ssdnm
środowiskowe
studia doktoranckie
z nauk matematycznych

Grzegorz Jabłoński

Uniwersytet Jagielloński

Computing persistent homology of maps

Praca semestralna nr 1

(semestr letni 2011/12)

Opiekun pracy: prof. Marian Mrozek

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Computing persistent homology of maps
Term paper

Grzegorz Jabłoński
Department of Mathematics and Computer Science
Uniwersystet Jagielloński, Kraków, Poland
grzegorz.jablonski@uj.edu.pl

June 21, 2012

Abstract

Persistent homology is an important tool used to analyze data from sampled topological spaces. In this paper we would like to present how one can use this method to examine information about a map acting on the sampled space. First step is to build nested sequence of simplicial complexes, and then find the eigenspace of the function using graph of the function. Lastly we compute persistent homology of the eigenspace.

Acknowledgements

This is joint work with Marian Mrozek and Herbert Edelsbrunner.

1 Introduction

In this section we introduce the notion of persistent homology, its history and applications. We also give some motivation of the problem we are interested in, namely the persistent homology of a map.

1.1 Persistent homology of maps

Examination of data from an experimental source leads to the problem of separating true data from the noise. The problem of denoising data is well known and studied, but often an observer has to decide which points, objects or shapes are noise, and which are not. Persistent homology is a tool from algebraic topology and computer science that helps with the problem of the noise identification. It gives a stable measurement of topological features of data, but the reader has to remember that persistent homology is not about denoising data.

Persistent homology was introduced in [5] in 2002, but the origins and early ideas were published in the last decades of twentieth century [4].

Possible applications of persistent homology range from reconstruction of the root systems of agricultural plants [10] to comparison of the human jaw shape[6].

Theory. Definition of the persistent homology requires some concepts from algebra and topology. First we start with the concept of filtrations.

Definition 1. *Filtered space is a nested sequence of subspaces of a topological space that starts with the empty space and ends with the whole space [5]:* $\mathbb{X}_1 \subset \mathbb{X}_2 \subset \mathbb{X}_3 \subset ... \subset \mathbb{X}_n$

Definition 2. *Persistence module is a sequence from definition 1 with applied homology functor on spaces and inlucsions:* $H(\mathbb{X}_1) \rightarrow H(\mathbb{X}_2) \rightarrow H(\mathbb{X}_3) \rightarrow ... \rightarrow H(\mathbb{X}_n)$

Such module could be split into *indecomposable summands* of the form: $0 \rightarrow F \rightarrow F \rightarrow .. \rightarrow F \rightarrow 0$, where every map is the identity. When summand appears we say that it is the *birth* of the corresponding homology generator, and *death* when the summand disappears in the decomposition. This decomposition is unique, thus we can study the length of the summand, which is called *persistence* of the homology class. It is worth mentioning that persistence makes sense in case of any sequence of vectors spaces connected with homomorphisms. We will exploit that fact in the section 2.2.5. We refer the reader interested in more formal aspects to [2].

Another approach is to define *persistent homology groups* as images of connecting maps [3]. Let us define $f_{i,j} : H(\mathbb{X}_i) \rightarrow H(\mathbb{X}_j)$ as homomorphism induced by the inclusion between spaces. Then homology class γ in $H(\mathbb{X}_i)$ is born at i if it is not in the image of $f_{i-1,i}$, and it dies entering j if $f_{i,j-1}(\gamma)$ is not contained in the image of $f_{i-1,j-1}$ but $f_{i,j}(\gamma)$ is contained in the image of $f_{i-1,j}$.

1.2 Motivation

The persistent homology was not yet used to describe self maps. There are papers such as [3], which approaches the problem of homological features of general maps, but only self maps can be used in dynamical systems. Algorithm presented in this paper will give us a flexible mechanism to study experimental

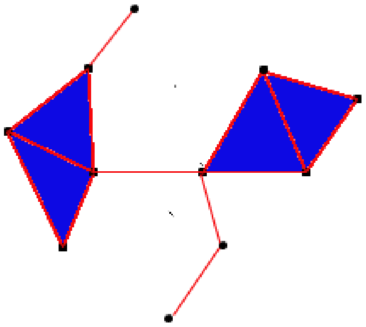

Figure 1: 1-dimensional simplicial complex (black - vertices, red - edges, blue - triangles)

data coming from maps acting on the topological spaces. One of the possible applications is topological analysis of time series.

2 Maps persistence

In this section we propose the way to compute the persistent homology of eigenspaces of some map known only from experimental sampling. In the first part we recall necessary definitions and give the theory for the algorithm described in the second part of the section.

2.1 Definitions

In our setting the sequence from definition 1 consists of simplicial complexes. It is worth noting, that homology of simplicial complex is computed using a chain complex, which is intermediate step between the topological space and homology groups.

Definition 3. *n-simplex is a n-dimensional convex polytope with $n+1$ vertices.*

We will use the shorter name *simplex* for all types of n-simplices.

Definition 4. *Simplicial complex is a set of simplices with following properties:*

- *any face of simplex is also in the complex*
- *the intersection of any two simplices is a face of both simplices*

Simplicial complex is a topological space, which one can imagine as a set of properly glued points, line segments, triangles and their n-dimensional counterparts (see figure 1). Such complex has an advantage of a very easy representation in the computer memory. One can label every simplex with an unique number, and remember all its faces as a list of integer numbers.

We use the following notation:

- $X \subset \mathbb{R}^d$ - topological space,
- $A \subset \mathbb{R}^d$ - finite sampling of X (point cloud).
- $f : X \to X$ - continuous self map
- $\alpha : A \to A$ - point cloud approximation of the map f (we replace values of the map with the closest points in A)
- $G(\alpha) = \{(x, y) \in A \times A | x \in A, y = \alpha(x)\}$ - graph of the function α

We restrict our case to $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$, which is a map on the real plane. We define p and q as projections from graph of the function to the argument and value of the map.

Definition 5.

$$p(x, y) = x$$

$$q(x, y) = y$$

where $\alpha(x) = y$.

The diagram below presents the connection between graph of function, and projections p and q:

$$\begin{tikzcd}
& G = \{(x_1, y_1, x_2, y_2)\} \arrow[dl, "p"'] \arrow[dr, "q"] & \\
A = \{(x_1, y_1)\} \arrow[rr, "\alpha"'] & & A = \{(x_2, y_2)\}
\end{tikzcd}$$

We also introduce the notion of matching matrix. Given linear map between vector spaces, and its matrix we say that matrix is a *matching matrix* if it has at most one one in every row and column. We say that linear map with respect to some bases is *a matching* if the matrix of the map is the matching matrix. Persistent homology could be defined a special case of a matching of homology groups (similar definition was introduced in [2]).

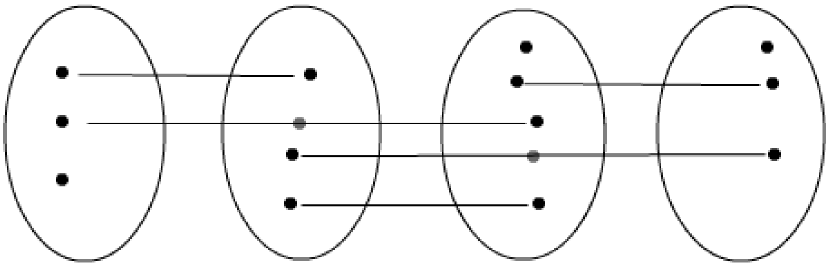

Figure 2: Matching of sets.

2.2 Algorithm

In this section we devise an algorithm to compute the persistence of maps.

2.2.1 Step 1 - Vietoris-Rips complexes

Given a finite point cloud data our goal is to prepare a filtration of simplicial complexes. We decided to use Vietoris-Rips complexes. Vietoris-Rips complex made from finite set of points is a simplicial complex, in which every set of points that has diameter at most ϵ form a simplex. Straightforward computation of Vietoris-Rips complex is time consuming thus we applied the fast algorithm introduced in [11].

Let us denote Vietoris-Rips complex constructed from set of points as $\mathcal{R}_\epsilon(A)$, where ϵ is value of diameter, and A is set of points. First we have to define sequence of diameters $\epsilon_i \in \mathbb{R}, i \in \{0, 1, ..., n\}, 0 < \epsilon_0 < \epsilon_1 < ... < \epsilon_n$. Next we build series of Rips complexes of:

- space A - $\mathcal{R}_{\epsilon_i}(A)$, and
- graph of function α: $\mathcal{R}_{\epsilon_i}(G(\alpha))$.

It is straightforward to observe that for every ϵ_i value we obtain counterparts of projections p and q. What's more, such maps are simplicial. It follows from the fact that distance between points decreases or stays the same after projection onto lower dimensional subspace. Let us call those maps p_i and q_i, and present them on the diagram:

$$\begin{tikzcd}
 & \mathcal{R}_{\epsilon_i}(G(\alpha)) \arrow[dl, "p_i"'] \arrow[dr, "q_i"] & \\
\mathcal{R}_{\epsilon_i}(A) & & \mathcal{R}_{\epsilon_i}(A)
\end{tikzcd}$$

For sequence of increasing real values $0 < \epsilon_0 < \epsilon_1 < ... < \epsilon_n$ we have:

$$\mathcal{R}_{\epsilon_0}(G(\alpha)) \subset ... \subset \mathcal{R}_{\epsilon_n}(G(\alpha)) \tag{1}$$

In the next steps we will use the above nested sequence to compute persistent homology.

2.2.2 Step 2 - Homology

The sequence from equation 1 is connected by natural inclusions, which induce homomorphisms on the homology level:

$$H_*(\mathcal{R}_{\epsilon_0}(G(\alpha))) \xrightarrow{\xi_*^0} ... \xrightarrow{\xi_*^{n-1}} H_*(\mathcal{R}_{\epsilon_n}(G(\alpha)))$$

Moreover, p_i and q_i are simplicial maps if we use the proper metric (euclidean metric is sufficient). The outcome is that we can use p_i and q_i to induce maps in homology. After applying homology functor on Vietoris Rips complexes and maps, we get the following diagram:

$$\begin{tikzcd}
 & H_*(\mathcal{R}_{\epsilon_i}(G(\alpha))) \arrow[dl, "p_*^i"'] \arrow[dr, "q_*^i"] & \\
H_*(\mathcal{R}_{\epsilon_i}(A)) & & H_*(\mathcal{R}_{\epsilon_i}(A))
\end{tikzcd}$$

p_*^i and q_*^i are induced by p_i and q_i.

Homology of the simplicial complex is obtained in two steps:

- In the first step we use the reduction and coredution algorithms from [9]. We used structures and implementation from the software package CAPD::RedHom available from [1].
- homology of the remaining complex is computed via Smith diagonalization of boundary matrices. The detailed information can be found in [7].

2.2.3 Step 3 - Computing map matrices

The third step in our algorithm is the computation of ξ_*^i for $i = 1, .., n-1$, p_*^i and q_*^i for $i = 1, ..n$. We are interested in matrix representation of these maps, so we start by taking an image of base vectors through every map. Then we represent every value of the map in vector bases of the range.

Algorithm 1 Homology maps matrices

```
**Input:** $(v_i)_{i=1,..,n}$ set of chains forming base of the domain, $F$ homology map
**for** every chain $v_i$ from domain base **do**
    compute the value $F(v_i)$
    do reductions and coreductions on $F(v_i)$
    represent $F(v_i)$ in the base of the range
**end for**
```

2.2.4 Step 4 - Eigenspaces

Given $H_*(G_i)$, $H_*(A_i)$ (shorthand for $H_*(\mathcal{R}_{\epsilon_i}(G(\alpha)))$ and $H_*(\mathcal{R}_{\epsilon_i}(A))$, $p^i_*, q^i_* : H_*(G_i) \to H_*(A_i)$ we define λ-*eigenspace* of (p^i_*, q^i_*) as follows:

Definition 6.

$$\bar{E}_i(\lambda, p^i_*, q^i_*) := \{\omega \in H_*(G_i) \,|\, q^i_*(\omega) = \lambda p^i_*(\omega)\}$$

$$E_i(\lambda, p^i_*, q^i_*) := \bar{E}_i(\lambda, p^i_*, q^i_*)/(\bar{E}_i(\lambda, p^i_*, q^i_*) \cap \ker\ p)$$

λ *is called an* eigenvalue *of pair* (p^i_*, q^i_*) *if* $E(\lambda, p^i_*, q^i_*) \neq \{0\}$.

The reason for this form of the definition comes from subsequent explanation: let us assume λ is an eigenvalue of α, p is invertible from the definition, then:

$$\lambda x = \alpha(x)$$

We also know that

$$\alpha(x) = q \circ p^{-1}(x)$$

$$\lambda x = q \circ p^{-1}(x)$$

Assume:

$$\omega = p^{-1}(x)$$

$$p(\omega) = x$$

substitute

$$q(\omega) = \lambda p(\omega)$$

The counterpart in induced homology is

$$q^i_*(x) = \lambda p^i_*(x)$$

.

ker q is computed using function imageKernel from [7]. The same function is used to find the $\bar{E}_i(\lambda, p^i_*, q^i_*) = ker(q^i_* - \lambda p^i_*)$. We use two additional functions quotientBase and map restriction. The former one is used to compute the quotient module. The latter one computes matrix restriction of the function in new bases.

Algorithm 2 Map restriction

```
  **Input:** $A$ - map matrix, $Q$ - new domain base, $R$ - new range base
  **Output:** $B$ - restricted map matrix
  **for** every column $c$ from $Q$ **do**
      $im \leftarrow c * Q$
      represent $im$ in new base (columns of $R$) by solving $R * im = x$
      add new column $x$ to $B$
  **end for**
```

Note that we also need the function to compute intersection of two vector spaces.

Algorithm 3 Space intersection

```
  **Input:** bases of vector space $A$ and $B$ represented as columns of matrices
  **Output:** matrix $C \leftarrow A \cap B$ representing vector basis in columns
  $n \leftarrow numberOfColumns(A)$
  $D \leftarrow A|B$ (we glue two matrices)
  $k \leftarrow$ first n rows of $kernel(D)$
  $C = A * k$ (matrix multiplication)
```

2.2.5 Step 5 - Matching bases

Since every eigenspace is subgroup of $H_*(G_i)$, by studying restriction of ξ^i_* to $E_i(\lambda, p^i_*, q^i_*)$ we get information about persistence of every eigenvector (homology generator) of $H_*(G_i)$. The following diagram explains how maps act on the homology groups:

$$\begin{tikzcd}
{} \arrow[r] & H_*(G_i) \arrow[rrr, "\xi^i_*"] \arrow[dl, "p^i_*"'] \arrow[dr, "q^i_*"] & & & H_*(G_{i+1}) \arrow[r, "\xi^{i+1}_*"] \arrow[dl, "p^{i+1}_*"'] \arrow[dr, "q^{i+1}_*"] & {} \\
H_*(A_i) & & H_*(A_i) & H_*(A_{i+1}) & & H_*(A_{i+1})
\end{tikzcd}$$

From every homology group $H_*(G_i)$ we could extract the subgroup $E_i(\lambda, p^1_*, q^1_*)$. Now we study the persistent homology of sequence of vector spaces: $E_1(\lambda, p^1_*, q^1_*) \subset E_2(\lambda, p^1_*, q^2_*) \subset ...$ connected by ξ^i_*.

Last step of the algorithm is to find bases in which every matrix in the sequence made of $(\xi^i_*)_{i=1,..,n-1}$ matrices is a matching matrix. Before demonstrating the algorithm we state a few important facts. Let us assume we have the linear mapping f represented as matrix A. Vector bases of the domain and the range are stored as columns respectively in matrices Q and R. We want to find bases in which matrix A is a matching. Elementary operations employed in the algorithm:

- column operations (exchange, add, multiply) - changes are made only to columns of A and Q. Every operation on A is mirrored as the same operation on columns of Q.
- row operations (exchange, add, multiply) - changes are made only to rows of A and R, however in this case we do inverse operations on rows of R. Assume we add ith column to jth in matrix A, it corresponds to subtracting jth row from ith in matrix R.

Matching algorithm is used to find the persistence of vectors forming $E_i(\lambda, p^i_*, q^i_*)$ and connected by $(\xi^i_*)_{i=1,..,n-1}$. We assume columns of $E_1, E_2, .., E_n$ represent bases, and $\xi^i : E_i \to E_{i+1}$. Pseudo-code of the method is presented in algorithm 4. Parameters for *rowAdd*, *columnAdd* are: the matrix, affected column/row, added column/row, multiplication of added column/row.

Some properties of the algorithm:

- in the first for-loop when applying column operations to ξ_i matrix, we have to change the basis E_i and respectively ξ_{i-1}. Such changes, however, do not violate column echelon form of the previously reduced matrices $\xi^{i+1}, \xi^{i+2}, .., \xi^{n-1}$.
- in the second for-loop the only necessary row operation is addition. Let us assume we add ith row multiplied by γ to the jth in ξ_i, where $i < j$ (it is the only possibility due to the column echelon form). This operation is represented as adding jth column multiplied by $-\gamma$ to column ith in ξ^{i+1} and E_{i+1}. As stated we add jth column, which is on the left side of the ith column, thus we preserve the column echelon form of ξ^{i+1}.

The running time of the algorithm is $O(nm^3)$, where m is the maximum of all dimensions of all ξ matrices. Base matrices E_i are necessary for column and row operations. The persistent homology is encoded in ξ matrices.

3 Experiments

We conducted three types of experiments, every based on different mapping. We begin with generating n equally distributed points on the circle. Then we add two dimensional noise with Gaussian distribution. In the last step we compute value of the function in these points. To overcome the problem with different range and domain, the value is exchanged with the nearest point from A. We present the results only for the $z \longrightarrow z^2$ map (omitted are maps $z \longrightarrow -z$, $z \longrightarrow \bar{z}$).

Due to a computational problems we used the coefficient from the field $\mathbb{Z}_{1009}$.

3.1 $z \longrightarrow z^2$

First map is $f : z \longrightarrow z^2$, which in the homology doubles the generator of the circle.

We show the results of our algorithm as the sequence of persistence barcodes for increasing level of an additive noise. The outcome is visible in figures: 3, 4 and 5. In the top of figures we see persistence barcodes of eigenvectors for zeroth and first dimension. In the bottom we show eigenvectors of the first homology group.

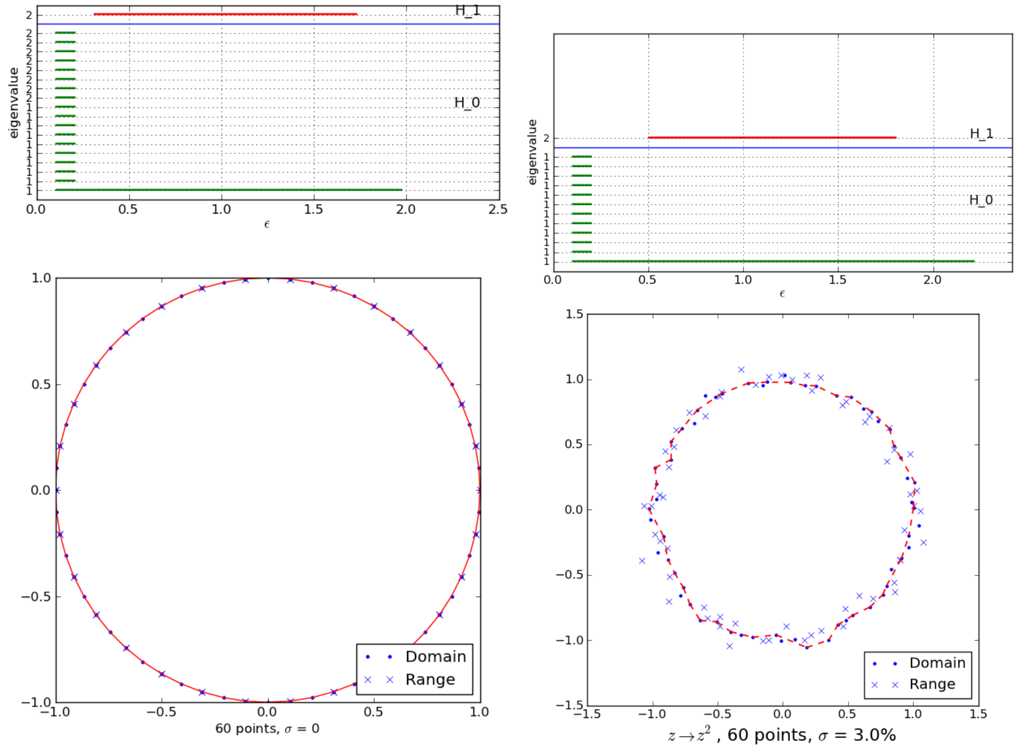

Figure 3: Persistence barcode: left - noise level 0% of radius, right - noise level 3% of radius

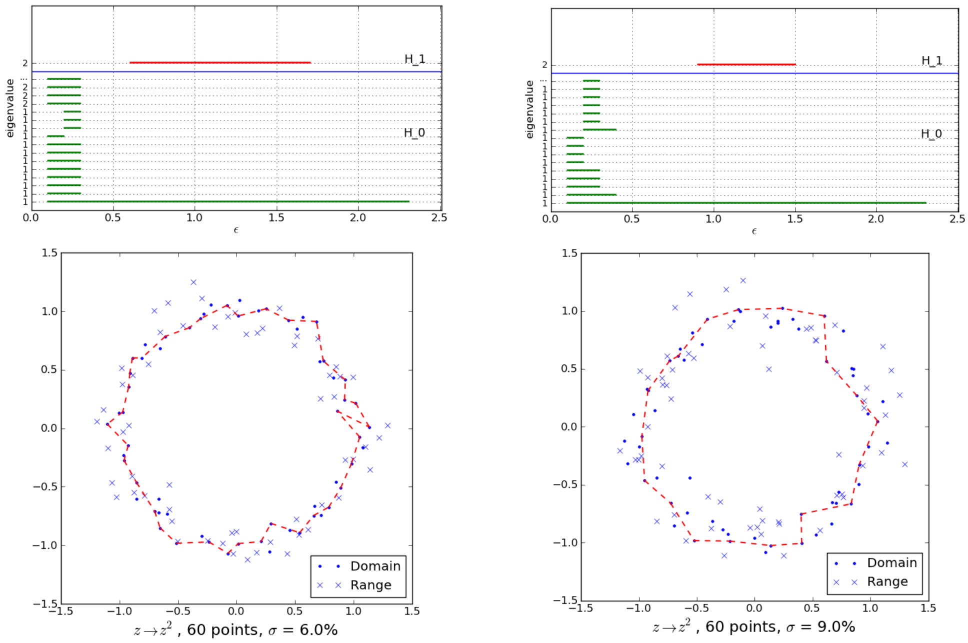

Figure 4: Persistence barcode: left - noise level 6% of radius, right - noise level 9% of radius

3.2 Technical details

The algorithm was implemented in C++ language as a part of the CAPD::RedHom [1] library. Points and noise were generated using the Python 2.6 language. Computation of one persistence barcode for 60 points took about 3 minutes on Intel Core 2 Duo CPU 2.26GHz with 2GB RAM.

3.3 Discussion

As expected we get only one eigenvector in the first homology group for the eigenvalue 2. It corresponds to the homology generator of the circle. Surprisingly for some parameters of ϵ we have all elements of $\mathbb{Z}_{1009}$ being eigenvalue. This fact was observed in complex numbers in the article [8] and comes from generalized eigenvalue problem.

Nevertheless the information about expected eigenvalue is visible even with very strong noise. Our approach allows us to reproduce the homology generators on every level. Drawback of such attempt is long running time of the whole process.

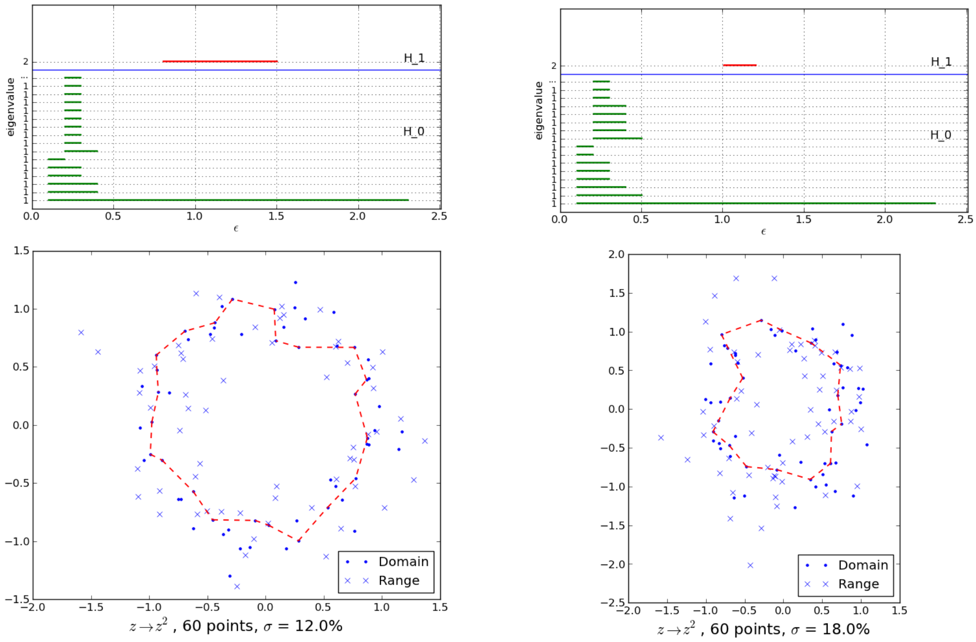

Figure 5: Persistence barcode: left - noise level 12% of radius, right - noise level 18% of radius

4 Future work

We see the potential applications of this algorithm in topological time series analysis. We also want to develop a faster method to compute the persistence of eigenvalues based on the classical persistent homology algorithm via reduction of sorted the boundary matrix.

References

[1] Capd::redhom website. `http://redhom.ii.uj.edu.pl/`, June 2012.

[2] Gunnar Carlsson and Vin De Silva. Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):32, 2008.

[3] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Dmitriy Morozov. Persistent homology for kernels, images, and cokernels. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1011–1020, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[4] Herbert Edelsbrunner and John Harer. Persistent homology – a survey. *Contemporary Mathematics*, 2008.

[5] Herbert Edelsbrunner, Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. 10.1007/s00454-002-2885-2.

[6] Jennifer Gamble and Giseon Heo. Exploring uses of persistent homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, 101(9):2184 – 2199, 2010.

[7] Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational Homology (Applied Mathematical Sciences)*. Springer, 1 edition, 2004.

[8] C B Moler and G W Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, 1973.

[9] Marian Mrozek and Bogdan Batko. Coreduction homology algorithm. *Discrete Comput. Geom.*, 41(1):96–118, January 2009.

[10] Ying Zheng, Steve Gu, Herbert Edelsbrunner, Carlo Tomasi, and Philip Benfey. Detailed reconstruction of 3d plant root shape, 2011.

[11] Afra Zomorodian. Fast construction of the vietoris-rips complex. *Computers & Graphics*, 34(3):263 – 271, 2010. Shape Modelling International (SMI) Conference 2010.

Algorithm 4 Matching base algorithm

```
Input: $E_1, E_2, .., E_n$ basis matrices, $\xi^1, \xi^2, .., \xi^{n-1}$ map matrices
for $k \leftarrow n-1$ to 1 do ▷ compute the column echelon form of $\xi^i$ using only column operations
    $N \leftarrow numberOfColumns(\xi^k)$; $M \leftarrow numberOfRows(\xi^k)$
    $pivotCol \leftarrow 1$
    for $i \leftarrow 1$ to $M$ do
        $l \leftarrow pivotCol$
        while $\xi^k[i,l] = 0$ and $l \leq N$ do        ▷ find first nonzero entry in row
            $l \leftarrow l + 1$
        end while
        if $l > N$ then                                   ▷ only zeros in the current row
            continue
        end if
        $columnExchange(\xi^k, pivotCol, l)$
        $columnExchange(E_k, pivotCol, l)$
        $rowExchange(\xi^{k-1}, pivotCol, l)$
        for $j = pivotCol + 1$ to $N$ do                          ▷ reduce $i$th row
            $\gamma = -\xi^k[i,j]/\xi^k[i, pivotCol]$
            $columnAdd(\xi^k, j, pivotCol, \gamma)$
            $columnAdd(E_k, j, pivotCol, \gamma)$
            $rowAdd(\xi^{k-1}, pivotCol, j, -\gamma)$
        end for
        $pivotCol \leftarrow pivotCol + 1$
    end for
end for
for $i \leftarrow 1$ to $n-1$ do  ▷ using row addition reduce entries below the pivot positions in $\xi^i$
    $N \leftarrow numberOfColumns(\xi^k)$;$M \leftarrow numberOfRows(\xi^k)$
    for $i \leftarrow 1$ to $M$ do
        $pivotCol \leftarrow$ column of pivot position in row $i$
        for $j = i + 1$ to $M$ do                      ▷ reduce $pivotCol$ column
            $\gamma = -\xi^k[j, pivotCol]/\xi^k[i, pivotCol]$
            $rowAdd(\xi^k, j, i, \gamma)$
            $columnAdd(E_k, i, j, -\gamma)$
            $columnAdd(\xi^{k+1}, i, j, -\gamma)$
        end for
    end for
end for
```