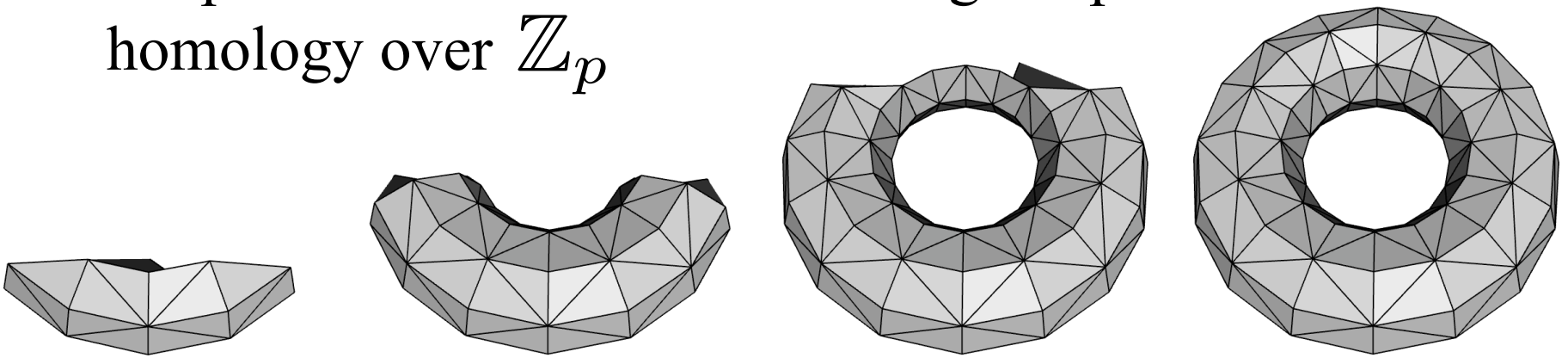
JPlex
Software Demonstration

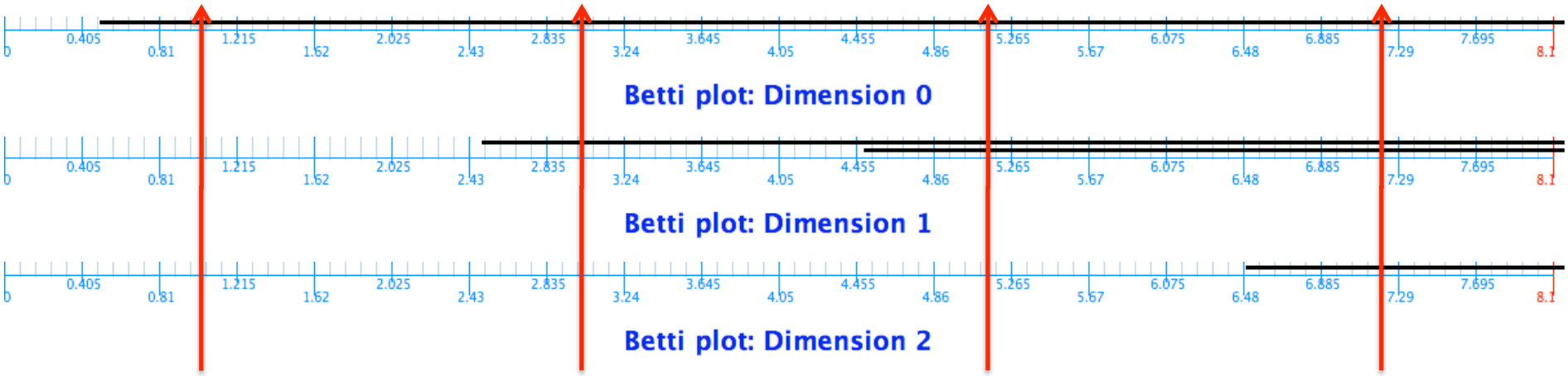
AMS Short Course on Computational Topology
New Orleans
Jan 4, 2011

Henry Adams
Stanford University

What does JPlex do?

- Input: a filtered simplicial complex or finite metric space
- Output: a Betti barcode describing the persistent homology over $\mathbb{Z}_p$



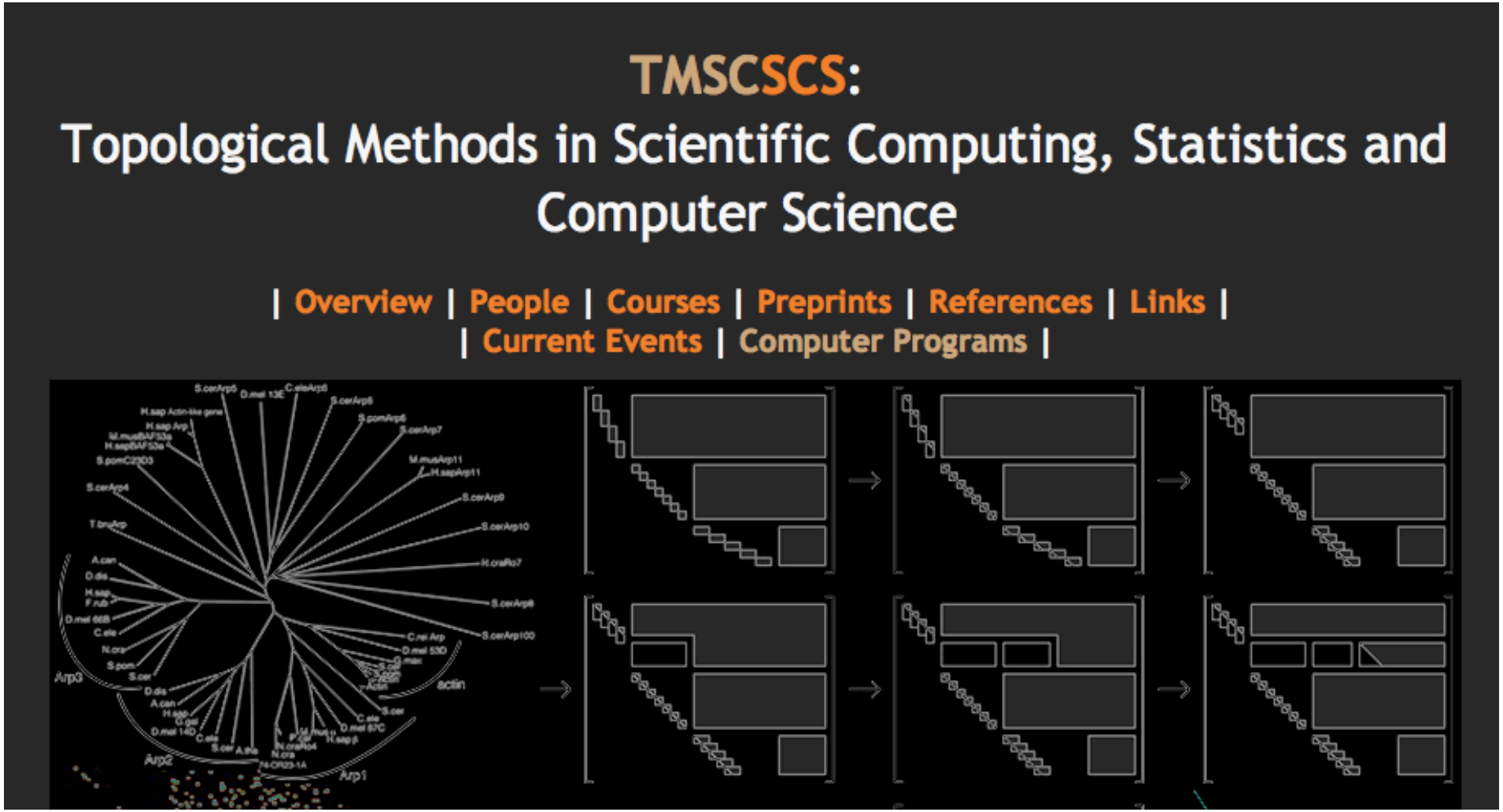


What does JPlex do?

- Input: a filtered simplicial complex or finite metric space
- Output: a Betti barcode describing the persistent homology over $\mathbb{Z}_p$
- Java software
- Matlab or standalone interface
- By Harlan Sexton and Mikael Vejdemo-Johansson. Previous version by Vin de Silva and Patrick Perry
- Algorithm from *Computing Persistent Homology* by Afra Zomorodian and Gunnar Carlsson (2005)

Getting started

- Download from http://comptop.stanford.edu/



Getting started

- Download from http://comptop.stanford.edu/
- Javadoc (follow links)

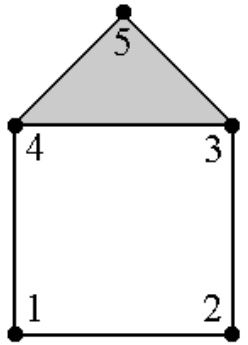
Package edu.stanford.math.plex

Class Summary	
BCPlot	The `BCPlot` class does simple plotting of PersistenceIntervals.
Chain	A `Chain` instance is an element of the module constructed by taking formal sums of ring elements times simplices.
CRC	The `CRC` class provides good hash methods for int/long and string data.
DiscreteSpace	The `DiscreteSpace` class implements a finite discrete metric space.
DistanceData	The `DistanceData` class is the simplest implementation of NSpace.
EuclideanArrayData	The `EuclideanArrayData` class is the simplest implementation of NSpace.
ExplicitStream	A `ExplicitStream` instance is SimplexStream whose elements are given explicitly, along with associated values of a persistence parameter.
ExplicitStream.DComplex	A DComplex (for Dimensional Complex) is the set of simplices (and associated persistence parameters) of specified dimension for an ExplicitStream.
LazyWitnessStream	A LazyWitnessStream is a SimplexStream whose elements are the simplices of the lazy Witness complex of a PointData instance.
MappedBufferData	The `MappedBufferData` class is another very simple implementation of NSpace.

Getting started

- Download from http://comptop.stanford.edu/
- Javadoc (follow links)
- Tutorials: toy examples, exercises, real data

3.3. **Subclass ExplicitStream and persistent homology.** Let's build a stream with nontrivial filtration times. We build a house, with the square appearing at time 0, the top vertex at time 1, the roof edges at times 2 and 3, and the roof 2-simplex at time 7.




```
>> house=ExplicitStream;
>> house.add([1;2;3;4;5], [0;0;0;0;1])
>> house.add([1,2;2,3;3,4;4,1;3,5;4,5], [0;0;0;0;2;3])
>> house.add([3,4,5], 7)
```

We compute the Betti intervals.

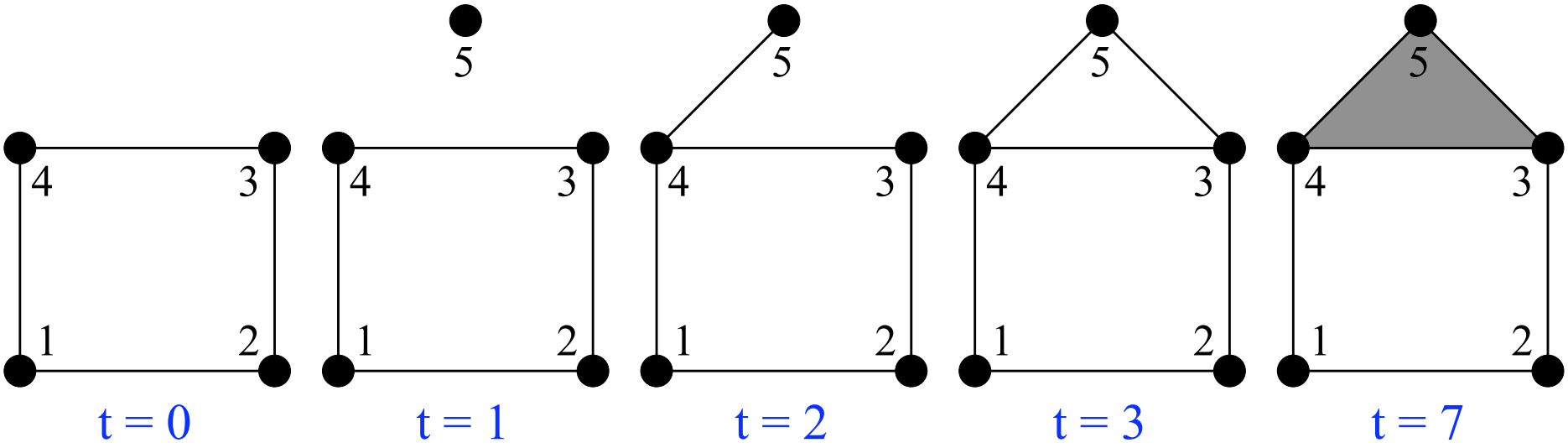
```
>> house.close
>> intervals=Plex.Persistence.computeIntervals(house);
```

There exist other options besides JPlex:

- CHomP
- CGAL
- Dionysus

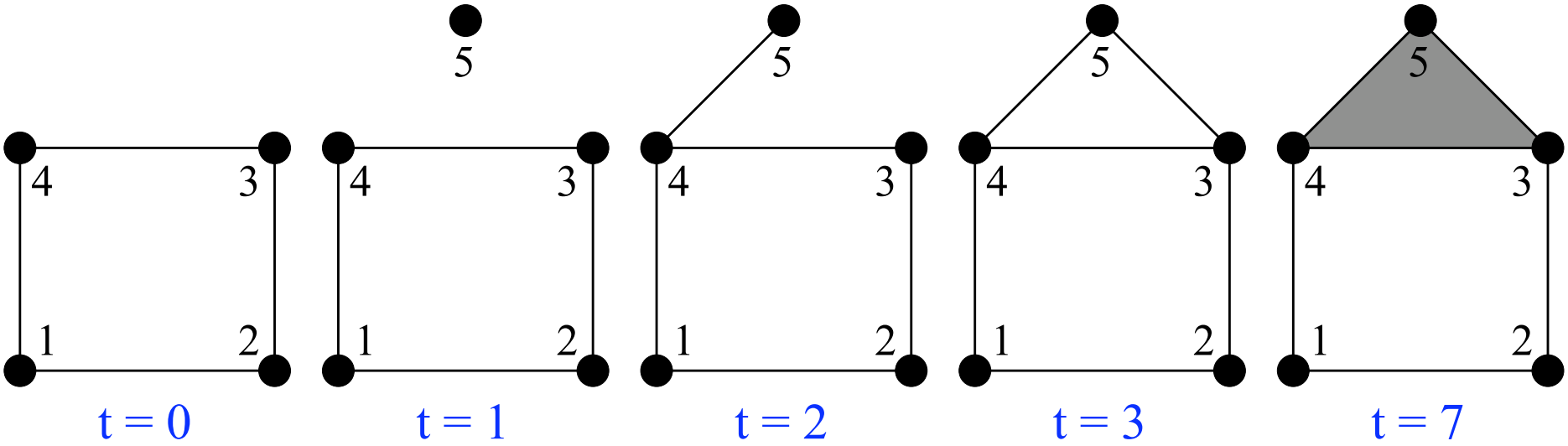
Four JPlex examples

1. Toy filtration



Four JPlex examples

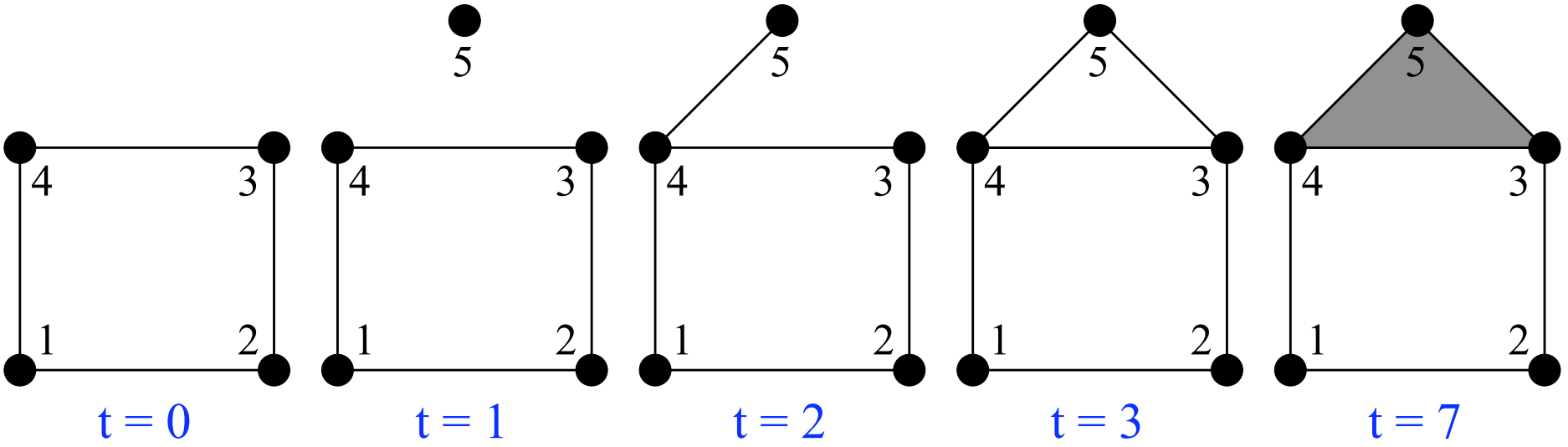
1. Toy filtration

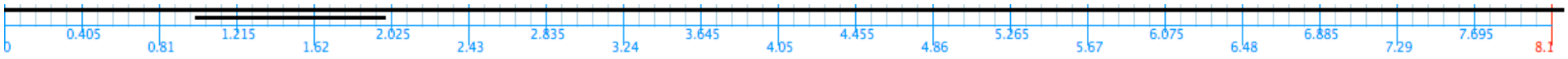



```
house = ExplicitStream;
house.add([1;2;3;4;5], [0;0;0;0;1])
house.add([1,2; 2,3; 3,4; 4,1; 3,5; 4,5], [0;0;0;0;2;3])
house.add([3,4,5], 7)
house.close
intervals = Plex.Persistence.computeIntervals(house);
Plex.plot(intervals, 'Barcode plot', 8)
```

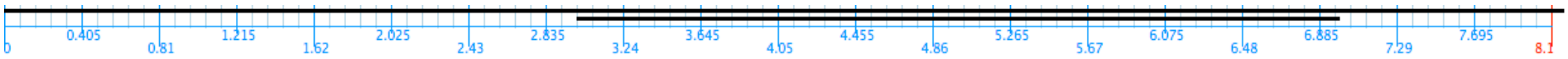
Four JPlex examples

1. Toy filtration





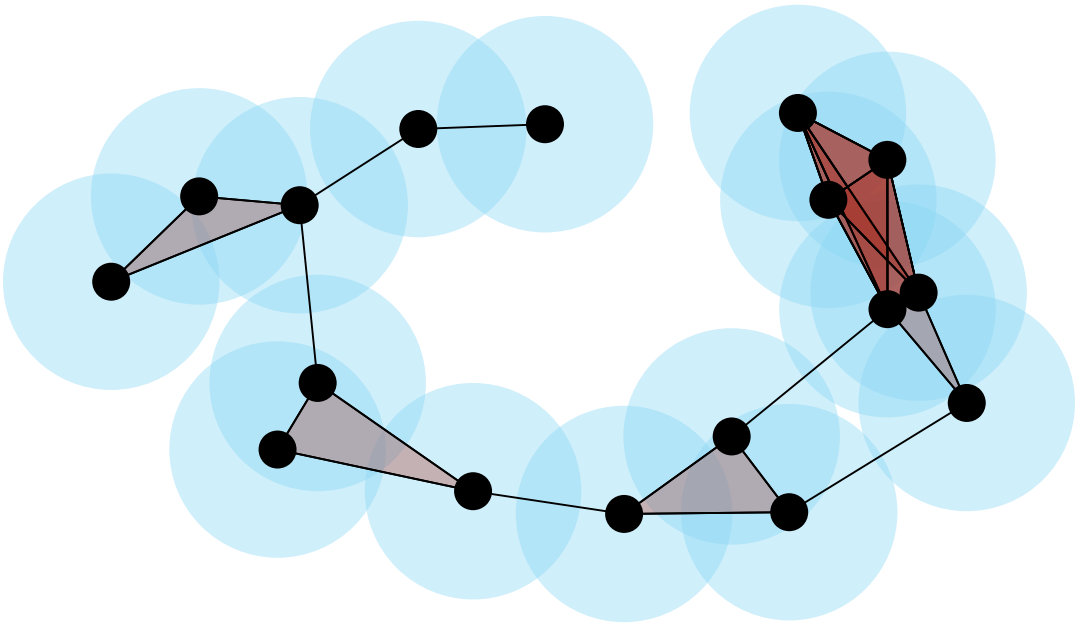
Barcode plot: Dimension 0



Barcode plot: Dimension 1

Four JPlex examples

2. Vietoris-Rips filtration on torus



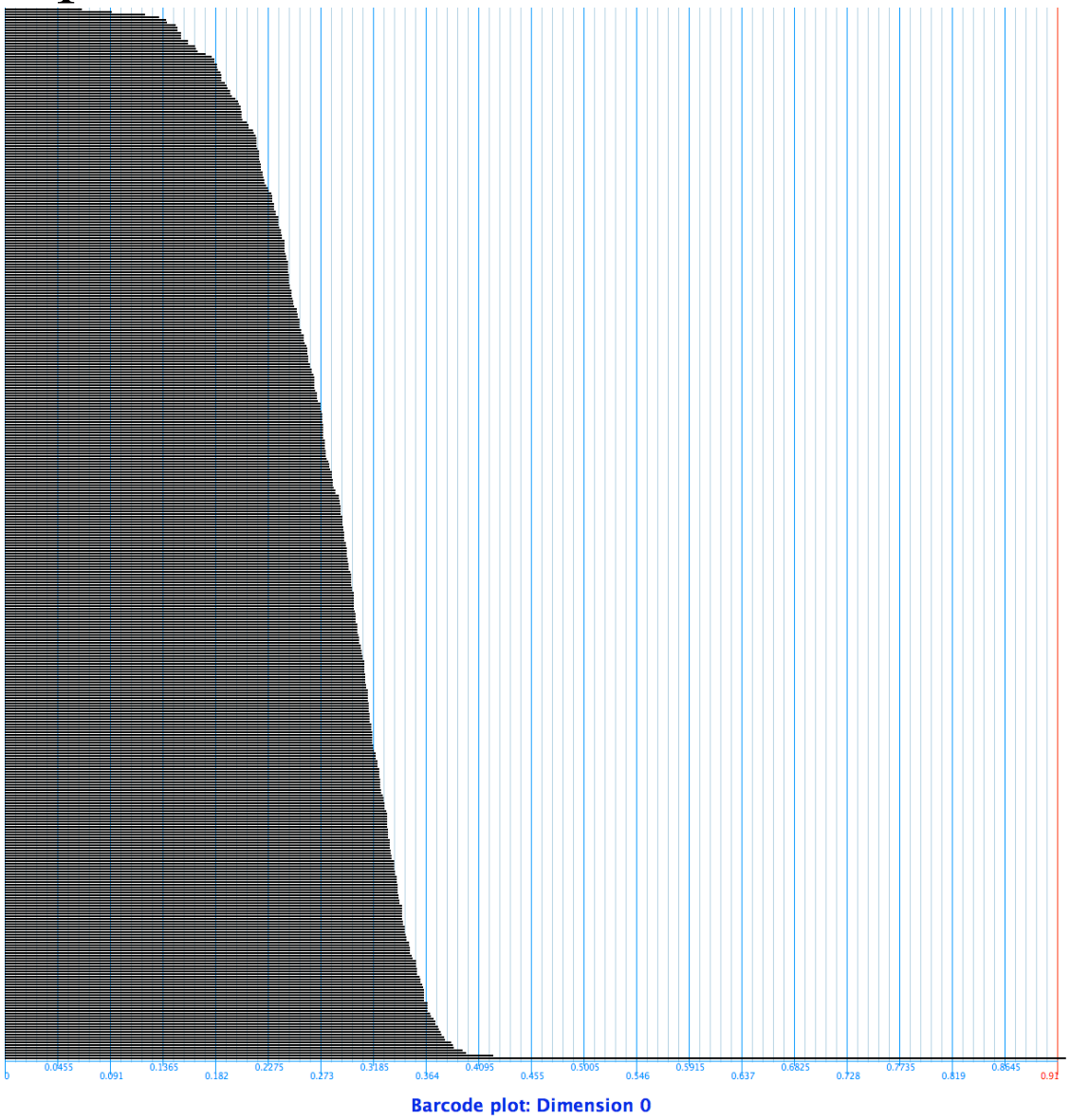
Four JPlex examples

2. Vietoris-Rips filtration on torus

```
points = pointsTorus(20);
size(points)                                          % 400 by 4
pdata = EuclideanArrayData(points);
rips = Plex.RipsStream(0.001, 3, 0.9, pdata);
intervals = Plex.Persistence.computeIntervals(rips);
Plex.plot(intervals, 'Barcode plot', 0.9)
```

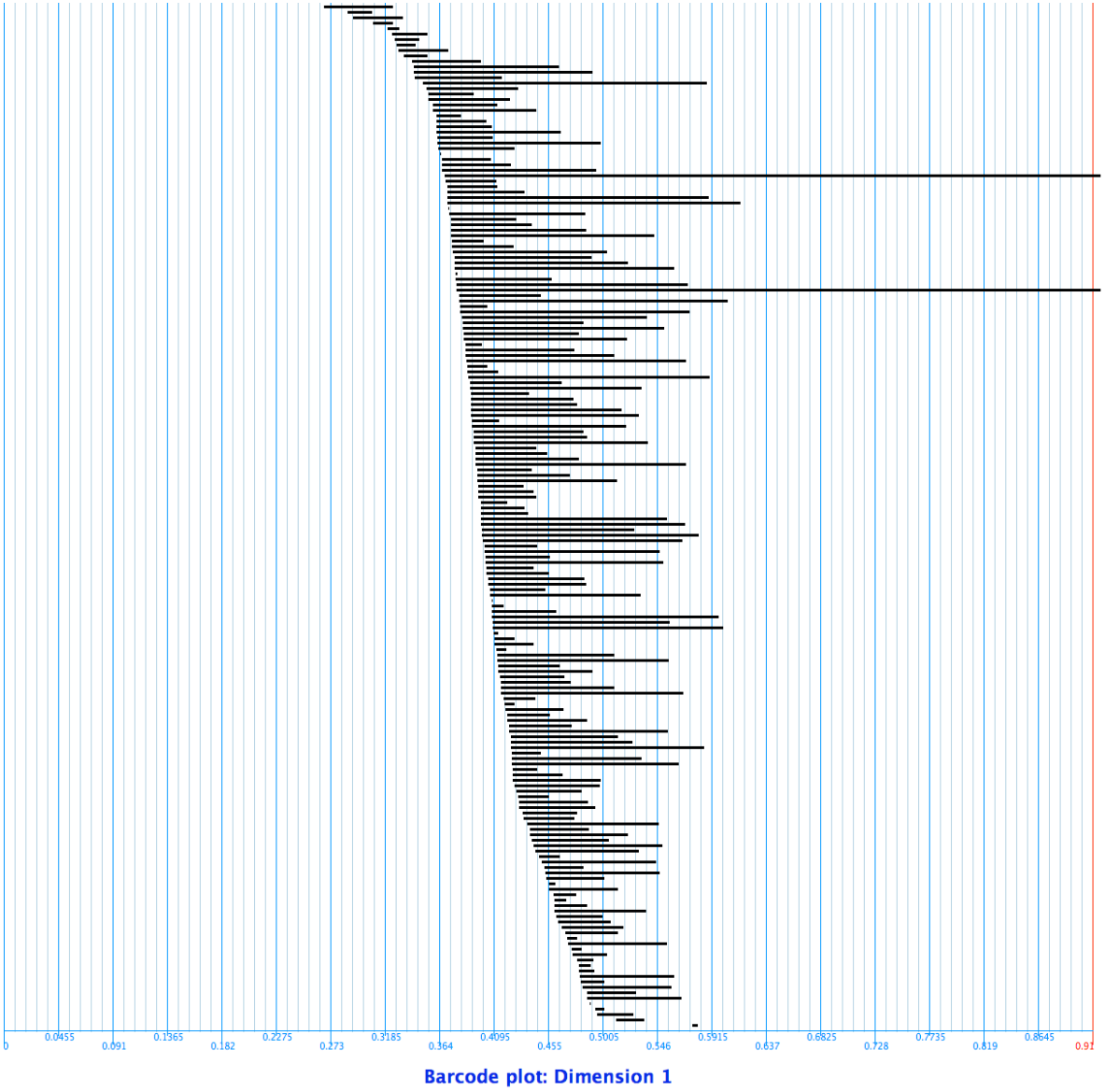
Four JPlex examples

2. Vietoris-Rips filtration on torus



Four JPlex examples

2. Vietoris-Rips filtration on torus



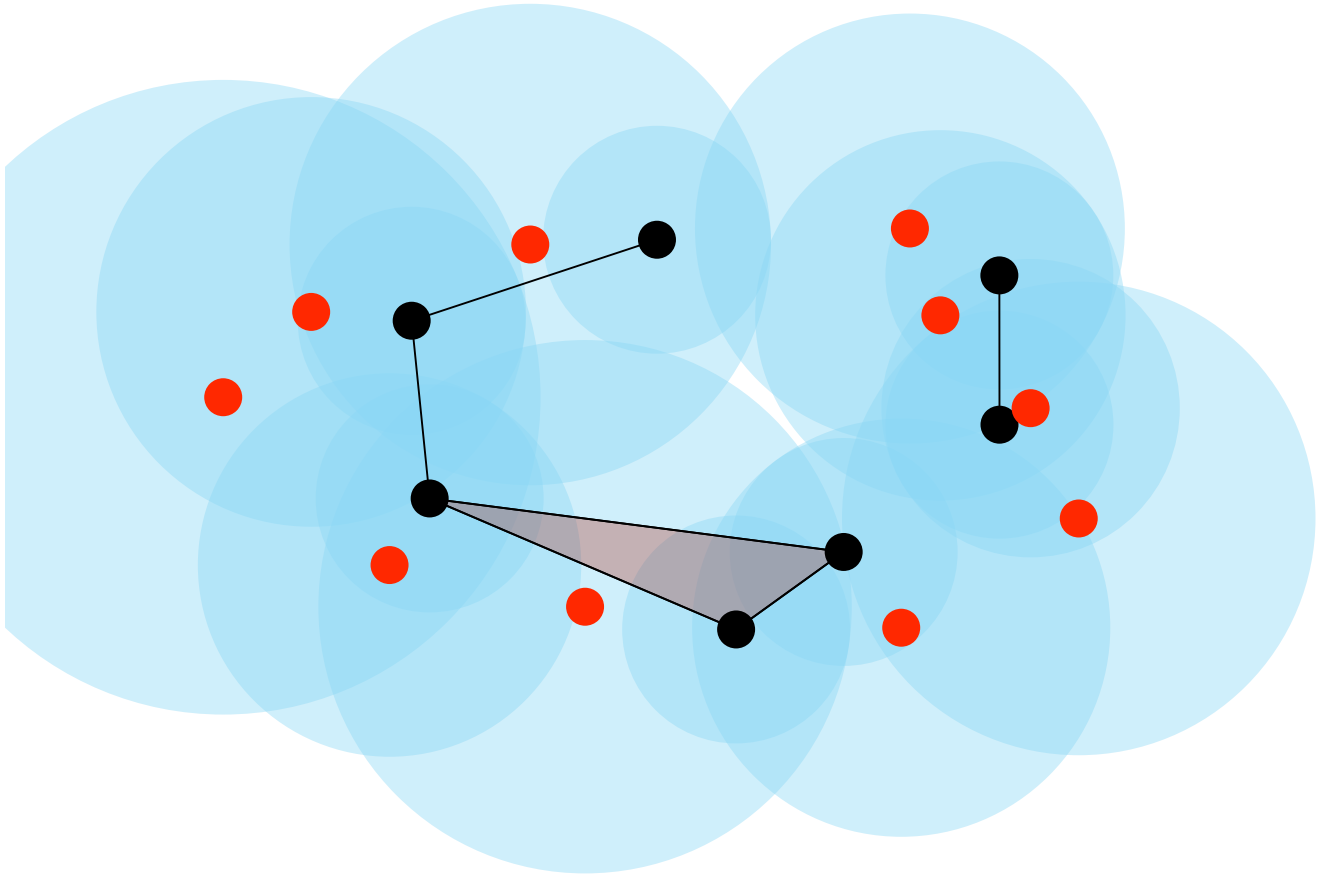
Four JPlex examples

2. Vietoris-Rips filtration on torus

Barcode plot: Dimension 2

Four JPlex examples

3. Witness filtration on torus



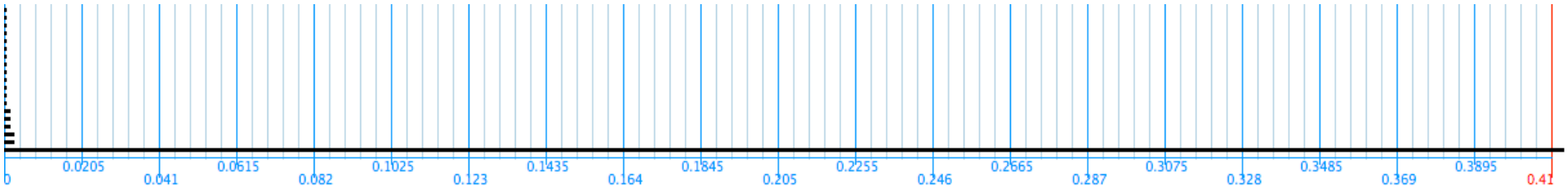
Four JPlex examples

3. Witness filtration on torus

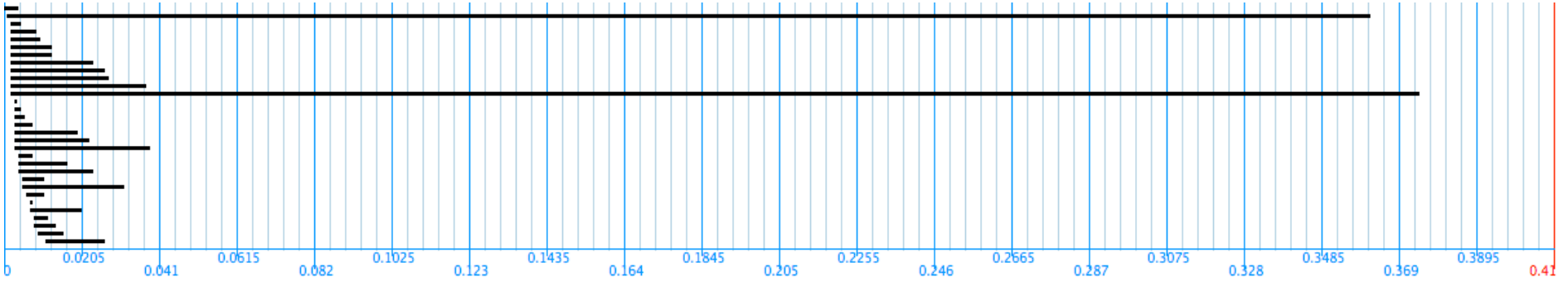
```
points = pointsTorus(100);
pdata = EuclideanArrayData(points);
L = maxminLandmarks(points, 50, 'e');
witness = Plex.LazyWitnessStream(0.001, 3, 0.4, 1, L, pdata);
intervals = Plex.Persistence.computeIntervals(witness);
Plex.plot(intervals, 'Barcode plot', 0.4)
```

Four JPlex examples

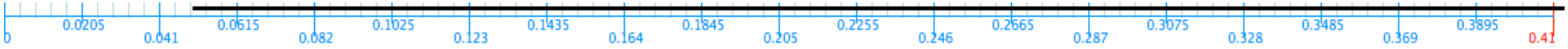
3. Witness filtration on torus



Barcode plot: Dimension 0



Barcode plot: Dimension 1



Barcode plot: Dimension 2

Four JPlex examples

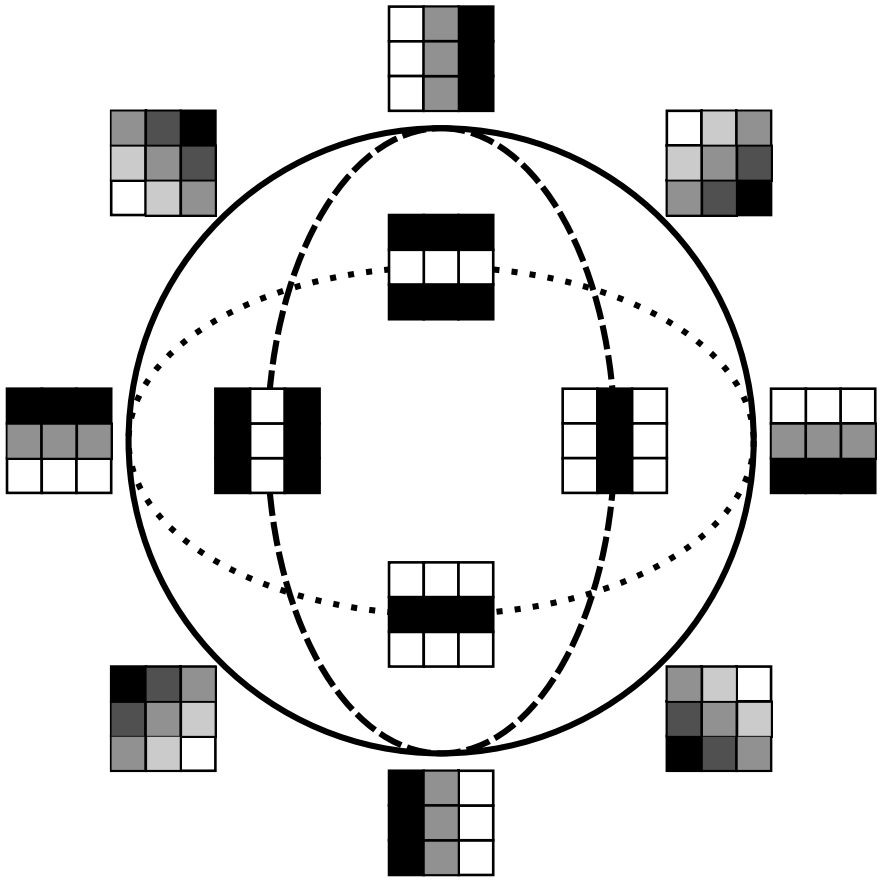
3. Witness filtration on torus

```
rips.size                    % 83,175 simplices
witness.size                 % 3,047 simplices
```

The witness complex has many fewer simplices than the Vietoris-Rips complex (not surprisingly so, as it has only 50 0-simplices).

Four JPlex examples

4. Three circle model from natural images



Four JPlex examples

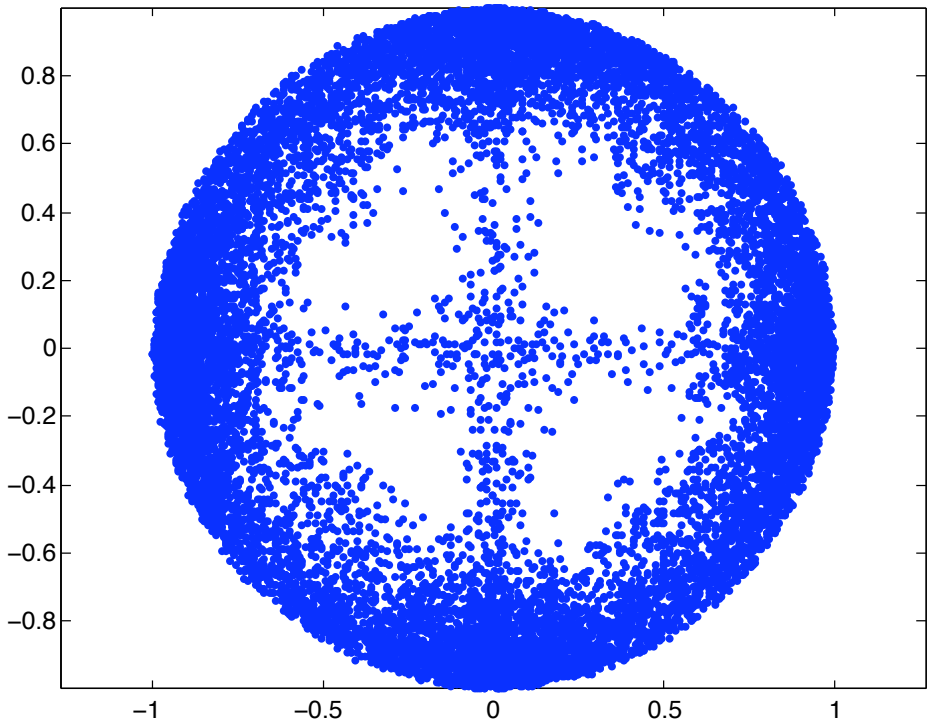
4. Three circle model from natural images

```
load pointsNatural
plot(pointsNatural(:,1), pointsNatural(:,2), '.'), axis equal
```

Four JPlex examples

4. Three circle model from natural images

```
load pointsNatural
plot(pointsNatural(:,1), pointsNatural(:,2), '.'), axis equal
```



Four JPlex examples

4. Three circle model from natural images

```
pdata = EuclideanArrayData(pointsNatural);
L = maxminLandmarks(pointsNatural, 50, 'e');
witness = Plex.LazyWitnessStream(0.001, 3, 0.15, 1, L, pdata);
intervals = Plex.Persistence.computeIntervals(witness);
Plex.plot(intervals, 'Barcode plot', 0.15)
```

Four JPlex examples

4. Three circle model from natural images

```
pdata = EuclideanArrayData(pointsNatural);
L = maxminLandmarks(pointsNatural, 50, 'e');
witness = Plex.LazyWitnessStream(0.001, 3, 0.15, 1, L, pdata);
intervals = Plex.Persistence.computeIntervals(witness);
Plex.plot(intervals, 'Barcode plot', 0.15)
```