagram(1,b)

    print "Cycles in tbec: " + str(len(tbec_pd))
    print "Cycles in bec: " + str(len(bec_pd))
    x = dionysus.bottleneck_distance(tbec_pd,bec_pd)
    return x,y,tbec

def altbec_bec(month, year, modified=False):
    z = pickle.load(open("month"+str(month)+str(year)+".p", "rb
                                        "))

    bec = inout.load('bec/bec'+str(year))
    timeInterval = len(bec)/12
    bec = bec[(month-1)*timeInterval:month*timeInterval]

    #Compute persistent homology given by bec
    f = cfc.makeCubicalComplex(np.array(bec), 't')
    f.computePersistentHomology(0)
    y = f.getDiagram(0)

    #bec
    y.cycles.sort(key=lambda c: c.birth-c.death, reverse=True)
    y.cycles[0].death = 0
    b = [(cycle.death, cycle.birth) for cycle in y.cycles]
    bec_pd = dionysus.PersistenceDiagram(1,b)
```

```
    #altbec
    if modified==True: #shorten the longest cycle to the length
                                    of the longest cycle in
                                    the bec
        z.cycles[0].death = y.cycles[0].birth
    a = [(cycle.birth, cycle.death) for cycle in z.cycles]
    altbec_pd = dionysus.PersistenceDiagram(1,a)

    print "Cycles in altbec: " + str(len(altbec_pd))
    print "Cycles in bec: " + str(len(bec_pd))
    x = dionysus.bottleneck_distance(altbec_pd,bec_pd)
    return x,y,z

def altbec_tbec(month, year):
    z = pickle.load(open("month"+str(month)+str(year)+".p", "rb
                                    "))
    tbec = inout.load('Time_diagrams/time_diagram_{:02d}_{:4d}'
                                    .format(month,year))

    #altbec
    a = [(cycle.birth, cycle.death) for cycle in z.cycles]
    altbec_pd = dionysus.PersistenceDiagram(1,a)

    #tbec
    tbec.cycles.sort(key=lambda c: c.birth-c.death, reverse=
                                    True)
    tbec.cycles[0].death = 0
    t = [(cycle.death, cycle.birth) for cycle in tbec.cycles]
    tbec_pd = dionysus.PersistenceDiagram(1,t)

    print "Cycles in altbec: " + str(len(altbec_pd))
    print "Cycles in tbec: " + str(len(tbec_pd))
    x = dionysus.bottleneck_distance(altbec_pd,tbec_pd)
    return x,z,tbec
```

Appendix B: Bottleneck distances

BOTTLENECK DISTANCES

February 2011
Amount of cycles for bec: 24
Amount of cycles for tbec: 62
Amount of cycles for altbec: 30
Bottleneck distance tbec-bec: 2.270
Bottleneck distance altbec-bec: 2.291
Bottleneck distance altbec-tbec: 2.270

February 2012
Amount of cycles for bec: 21
Amount of cycles for tbec: 158
Amount of cycles for altbec: 22
Bottleneck distance tbec-bec: 1.486
Bottleneck distance altbec-bec: 2.044
Bottleneck distance altbec-tbec: 1.499

March 2011
Amount of cycles for bec: 30
Amount of cycles for tbec: 203
Amount of cycles for altbec: 33
Bottleneck distance tbec-bec: 4.507
Bottleneck distance altbec-bec: 4.507
Bottleneck distance altbec-tbec: 2.535

March 2012
Amount of cycles for bec: 28
Amount of cycles for tbec: 140
Amount of cycles for altbec: 33
Bottleneck distance tbec-bec: 2.507
Bottleneck distance altbec-bec: 2.835
Bottleneck distance altbec-tbec: 2.539

April 2011
Amount of cycles for bec: 29
Amount of cycles for tbec: 306
Amount of cycles for altbec: 32
Bottleneck distance tbec-bec: 3.229
Bottleneck distance altbec-bec: 3.507
Bottleneck distance altbec-tbec: 2.116

April 2012
Amount of cycles for bec: 26
Amount of cycles for tbec: 191
Amount of cycles for altbec: 26
Bottleneck distance tbec-bec: 1.553
Bottleneck distance altbec-bec: 2.442
Bottleneck distance altbec-tbec: 2.442

May 2011
Amount of cycles for bec: 27
Amount of cycles for tbec: 829
Amount of cycles for altbec: 40
Bottleneck distance tbec-bec: 7.091
Bottleneck distance altbec-bec: 7.091
Bottleneck distance altbec-tbec: 4.694

May 2012
Amount of cycles for bec: 28
Amount of cycles for tbec: 78
Amount of cycles for altbec: 30
Bottleneck distance tbec-bec: 1.814
Bottleneck distance altbec-bec: 1.814
Bottleneck distance altbec-tbec: 1.080

June 2000
Amount of cycles for bec: 25
Amount of cycles for tbec: 1015
Amount of cycles for altbec: 27
Bottleneck distance tbec-bec: 5.255
Bottleneck distance altbec-bec: 5.700
Bottleneck distance altbec-tbec: 4.844

June 2001
Amount of cycles for bec: 26
Amount of cycles for tbec: 562
Amount of cycles for altbec: 23
Bottleneck distance tbec-bec: 9.403
Bottleneck distance altbec-bec: 9.403
Bottleneck distance altbec-tbec: 3.901

June 2002
Amount of cycles for bec: 29
Amount of cycles for tbec: 549
Amount of cycles for altbec: 30
Bottleneck distance tbec-bec: 5.869
Bottleneck distance altbec-bec: 5.869
Bottleneck distance altbec-tbec: 1.840

June 2003
Amount of cycles for bec: 27
Amount of cycles for tbec: 778
Amount of cycles for altbec: 29
Bottleneck distance tbec-bec: 6.258
Bottleneck distance altbec-bec: 6.614
Bottleneck distance altbec-tbec: 4.371

June 2004
Amount of cycles for bec: 26
Amount of cycles for tbec: 155
Amount of cycles for altbec: 24
Bottleneck distance tbec-bec: 3.438
Bottleneck distance altbec-bec: 3.438
Bottleneck distance altbec-tbec: 1.903

June 2005
Amount of cycles for bec: 27
Amount of cycles for tbec: 791
Amount of cycles for altbec: 32
Bottleneck distance tbec-bec: 4.803
Bottleneck distance altbec-bec: 6.034
Bottleneck distance altbec-tbec: 3.922

June 2006
Amount of cycles for bec: 27
Amount of cycles for tbec: 345
Amount of cycles for altbec: 31
Bottleneck distance tbec-bec: 4.109
Bottleneck distance altbec-bec: 3.864
Bottleneck distance altbec-tbec: 4.109

June 2007
Amount of cycles for bec: 22
Amount of cycles for tbec: 200
Amount of cycles for altbec: 21
Bottleneck distance tbec-bec: 3.925
Bottleneck distance altbec-bec: 5.212
Bottleneck distance altbec-tbec: 2.108

June 2008
Amount of cycles for bec: 29
Amount of cycles for tbec: 203
Amount of cycles for altbec: 31
Bottleneck distance tbec-bec: 2.931
Bottleneck distance altbec-bec: 3.923
Bottleneck distance altbec-tbec: 2.377

June 2009
Amount of cycles for bec: 28
Amount of cycles for tbec: 154
Amount of cycles for altbec: 30
Bottleneck distance tbec-bec: 2.291
Bottleneck distance altbec-bec: 2.291
Bottleneck distance altbec-tbec: 1.947

June 2010
Amount of cycles for bec: 26
Amount of cycles for tbec: 139
Amount of cycles for altbec: 26
Bottleneck distance tbec-bec: 8.730
Bottleneck distance altbec-bec: 8.730
Bottleneck distance altbec-tbec: 1.634

June 2011
Amount of cycles for bec: 24
Amount of cycles for tbec: 1108
Amount of cycles for altbec: 22
Bottleneck distance tbec-bec: 5.458
Bottleneck distance altbec-bec: 9.445
Bottleneck distance altbec-tbec: 4.578

June 2012
Amount of cycles for bec: 26
Amount of cycles for tbec: 181
Amount of cycles for altbec: 24
Bottleneck distance tbec-bec: 5.373
Bottleneck distance altbec-bec: 5.393
Bottleneck distance altbec-tbec: 1.994

July 2011
Amount of cycles for bec: 27
Amount of cycles for tbec: 468
Amount of cycles for altbec: 29
Bottleneck distance tbec-bec: 6.216
Bottleneck distance altbec-bec: 9.394
Bottleneck distance altbec-tbec: 5.312

July 2012
Amount of cycles for bec: 27
Amount of cycles for tbec: 916
Amount of cycles for altbec: 25
Bottleneck distance tbec-bec: 5.463
Bottleneck distance altbec-bec: 4.393
Bottleneck distance altbec-tbec: 4.908

August 2011
Amount of cycles for bec: 24
Amount of cycles for tbec: 358
Amount of cycles for altbec: 28
Bottleneck distance tbec-bec: 5.991
Bottleneck distance altbec-bec: 7.956
Bottleneck distance altbec-tbec 3.752

August 2012
Amount of cycles for bec: 29
Amount of cycles for tbec: 542
Amount of cycles for altbec: 29
Bottleneck distance tbec-bec: 4.653
Bottleneck distance altbec-bec: 4.924
Bottleneck distance tbec-bec: 3.446

September 2011
Amount of cycles for bec: 30
Amount of cycles for tbec: 339
Amount of cycles for altbec: 39
Bottleneck distance tbec-bec: 5.412
Bottleneck distance altbec-bec: 5.788
Bottleneck distance altbec-tbec: 3.152

September 2012
Amount of cycles for bec: 30
Amount of cycles for tbec: 211
Amount of cycles for altbec: 35
Bottleneck distance tbec-bec: 3.264
Bottleneck distance altbec-bec: 3.260
Bottleneck distance altbec-tbec: 2.377

October 2011
Amount of cycles for bec: 29
Amount of cycles for tbec: 135
Amount of cycles for altbec: 43
Bottleneck distance tbec-bec: 3.569
Bottleneck distance altbec-bec: 2.982
Bottleneck distance altbec-tbec: 3.348

October 2012
Amount of cycles for bec: 31
Amount of cycles for tbec: 450
Amount of cycles for altbec: 37
Bottleneck distance tbec-bec: 2.019
Bottleneck distance altbec-bec: 3.003
Bottleneck distance altbec-tbec: 2.855

November 2011
Amount of cycles for bec: 22
Amount of cycles for tbec: 129
Amount of cycles for altbec: 31
Bottleneck distance tbec-bec: 2.858
Bottleneck distance altbec-bec: 2.806
Bottleneck distance altbec-tbec: 2.295

November 2012
Amount of cycles for bec: 31
Amount of cycles for tbec: 159
Amount of cycles for altbec: 34
Bottleneck distance tbec-bec: 2.189
Bottleneck distance altbec-bec: 2.282
Bottleneck distance altbec-tbec: 1.289

December 2011
Amount of cycles for bec: 32
Amount of cycles for tbec: 175
Amount of cycles for altbec: 32
Bottleneck distance tbec-bec: 1.938
Bottleneck distance altbec-bec: 1.938
Bottleneck distance altbec-tbec: 1.699

December 2012
Amount of cycles for bec: 21
Amount of cycles for tbec: 48
Amount of cycles for altbec: 24
Bottleneck distance tbec-bec: 1.364
Bottleneck distance altbec-bec: 1.972
Bottleneck distance altbec-tbec: 1.063