Persistent Cohomology and Circle-valued coordinates

Dmitriy Morozov Vin de Silva
Mikael Vejdemo-Johansson

July 6, 2009

Outline

Coordinatization

Essentially

It's all about finding *coordinate function* on a dataset $X \subseteq \mathbb{R}^d$.
Preferably few coordinates - cognitive tools.

Coordinatization

Essentially

It's all about finding *coordinate function* on a dataset $X \subseteq \mathbb{R}^d$. Preferably few coordinates - cognitive tools.

Classically

- Linear coordinatization: Find maps $X \to \mathbb{R}$, concentrating information.
- Principal Component Analysis
- Projection pursuit

Coordinatization

Essentially

It's all about finding *coordinate function* on a dataset $X \subseteq \mathbb{R}^d$. Preferably few coordinates - cognitive tools.

Classically

- Linear coordinatization: Find maps $X \to \mathbb{R}$, concentrating information.
- Principal Component Analysis
- Projection pursuit

Recently

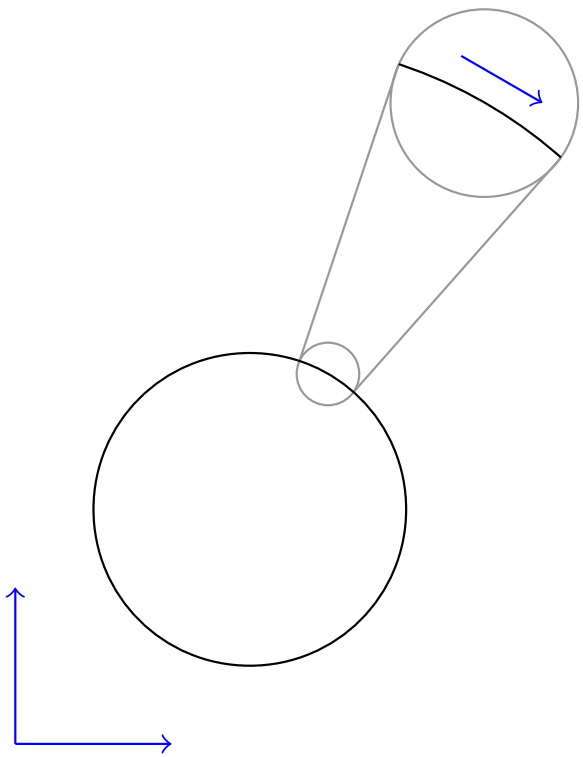
- Nonlinear methods: drop expectation that for f coordinate: $f(\lambda x + \mu y) = \lambda f(x) + \mu f(y)$.
- MDS, Kernel methods, Locally linear methods

Problematic cases

Some shapes take up too many coordinates.

Problematic cases

Some shapes take up too many coordinates.



Locally 1-dimensional.
Globally 2 coordinates needed to describe all points.
The shape doesn't fit in $\mathbb{R}$.

Fixes

How can we fix this?

Circle-valued coordinates

- Use $S^1 = [0,1]/(0 \sim 1)$ as additional coordinate space
- Fixes the circle
- Fixes the torus
- Occurs naturally:
  - Phase coordinates for waves
  - Angle coordinates for directions

Outline

Circle-valued coordinates

Problem remains: how do we find circle-valued coordinates?

Circle-valued coordinates and cohomology

Problem remains: how do we find circle-valued coordinates?

Persistent cohomology

- Degree one cohomology equivalent to circle-valued maps
- Persistence picks out relevant features from noise
- Once a feature-rich parameter has been found, we can work in ordinary (non-persistent) cohomology theories

The algorithm we use is a variation on the Persistence algorithm. We introduce simplices one after the other, and reduce a cumulative coboundary matrix.

From cohomology to circle-valued coordinatizations

Use canonical isomorphism

$$H^1(X;\mathbb{Z}) \cong [X, S^1]$$

Issues

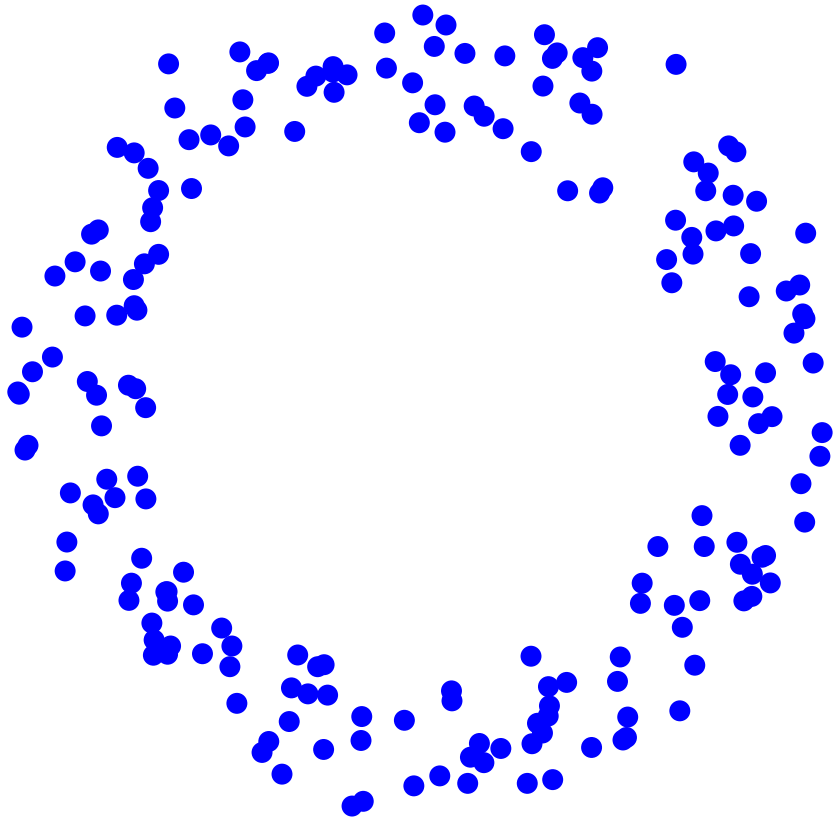
- Easy to compute: Modular cohomology, coefficients in $\mathbb{F}_p$ for small primes p.
  Need for the isomorphism: Integer-valued cohomology.
  Smoothness: Integer cohomology gives constant values on all vertices, and wraps edges in the complex around the target circle.
- Numerical stability of cohomology computation and of the smoothing operations.

Smoothing

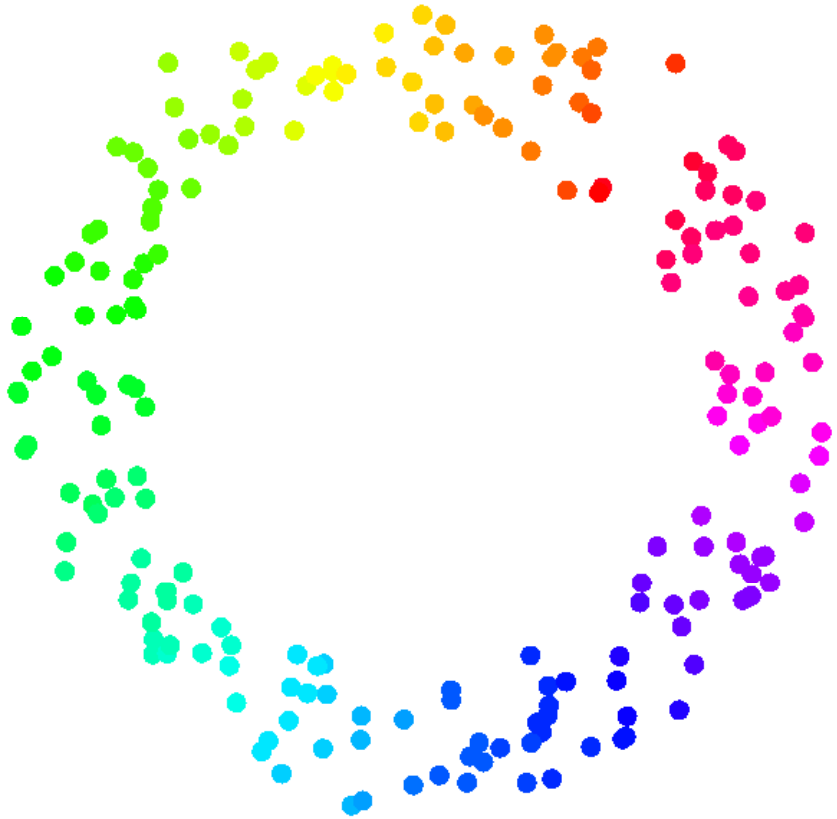
- Integral 1-cocycle: integer weighted edge graph.
- Coordinates found by edge traversal, increasing by edge weights.
- Each cohomologous cocycle guaranteed by cocycle condition to give compatible values, mod 1.0, to each vertex.
- Application straight on integral cocycle yields value 0 at each vertex.
- Given ζ integral cocycle, we wish to find cohomologous cocycle z such that the edges are small.
- Hence, we wish to find x such that $\zeta + \partial x$ has minimal L_2-norm.
- This is a well-known optimization problem. We use the LSQR algorithm.

Outline

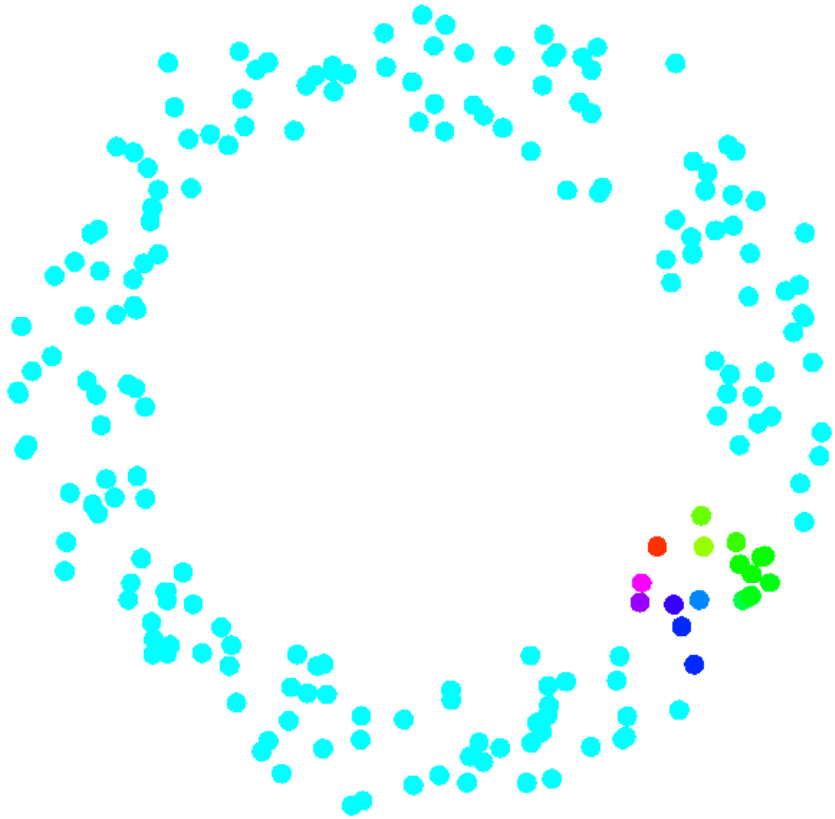
Parametrized circles



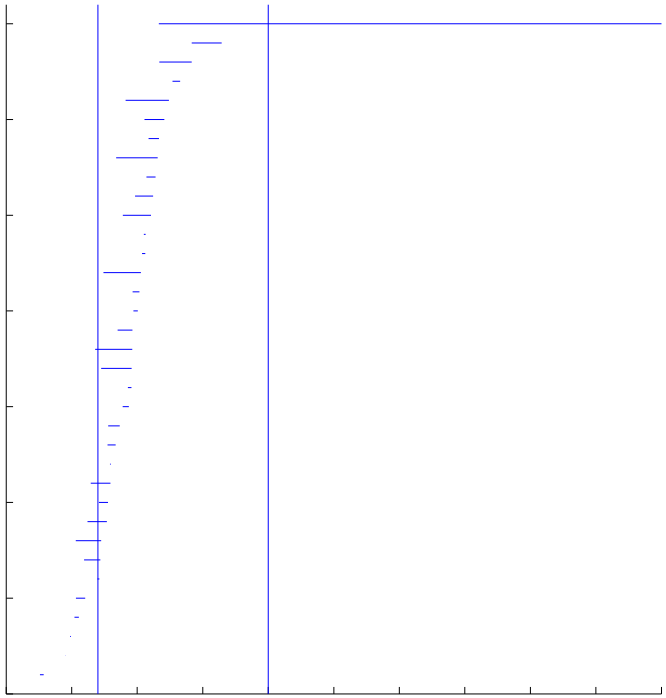
Parametrized circles

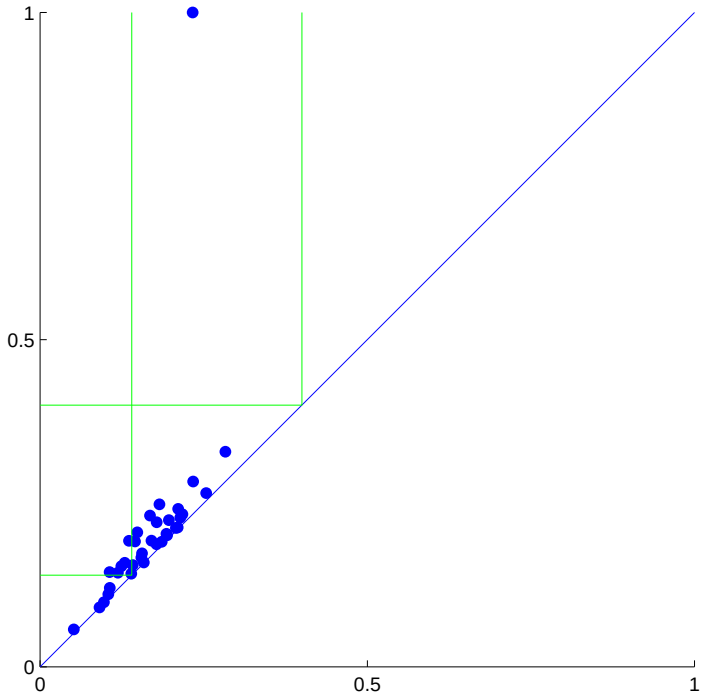


Parametrized circles

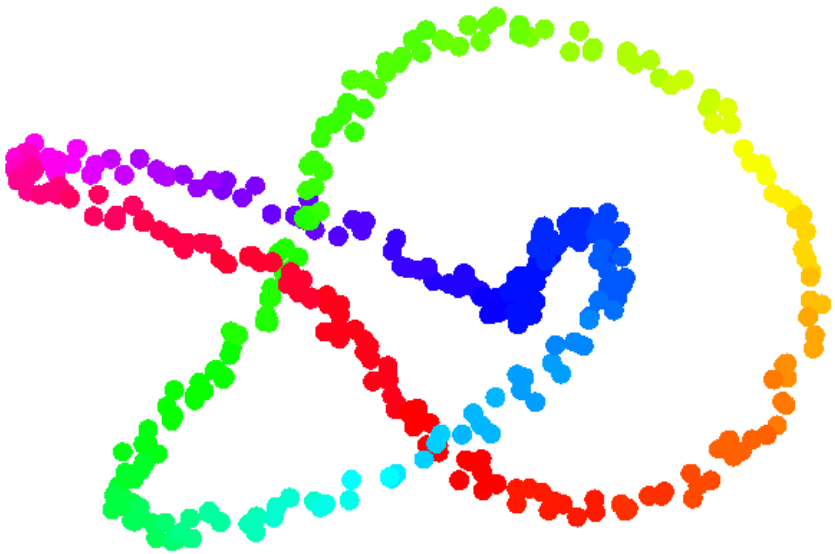


Parametrized circles

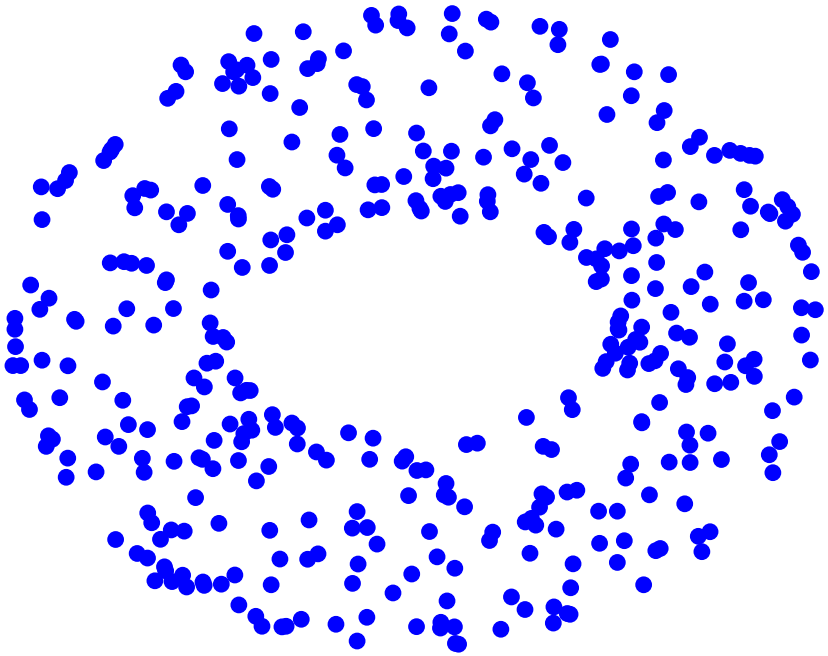




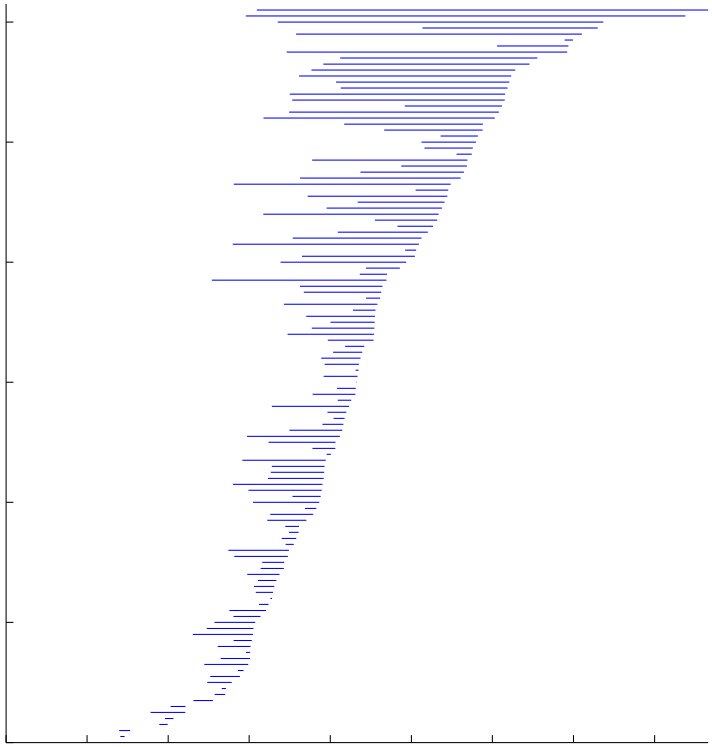
Knots



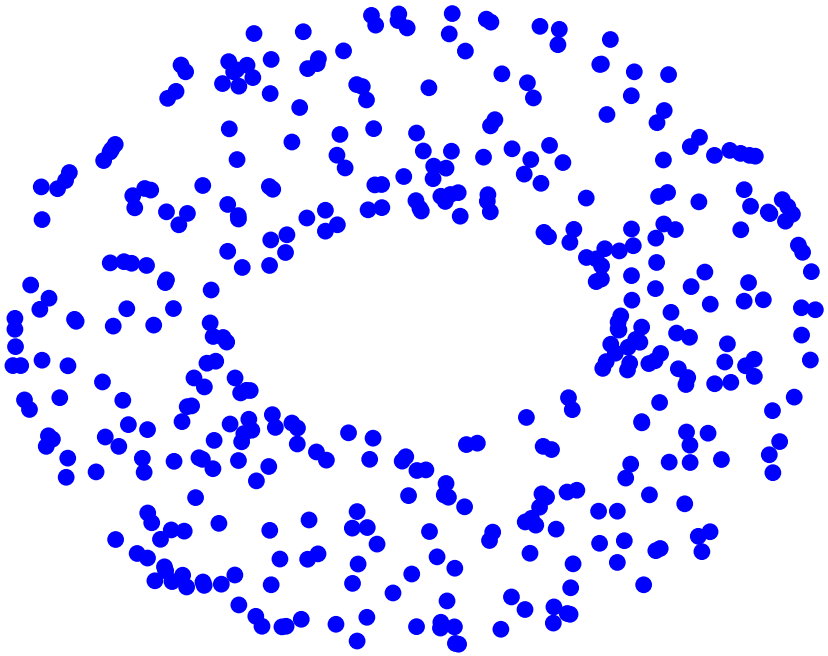
Torus



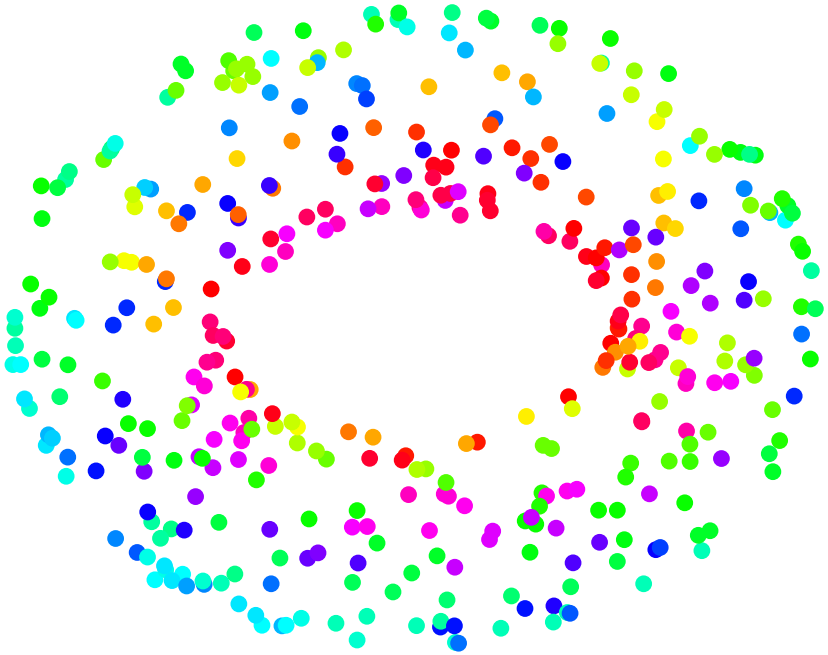
Torus



Torus

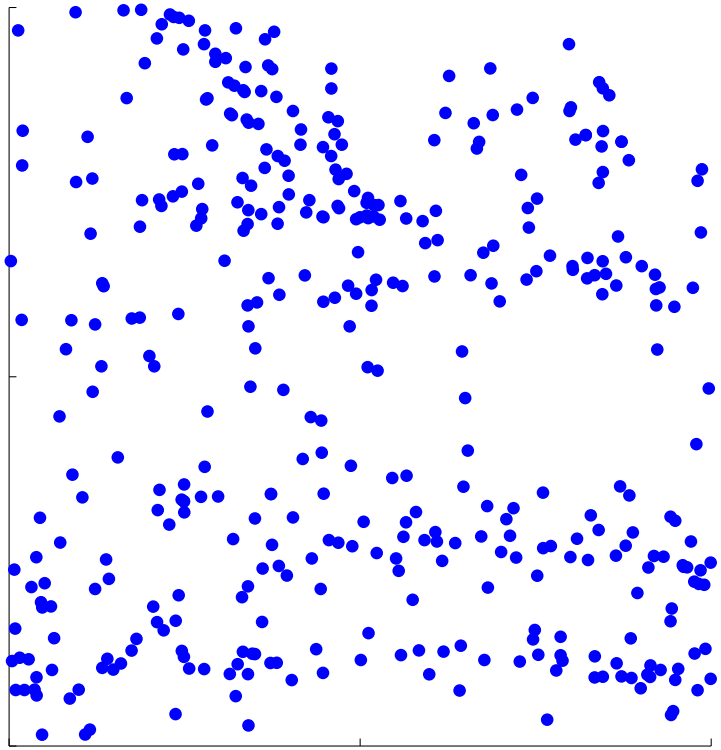


Torus



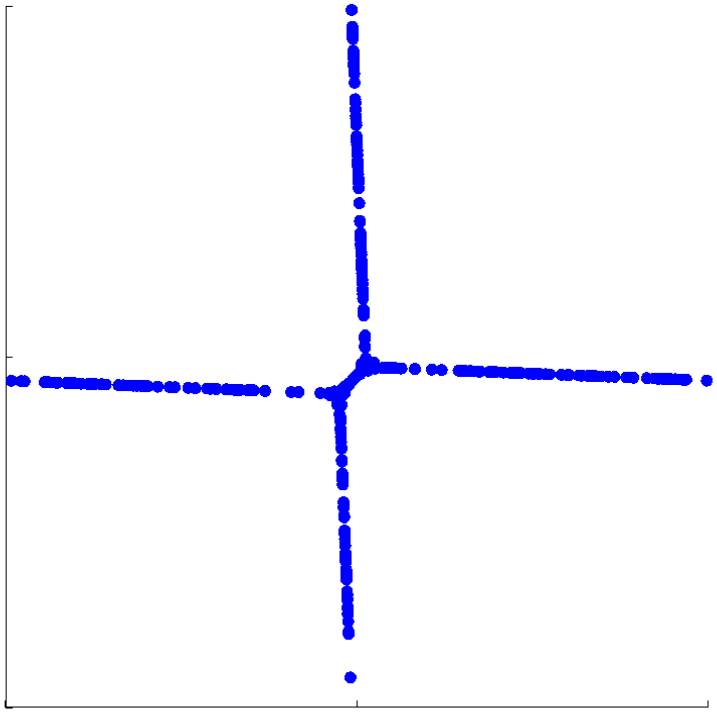
Torus

Torus



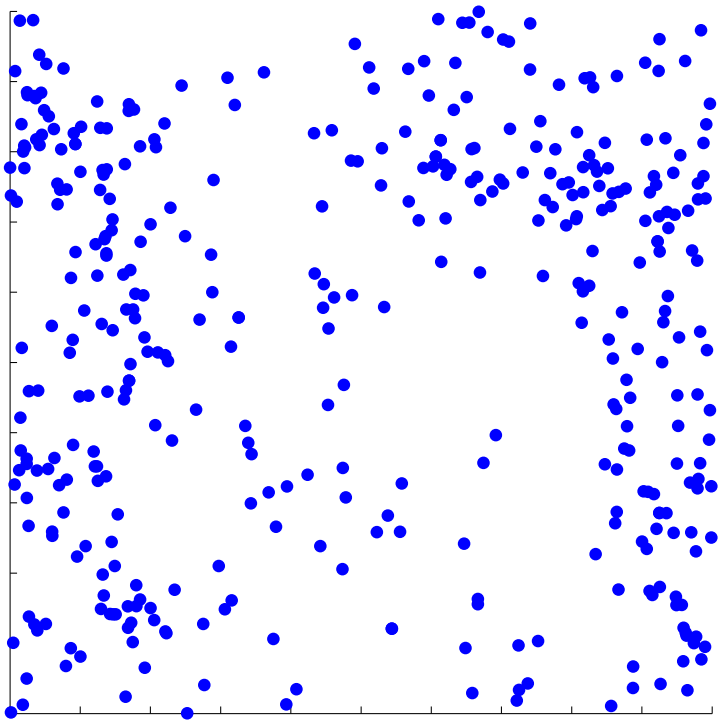
Correlation plot for this torus parametrization

Torus



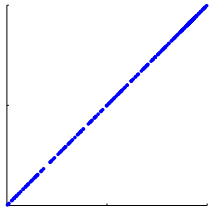
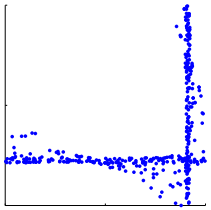
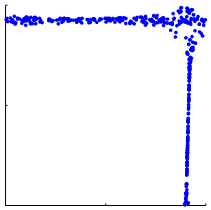
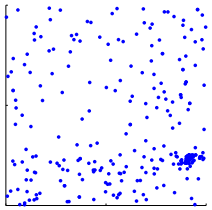
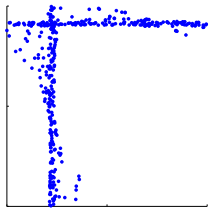
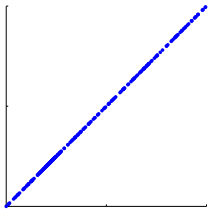
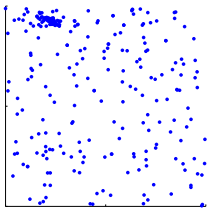
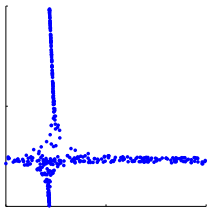
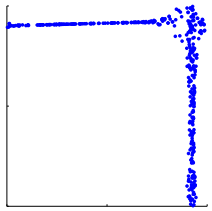
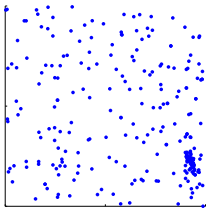
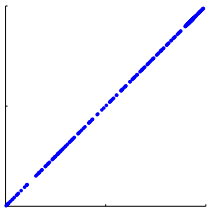
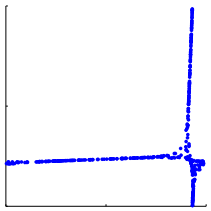
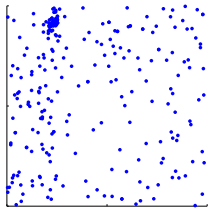
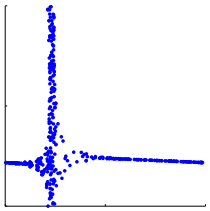
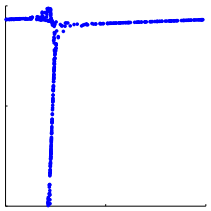
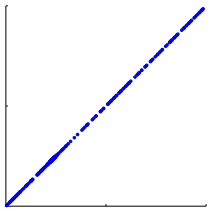
Correlation plot for a wedge of two circles

Torus

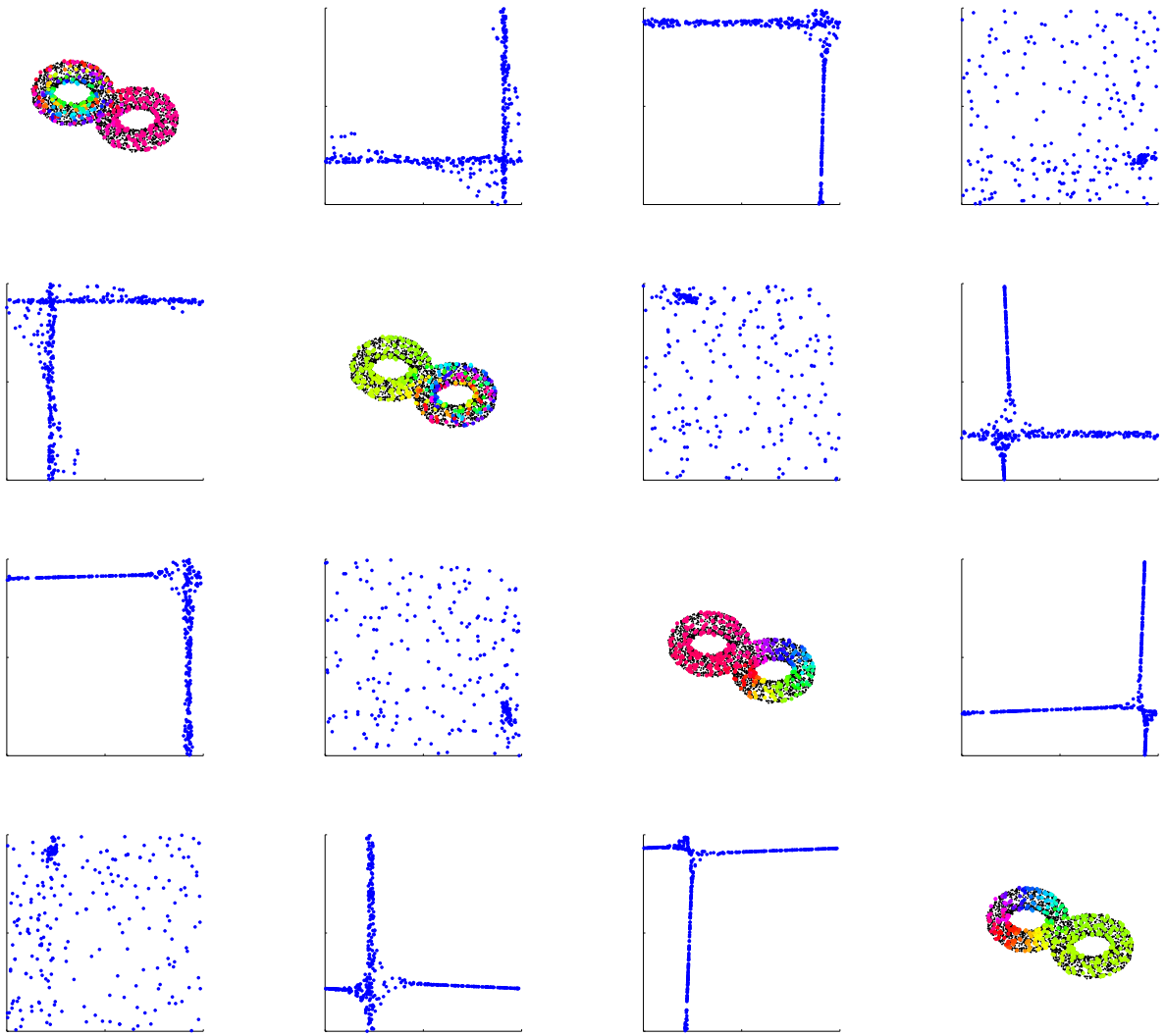


Correlation plot for the elliptic curve given by $y^2z - x^3 - xz^2 = 0$ in $\mathbb{C}P^2$. Metric is given by $d(p,q) = \tan^{-1}(p_x\overline{q_x} + p_y\overline{q_y} + p_z\overline{q_z})$

Pop quiz

Pop quiz – the Double Torus



Software and performance

The persistent cohomology algorithm is implemented in two places:

jPlex

- `http://comptop.stanford.edu/programs/jplex`
- Java based. Matlab integration.
- Will do cohomology in the next release.

Dionysus

- `http://www.mrzv.org/software/dionysus/`
- C++ based. Integrates with Python and CGAL.
- Still very much under development.
- Orders of magnitude faster than jPlex on these examples.

Timings for Dionysus

Example	# data points	# simplices	total time	time/size (μs/spx)
Noisy circle	200	23 475	0.10s	4.26
Torus knot	400	36 936	0.16s	4.33
Wedge of 2 circles	400	76 763	0.36s	4.69
2 disjoint circles	400	45 809	0.20s	4.37
Torus	400	61 522	0.29s	4.71
Elliptic curve torus	400	44 184	0.14s	3.17
Double torus	3 120	764 878	5.28s	6.90

An application: curve reconstruction

The following is joint work with Bei Wang and Sayan Mukhergee at Duke University.

Problem statement

Given a finite point sample P, with noise, from a non-singular curve $\gamma : I -> X$, can we recover the original curve γ? Or can we at least find a way that - up to speed adjustments for the curve - parametrizes the points we've sampled, giving an order of the points in the order the curve visits them.

This connects to work Mukhergee is doing on finding invariants of group actions from point samples.

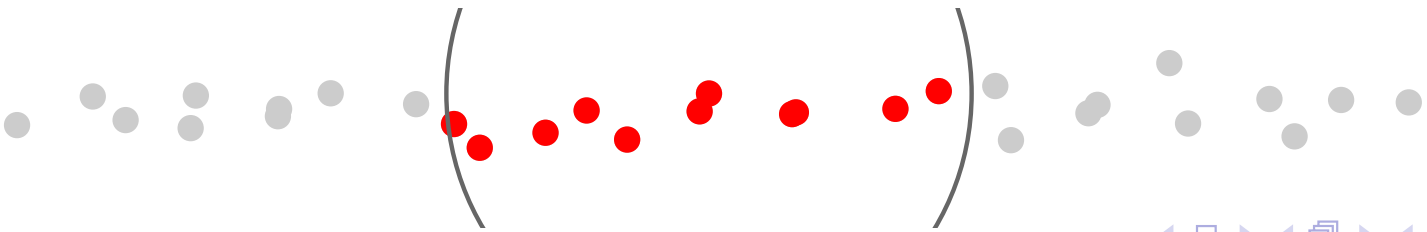
Approach: local cohomology and circular coordinates

- Curves are locally like $\mathbb{R}$. So we can parametrize locally, and glue parametrizations together.
- Once we settle on a neighbourhood of a sample point, we can compute *local cohomology* – we intersect the entire simplicial complex with our neighbourhood, and compute cohomology relative to everything outside this neighbourhood.
- Cohomology relative subspaces collapses the subspaces to a single point. If we do this outside a ball intersecting a curve, the ends of the curve passing through the boundary of the ball get identified.
- The result should have homotopy type a circle.
  We can parametrize this circle. And then unroll the parametrization to form a parametrized line.

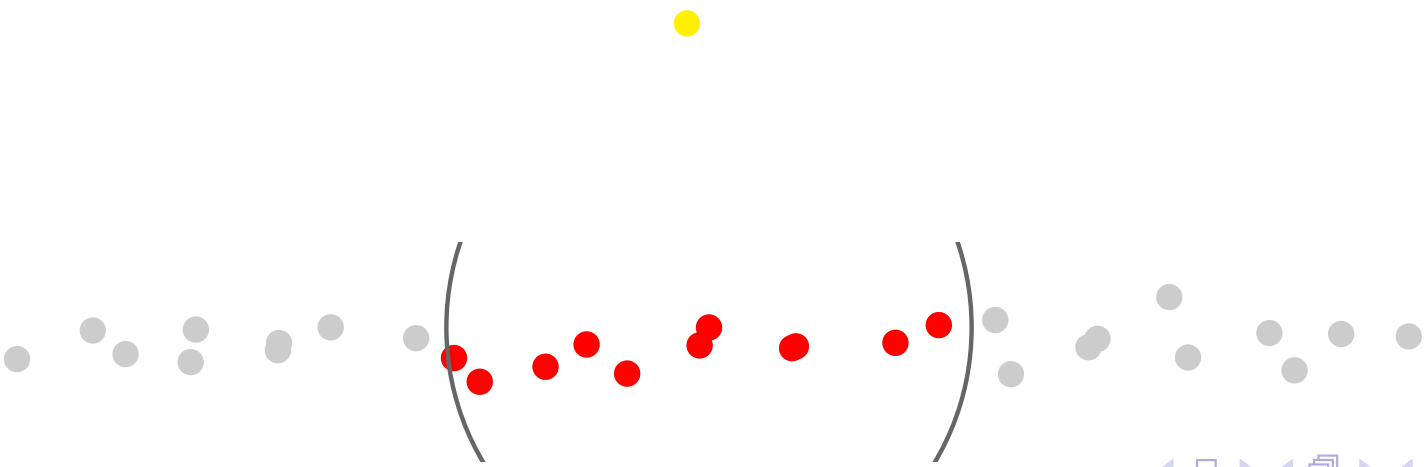
We wish to reconstruct the underlying curve from a predominantly 1-dimensional data set with noise.

We shall work locally - so we pick a point...

We shall work locally - so we pick a point...
...and a neighbourhood of this point.



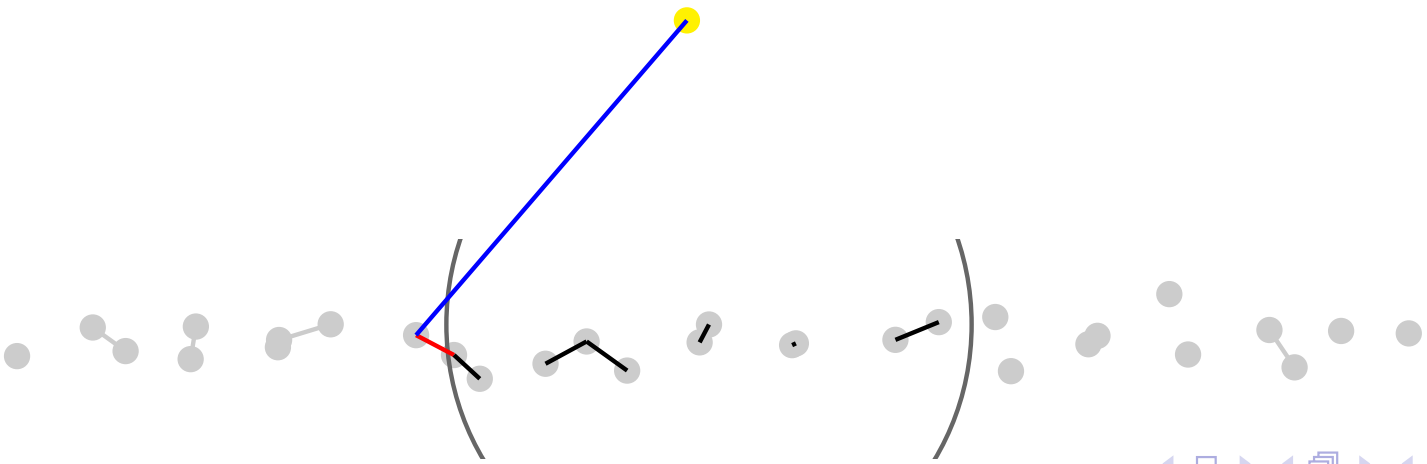
We then proceed to compute persistent cohomology - but with a twist. We introduce a "virtual point" outside our pointcloud, and any simplices that lead outside end up leading to this point instead of to the points outside.
This gives us cohomology relative to the complement of our chosen neighbourhood.



As we progress through the filtration, we collect the 2-skeleton, and compute the persistent degree 1 cohomology. As long as the interior of our chosen ball stays disconnected, nothing interesting will happen.

Once things start crossing the boundary of our ball, we replace any simplex crossing the boundary with one that connects the participating interior vertices with the virtual conepoint. Replaced simplices are in red while the replacing simplices appear in blue.

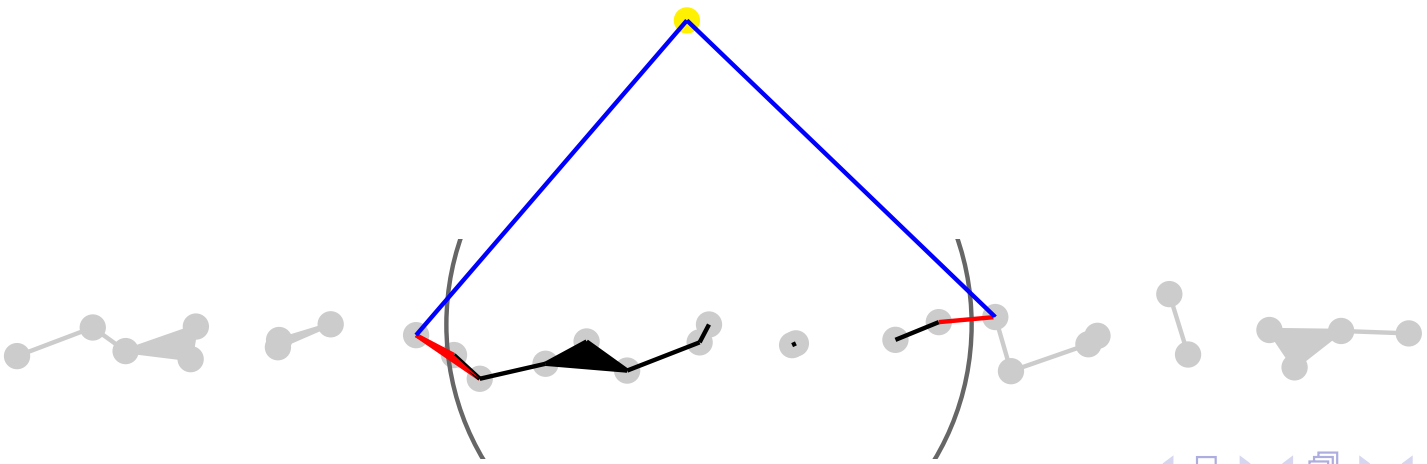
For $\varepsilon = 0.5$:



As we progress through the filtration, we collect the 2-skeleton, and compute the persistent degree 1 cohomology. As long as the interior of our chosen ball stays disconnected, nothing interesting will happen.

Once things start crossing the boundary of our ball, we replace any simplex crossing the boundary with one that connects the participating interior vertices with the virtual conepoint. Replaced simplices are in red while the replacing simplices appear in blue.

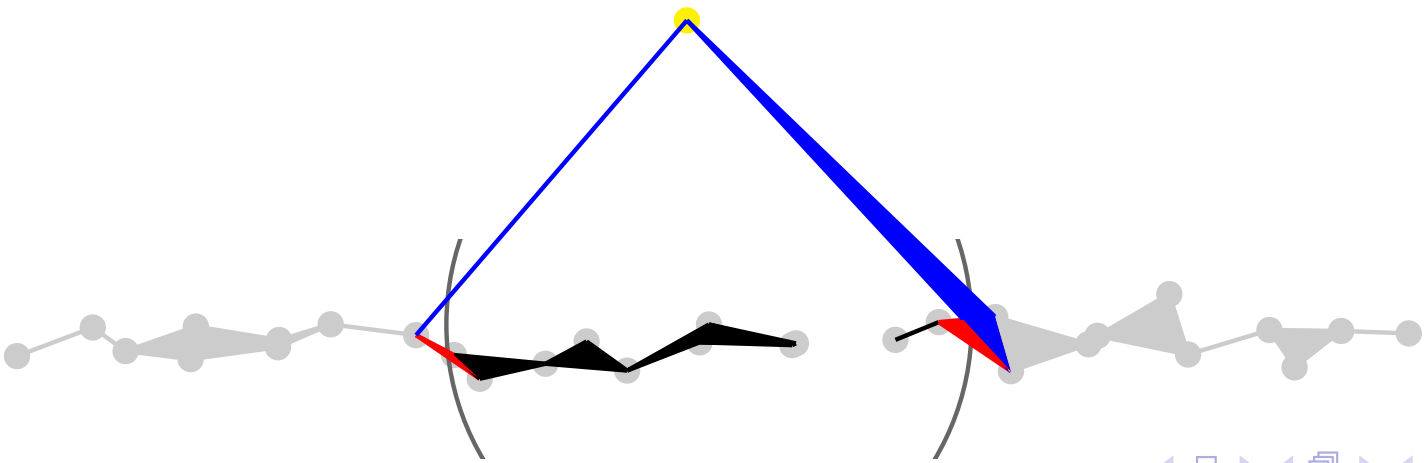
For $\varepsilon = 0.75$:



As we progress through the filtration, we collect the 2-skeleton, and compute the persistent degree 1 cohomology. As long as the interior of our chosen ball stays disconnected, nothing interesting will happen.

Once things start crossing the boundary of our ball, we replace any simplex crossing the boundary with one that connects the participating interior vertices with the virtual conepoint. Replaced simplices are in red while the replacing simplices appear in blue.

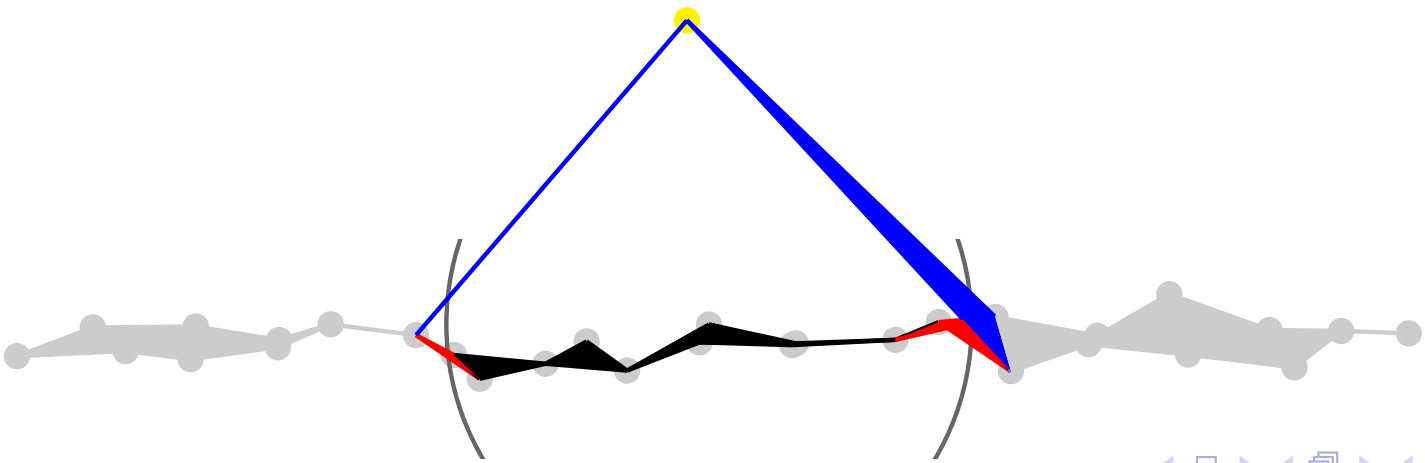
For $\varepsilon = 0.9$:



As we progress through the filtration, we collect the 2-skeleton, and compute the persistent degree 1 cohomology. As long as the interior of our chosen ball stays disconnected, nothing interesting will happen.

Once things start crossing the boundary of our ball, we replace any simplex crossing the boundary with one that connects the participating interior vertices with the virtual conepoint. Replaced simplices are in red while the replacing simplices appear in blue.

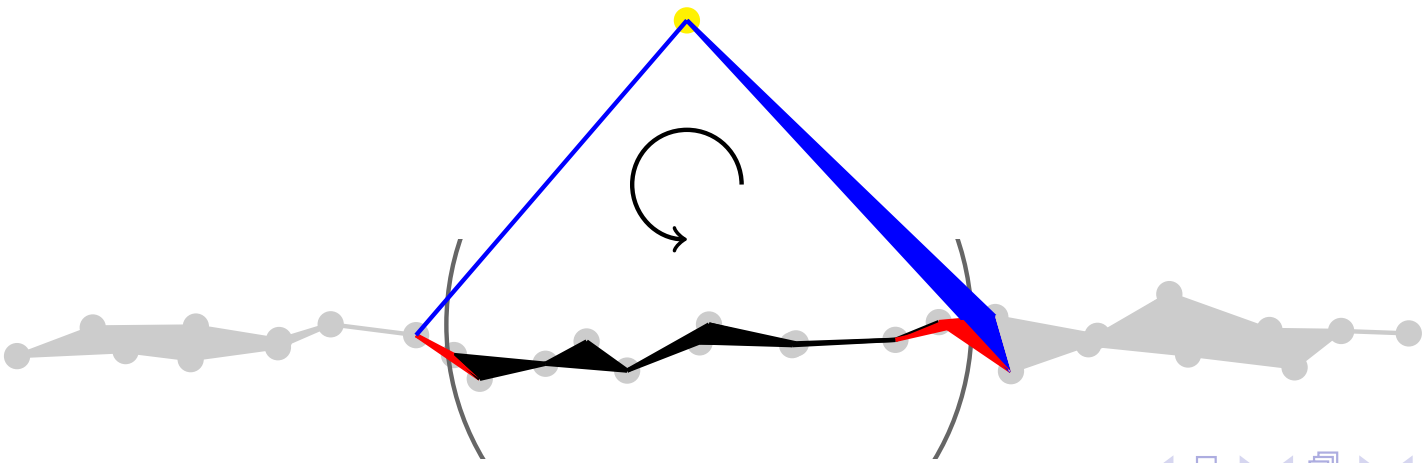
For $\varepsilon = 1.0$:



Now - here, at last, we actually see a cycle appearing!
This cycle will give rise to a non-trivial coclass in

$$H^1(X, X \setminus (X \cap B); \mathbb{Z})$$

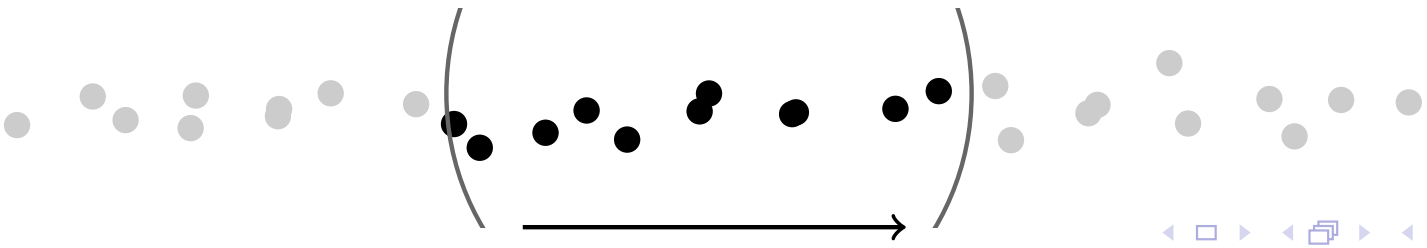
which we can use to give a coordinate function on the abstract simplicial complex depicted here.
Hence, we can translate the resulting coordinates so that $0.0 = 1.0$ is located at the cone point.



Now - here, at last, we actually see a cycle appearing!
This cycle will give rise to a non-trivial coclass in

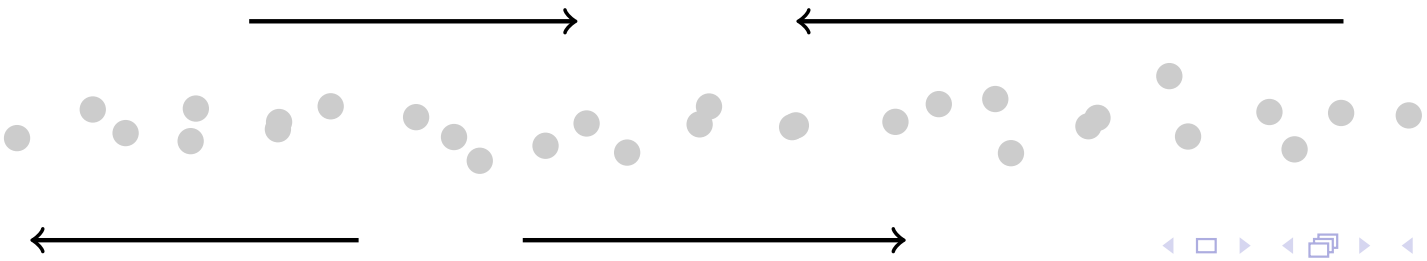
$$H^1(X, X \setminus (X \cap B); \mathbb{Z})$$

which we can use to give a coordinate function on the abstract simplicial complex depicted here.
Hence, we can translate the resulting coordinates so that $0.0 = 1.0$ is located at the cone point.
Then, the result is a continuous function $X \cap B \to [0, 1]$. In other words, we get a *real-valued* local coordinate function.



Doing this repeatedly, for an open cover of the whole point cloud, we can get overlapping parametrizations that we can patch together to give a parametrization of the entire curve the points were sampled from.

As long as we have a non-singular 1-manifold, this will work well. For singular cases, we need more work, but can still use this as a basis for the constructions.



Acknowledgements

Thanks are due for this to:

- Vin de Silva, Dmitriy Morozov – my collaborators
- Gunnar Carlsson
- The organizers
- ONR, DARPA-TDA, Pomona College and Stanford University – funding